Программирование приложений под операционную систему от Google

Android за 24 часа

Производственно-практическое издание

ПРОФЕСИОНАЛЬНЫЕ КОМПЬЮТЕРНЫЕ КНИГИ

Лорен Дэрси. Шейн Кондор

ANDROID ЗА 24 ЧАСА. ПРОГРАММИРОВАНИ Е ПРИЛОЖЕНИЙ ПОД ОПЕРАЦИОННУЮ СИСТЕМУ GOOGLE

Главный редактор И. Федосова
Зав. редакцией А. Баранов
Ведущий редактор И. Липкин
Художественный редактор М. Левыкин
Компьютерная верстка А. Попов

ООО «Рид Групп»
119021, Москва, ул, Россолимо, д. 17, стр. 1; тел.: 788-0075(76)

Свои пожелания и предложения по качеству и содержанию книг
Вы можете сообщить по эл. адресу: editorial@readgroup.ru

Издание осуществлено при техническом содействии ООО «Издательство АСТ»

Подписано в печать 27.04.2011. Формат 70х100/ 16.
Уел. печ. л. 37,7. Печать офсетная. Бумага писчая.
Тираж 3 ООО экз. Заказ № 5092.

Отпечатано с готовых диапозитивов
в типографии ООО «Полиграфиздат»
144003, г. Электросталь, Московская область, ул. Тевосяна, д. 25

Дэрси Л.
Д 94 Android за 24 часа. Программирование приложений под операционную систему Google/ ДэрсиЛ., КондерШ. — М.: Рид Групп, 2011. — 464 с. — (Профессиональные компьютерные книги).
ISBN 978-5-4252-0318-2

Если у вас есть мобильный телефон на базе платформы Android и несколько хороших идей по разработке мобильного приложения, эта книга подойдет для начального обучения.

Если вы программист, стремящийся освоить мобильные технологии, или предприниматель, нуждающийся в разработке успешного приложения, — эта книга для вас.

Если у вас лишь базовые знания о языке программирования Java, то Android, помимо прочего, — прекрасная платформа для его изучения.

Книга «Android за 24 часа» разбита на 24 урока продолжительностью по одному часу. Каждое новое задание основывается на предыдущих уроках, таким образом, с освоением каждого нового часа вы будете итеративно совершенствовать свое приложение.

По окончании курса вы будете в состоянии спроектировать и разработать собственное полнофункциональное приложение Android.

УДК 004.4
ББК 32.973.26-018.2

ISBN 978-5-4252-0318-2

СОДЕРЖАНИЕ

ОБ АВТОРАХ

Лоурен Дэрси (Lauren Darcey) занимает пост руководителя отдела технического управления небольшой компании, специализирующейся на мобильных технологиях, таких как Android, iPhone, Blackberry, Palm Pre, BREW и J2ME. Лоурен — признанный специалист по архитектуре систем управления предприятием и разработке коммерческих мобильных приложений, имеющий более чем двадцатилетний опыт в области профессионального программного обеспечения. Лоурен получила высшее образование специалиста по вычислительной технике и информатике в университете города Санта-Круз, штат Калифорния.

Большую часть своего свободного времени она тратит на путешествия по миру со своим мужем, также фанатом мобильных технологий и любительской фотосъемки. Работы Лоурен опубликованы в книгах и газетах по всему миру. В Южной Африке ей довелось плавать рядом большими четырехметровыми акулами, быть окруженной стадом разозленных гиппопотамов и испытать на себе агрессию африканских слонов. Она пережила нападение обезьян в Японии, спаслась от двух голодных львов в ущелье (Кения), испытала долгую мучительную жажду в Египте, избежала беспорядков, связанных с государственным переворотом в Таиланде, прошла через швейцарские Альпы и знаменитые пивные фестивали Германии. Ей довелось провести несколько ночей в полуразрушенных средневековых замках Европы и даже спастись с отколовшегося айсберга в Исландии.

Шейн Кондер (Shane Conder) имеет большой опыт в разработке программного обеспечения. Последние десять лет он специализируется на мобильных технологиях и разработке встроенных систем управления. Он спроектировал и разработал множество коммерческих приложений на платформах BREW, J2ME, Palm, Windows Mobile и Android — некоторые из них установлены миллионами пользователей на свои мобильные телефоны по всему миру. Шейн написал большое количество материалов об индустрии мобильных технологий и распространенных платформах разработки в своем официальном блоге, которые стали очень популярны в Сети. Шейн также получил высшее образование специалиста по вычислительной технике и информатике в калифорнийском университете.

Шейн, как фанат новинок рынка мобильных устройств, всегда имеет при себе последнюю модель дорогого смартфона или ноутбука. Он часто проводи! время на популярных веб-ресурсах за изучением новых технологий, таких как Amazon Web Services, Android, iPhone, Google App Engine и др., развивая свои креативные способности. Он также любит

путешествовать по миру со своей женой Лоурен, несмотря на ее экстремальные желания нырять рядом с большими белыми акулами и совершать опасные прогулки со львами в Кении. Шейн признает свою вину в том, что они были окружены опасными обезьянами в Японии, что ему было весело фотографировать свою жену, когда она застряла на айсберге в Исландии и что он так и не потрудился написать хорошую автобиографию для книги.

Другие публикации авторов

Авторами этой книги была также написана книга «Android Wireless Application Development», входящая в серию «Developer's Library», и множество технических статей для сайтов developer.com, informlT.com, а также их собственного блога android-book.blogspot.com.

ПОСВЯЩЕНИЕ

Уважаемые бабушки и дедушки по всему миру, особенно те из вас, кто любит печь вкуснейшие пироги и смотреть по вечерам передачу «Особо опасен». Вы обеспечивали нам прекрасную моральную поддержку и совершенно не доставляли нам какого-либо беспокойства. И если вы не помните, что мы вам об этом уже говорили, не беспокойтесь — мы повторим эти слова еще раз.

БЛАГОДАРНОСТИ

Эта книга никогда бы не была написана без помощи и поддержки, которую мы получили от подбадривающих нас людей, включая нашу команду редакторов, других сотрудников, друзей и семью.

Мл протяжении всего проекта наша команда редакторов из издательства Sams Publishing (группа Pearson) проделала огромную работу. Особая наша ошиодарпость Трине МакДональд (Trina MacDonald), Оливии Бейсегио (Olivia Itasegio) и Бетси Харрис (Betsy Harris). Благодаря нашему замечательному техническому обозревателю Джонатану Джексону (Jonathan Jackson) мы уверены, что в этой книге представлена достоверная и технически правильная информация. И, наконец, мы хотим поблагодарить наших друзей и членов семьи, поддержавших нас в трудную минуту, когда мы вынуждены были продолжать работу над книгой, несмотря на инфекционное заболевание, из за чего мы должны были писать книгу кухне. (Да, вы можете разрабатывать приложении Android в любом месте.) Мы хотим также выразить отдельную благодарность Лиз Рейд (Liz Reid), Гаю Грейсону (Guy Grayson), семье Ленц (Lenz) (особенно Томасу и Патрику), Шоши Брауну (Shoshi Brown) и его семье (особенно Якобу), семье Бэджеру (Badger) (особенно Ви-Ви и Нолии), Ричарду де Кэстонгрену (Richard de Castongrene), Эшер Сидцику (Asher Siddiqui), Энтони Шэфферу (Anthony Shaffer), Спенсеру Нэссару (Spencer Nassar) и Мэри Томпсон (Mary Thompson) за их поддержку и ободрение.

НАМ ВАЖНО ЗНАТЬ ВАШЕ МНЕНИЕ

Как читатель нашей книги, именно вы — наш главный критик, и только вы можете оценить нашу работу по достоинству. Мы ценим ваше мнение и хотим знать, что мы сделали неправильно, что мы можем сделать лучше, нам важны любые другие ваши пожелания.

Пожалуйста, укажите в письме название книги и ее авторов, ваше имя и контактный телефон или адрес электронной почты.

E-mail: **baranov@readgroup.ru**

Почтовые реквизиты:

Кому: Москва, 119121, ул. Россолимо, 17, Издательство «Рид Групп».

ВВЕДЕНИЕ

Платформа Android в последнее время стала очень популярна на рынке мобильных устройств и получила широкое признание во всем мире. Эта платформа постоянно совершенствуется, улучшается функциональность комплекта разработки программного обеспечения (SDK, Software Development Kit), поддержка мобильными телефонами, расширяются ее возможности. В магазинах представлено (и уже используется потребителями, разумеется) большое разнообразие мобильных устройств на базе платформы Android. Речь не только о мобильных телефонах: платформу устанавливают на нетбуки, планшетные устройства (такие как ARCHOS 5), e-books-ридеры (например, от фирмы Barnes & Noble), цифровые рамки для фотографий и другие виды бытовой электроники. Существуют даже прототипы устройств, соединяющих микроволновую печь и стиральную машину на базе этой платформы. (Почему бы и нет? См. bit.ly/bGqmZp.) Операторы мобильной связи и крупные сети магазинов относятся к платформе с большим оптимизмом и тратят миллиарды на ее продвижение — можно вспомнить рекламную кампанию Droid, проведенную американским оператором мобильной связи Verizon.

За короткий срок Android прошла путь от системы для энтузиастов до серьезной платформы, способной соревноваться с более традиционными системами (например, iPhone).

Но давайте не будем отвлекаться на выяснение того, какая платформа лучше. Вы потратите время впустую, если думаете, что существует лишь одна платформа, которая по всем характеристикам лучше остальных. Люди во тем мире выбирают разные телефоны для работы в разных стандартах связи ((DMA, GSM) и по разным причинам (цена, доступность, качество связи, набор функций, внешний вид, удобство, совместимость). В этом споре невозможно выявить победителя.

Мы имеем четкое представление о плюсах и минусах каждой из современных мобильных платформ. Ни одной из них мы не отдавали предпочтения; v каждой платформы есть явные преимущества над остальными, и эти преимущества могут быть усилены.

Сложность заключается в том, какую платформу предпочтительнее использовать в каждом конкретном проекте. Иногда бывает необходимо адаптировать проект ко всем платформам. Но на своем опыте мы убедились, что оптимальна именно платформа Android: приложения для нее разрабатываются быстро и дешево, она доступна миллионам потенциальных пользователей во всем мире, и у нее меньше ограничений.

Платформа Android относительно молода, и у нее еще большой потенциал для развития. Это означает, что неизбежны частые обновления SDK, появление новых устройств на

рынке и необходимо постоянное отслеживание всех событий, происходящих в мире Android.

Другими словами, вам предстоит преодолеть непростой путь, но еще есть время, чтобы перейти на побеждающую сторону, написать несколько превосходных приложений и сделать себе на этом имя.

Так что давайте приступим.

ДЛЯ КОГО ЭТА КНИГА?

Если у вас есть мобильный телефон на базе платформы Android и несколько хороших идей по разработке мобильного приложения, эта книга подойдет для начального обучения. Программист ли вы, стремящийся освоить мобильные технологии, или предприниматель, нуждающийся в разработке успешного приложения, — эта книга для вас.

Если у вас лишь базовые знания о языке программирования Java (понимание классов, методов, принципов наследования и т.д.), то Android, помимо прочего, — прекрасная платформа для его изучения. Мы старались избегать подробного описания Java, поэтому, если вы только начинаете свой путь в программировании, то должны прочесть первые главы какого-нибудь самоучителя по этому языку или пройти курсы обучения онлайн и получить достаточно знаний о Java, чтобы понимать, о чем идет речь в этой книге.

Мы предполагаем, что вы уже чувствуете себя уверенно при установке приложений (например, Eclipse, Java JDK и Android SDK), а также инструментов и драйверов (для доступа к мобильному телефону через USB) на компьютер. Мы предполагаем также, что вы владеете функциями телефона на базе Android достаточно хорошо, чтобы запускать приложения и совершать другие действия. При прочтении книги не потребуется каких-либо специальных знаний о беспроводных технологиях.

СТРУКТУРА КНИГИ

Книга разбита на 24 простых урока продолжительностью по одному часу. По окончании курса вы спроектируете и разработаете полнофункциональное приложение Android с поддержкой сети и геолокационных сервисов (LBS, Location-Based Service), дополненное социальными функциями. Каждый новое задание основывается на предыдущих уроках, таким образом, с освоением каждого нового часа вы будете итеративно совершенствовать свое приложение.
Эта книга состоит из шести частей.

- **Часть I. Основные принципы Android.**

  В части I вы получите базовые знания о платформе Android, познакомитесь с Android SDK и сопутствующими инструментами, установите инструменты разработки и напишете свое первое приложение Android. В части I даны также

основы проектирования, необходимые для создания приложений Android, включая структуру и конфигурирование приложений, а также использование ресурсов приложения, таких как строки, графика и компоненты пользовательского интерфейса.

- **Часть II. Создание базовой основы приложения.**

  Во второй части вы начнете разработку каркаса игрового приложения, которое будет служить практическим примером в следующих часах книги. Вы начнете с анимированного экрана-заставки, затем экранов главного меню, меню настроек, справки и таблицы рекордов. Вы изучите основы проектирования пользовательского интерфейса, узнаете, как обрабатываются введенные пользователем данные и как выводятся диалоговые окна на экран. В итоге вы реализуете базовую логику интерфейса игры.
- **Часть III. Совершенствование приложения средствами Android.**

  В части III вы ближе познакомитесь с Android SDK, добавите больше специальных возможностей к созданному вами приложению. Вы узнаете, как работать с графикой и встроенной камерой, как усовершенствовать LBS, как адаптировать ваше приложение для работы в сети и дополнить его социальными функциями.
- **Часть IV. Локализация, настройка для различных устройств и тестирование приложения Android.**

  В части IV вы узнаете, как настроить ваше приложение под различные характеристики мобильных телефонов, включая размер экрана и поддержку иностранных языков. Вы узнаете также о различных способах тестирования мобильных приложений.
- **Часть V. Публикация вашего приложения.**

  В части V вы узнаете о тех действиях, которые необходимо предпринять для правильной подготовки и опубликования приложения на Android Market.
- **Часть VI. Приложения.**

  В части VI вы найдете полезные справочные данные для настройки среды разработки Android с помощью Eclipse IDE, а также информации о дополнительных материалах, таких как исходные коды программ.

ЧТО ЕСТЬ И ЧЕГО НЕТ В ЭТОЙ КНИГЕ

Несмотря на то, что в этой книге сделан акцент на Android SDK версии 2.1 большинство примеров были протестированы на мобильных телефона) с различными версиями Android SDK.

Android SDK обновляется очень часто (каждые несколько месяцев). Мы учли эту особенность, когда выбирали возможности SDK, которые необходимо подчеркнуть, чтобы обеспечить максимальную прямую и обратную совместимость. При необходимости мы будем обращать внимание на т< области, где определенная версия Android SDK влияет на возможности и функциональность, доступные разработчику.

Эта книга написана в стиле учебного пособия для начинающих. Если же вы ищете исчерпывающую информацию о разработке приложений Android в стиле «поваренной

книги» и с более полным обзором всех возможностей! платформы Android, мы рекомендуем вам другую книгу об Android - «Android Wireless Application Development», входящую в серию «Developer' Library» издательства Addison-Wesley.

ПРОГРАММНАЯ СРЕДА

Код для этой книги был написан с использованием следующих программных средств:

- операционные системы Windows 7 и Mac OS X 10.6;
- Eclipse Java IDE версии 3.5 (Galileo);
- плагин Eclipse JDT и Web Tools Platform (WTP);
- набор программ Sun Java SE (JDK) версии 6.18;
- Android SDK версии 2.1 (разработана и протестирована на нескольки версиях SDK);
- различные мобильные телефоны на базе платформы Android (Androi SDK 1.6, 2.0.1 и 2.1).

СОГЛАШЕНИЯ, ПРИНЯТЫЕ КНИГЕ

В этой книге представлено несколько типов врезок с дополнительной информацией:

Знаете ли вы что...

Здесь приводятся полезная информация и подсказки, относящиеся к текущему тексту.

Кстати

Здесь приводится дополнительная информация, которая может быть интересной.

Внимание!

Здесь приводятся замечания и предостережения об опасных ситуациях, с которыми вы можете встретиться, и о том, как их избежать.

В этой книге используются следующие соглашения, относящиеся к коду программ:

- Код и фрагменты кода напечатаны моноширинным шрифтом.
- Код, следующий за символом »*, означает, что он принадлежит той же самой строке, что и предшествующий код.
- Обработка исключений и проверка на ошибки часто удаляются из печатных вариантов кода для наглядности и сохранения разумного объема книги.

В этой книге используются следующие соглашения для пошаговых инструкций и пояснений:

- Основное приложение, представленное в этой книге, разрабатывается постепенно. Это значит, что сначала объясняется новое понятие, и каждый момент, связанный с ним, подробно обсуждается. По мере того, как мы будем переходить к более сложным темам, вы должны освоить большую часть пройденного материала,

потому что информация нигде не повторяется. В некоторых случаях вы будете применять на практике знания, подробнее о которых будет рассказано в одном из следующих часов.

- Предполагается, что вы читаете книгу по порядку. По мере освоения книги вы заметите, что для упрощения работы над основным примером мы по объясняем каждый шаг в мельчайших подробностях. Например, если на экране необходимо создать три кнопки, мы шаг за шагом объясним вам, как сделать первую, но оставшиеся вы должны создать самостоятельно. В последующих уроках по различным темам мы можем просто попросить вас реализовать некоторые кнопки аналогичным образом для другого экрана интерфейса.
- Там, где встречаются пошаговые команды меню, мы разделяем каждый пни символом =>. Например, если вам необходимо открыть новый документ, вы увидите бы текст «Выберите команду меню **File => New Document** (Файл > Новый Документ)».

ЧАСТЬ I. ОСНОВНЫЕ ПРИНЦИПЫ ANDROID

Час 1. НАЧАЛО РАБОТЫ С ANDROID

Вопросы, рассматриваемые в этом часе:

- **краткая история платформы Android;**
- **особенности работы в среде разработки Eclipse;**
- **как создаются проекты Android;**
- **как запускаются приложения и проводится их отладка.**

Android — первая открытая, бесплатная и полноценная мобильная платформа. Комплект разработки программного обеспечения полностью удовлетворяет требованиям разработчиков благодаря профессиональным инструментам для разработки мощных, многофункциональных приложений. Исходный код платформы открыт, что могут оценить разработчики, предпочитающие проверенные временем открытые стандарты. И, самое главное, разработчики могут пользоваться всеми программными средствами практически бесплатно. (Скромная плата необходима лишь для публикации готовых приложений сторонними дистрибьюторами, такими как Android Market.) У разработчиков Android есть множество путей для распространения и коммерциализации своих приложений.

ПРЕДСТАВЛЕНИЕ О ПЛАТФОРМЕ ANDROID

Чтобы понять, что общего у Android с другими мобильными технологиями, дапайте разберемся в причинах появления этой платформы.

Google и Альянс разработчиков открытых стандартов мобильных устройств

В 2007 году группой производителей мобильных устройств, провайдеров беспроводной связи и разработчиков программного обеспечения (особенно Google) был сформирован Альянс разработчиков открытых стандартов мобильных устройств (OHA, Open Handset Alliance) с целью создания беспроводных технологий следующего поколения. В отличие от существующих платформ, новая платформа должна быть бесплатной, основанной на открытых стандартах, позволяющих снизить затраты на разработку и увеличить прибыль. Разработчики мобильных приложений должны также иметь полный доступ к функциям телефона для создания инновационных приложений.

Поскольку пропристарные платформы, такие как RIM BlackBerry и Apple iPhone, получили широкое распространение, сообщество разработчиков мобильных технологий внимательно следит за событиями вокруг этой перспективной платформы.

Появление платформы Android

И 2008 году Альянс разработчиков открытых стандартов мобильных устройств анонсировал платформу Android и объявил о запуске открытого бета-тестирования. Как и большинство подобных проектов, платформа Android прошла через серию ревизий. В итоге было выпущено несколько предрелизных версий Android SDK. Первым мобильным телефоном на базе платформы Android стал T-Mobile G1, продажи которого начались в

конце 2008 года. В течение 2009 года все больше и больше мобильных телефонов, а также других типов устройств, основанных на платформе Android, достигли мировых рынков. На данный момент произведено более 65 моделей мобильных телефонов, доступных в продаже по всему миру. Помимо них есть многочисленные планшеты Android и устройства для чтения электронных книг, десятки готовящихся к выпуску устройств и даже бытовая электроника под управлением Android. Причем скорость, с которой новые мобильные устройства на базе Android достигают мировых рынков, продолжает увеличиваться. В Соединенных Штатах все основные поставщики электронной техники уже включают устройства на базе платформы Android и свои линейки продуктов.

Компания Google с самого начала способствовала развитию Альянса разработчиков открытых стандартов мобильных устройств. Интернет-гигант фактически владеет

Рис. 1.1 Логотип Android

проектом Android и руководит сообществом разработчиков Android **(developer.android. com).** Этот веб-сайт может послужить для вас отличным источником для загрузки Android SDK, получения последней документации по платформе, а также для общения с другими разработчиками на форумах. Компания Google владеет также самым популярным сервисом по продаже приложений Android конечным пользователям — Android Market. На рис. 1.1 изображен логотип Android, который представляет собой силуэт небольшого зеленого робота.

Легкий и бесплатный процесс разработки

Простота и дешевизна разработки всегда являлись неоспоримыми преимуществами. Теперь это относится и к мобильным приложениям. Раньше разработка приложения с поддержкой беспроводной связи, с ее непомерно дорогими компиляторами и программами для разработчиков была довольно затратной по сравнению с разработкой настольных приложений. В этом смысле Android ломает устоявшиеся стереотипы. В отличие от других мобильных платформ, разработка приложений Android почти ничего не стоит'.

Android SDK и сопутствующие инструменты есть в свободном доступе на официальном веб-сайте разработчиков Android — **developer.android.com.** Программа Eclipse, также доступная для свободного пользования, стала самой популярной интегрированной средой разработки (IDE, Integrated Development Environment) для разработки приложений Android. Чтобы облегчить процесс разработки приложений Android, вы можете воспользоваться многофункциональным плагином для Eclipse, который доступен на сайте разработчиков Android.

Итак, мы объяснили причину низкой стоимости, теперь немного слов о легкости разработки приложений Android. Поскольку приложения Android пишутся на языке Java, разработчикам должны быть знакомы многие из пакетов, поставляемых в комплекте с Android SDK, например, такие как **java.net.** Надеемся, что вы будете приятно удивлены, насколько просто научиться работать с Android.
Итак, давайте начнем.

ЗНАКОМСТВО С ECLIPSE

Начнем с создания простейшего Android-приложения «Hello, World», отображающего на экране пользователя соответствующий текст. По мере сознания вы получите общее представление о среде Eclipse. Кроме того, вы узнаете о возможностях, предлагаемых плагином ADT (Android Developmeni Tools). Плагин ADT — это комплекс возможностей для разработки, компиляции, сжатия и развертывания приложений Android:

- **Android Project Wizard** (Мастер проектов Android), генерирующий все основные файлы проекта;
- специфичные для Android редакторы ресурсов;
- **Android SDK and AVD Manager** (Менеджер Android SDK и виртуальны устройств);
- перспектива DDMS для мониторинга и отладки приложений Android; интеграция с системой ведения журнала Android LogCat;
- автоматизированная сборка и развертывание приложений на эмуляторах Android и физических мобильных устройствах;
- сжатие приложения и инструменты лицензирования кода для опубликования.

УСТАНОВКА ANDROID SDK И ДРУГИХ ИНСТРУМЕНТОВ

Всю информацию об установке и настройке вашего компьютера для разработки приложений Android вы можете найти в приложении А. Вы должны будете установить и сконфигурировать Eclipse, Android SDK и плагин ADT для Eclipse. Вы, возможно, также должны будете установить драйверы USB для мобильных устройств Android, которые вы будете использовать в процессе разработки.

Теперь давайте рассмотрим некоторые из этих возможностей подробнее.

Создание проектов Android

Android Project Wizard (Мастер проектов Android) создает все необходимые файлы для приложения Android. Чтобы создать новый проект, запустите Eclipse и сделайте следующее:

1. Выберите команду меню **File** => **New => Android Project** (Файл => Новый => проект Android) или нажмите на значок создания проекта Android в виде папки с буквой «а» и знаком «плюс» ( ) на панели инструментов Eclipse.

ВНИМАНИЕ

Когда вы впервые попробуете создавать проект Android, вам, возможно, придется выбрать другую команду: **File** => **New** => **Project** (Файл =* Новый => Проект), а затем в меню открывшегося диалогового окна — **Android => Android Project** [Android => Проект Android). После создания первого проекта в списке типов проектов Eclipse появится новый тип, и вы сможете пользоваться командой, приведенной в первом пункте.

2. Введите название проекта. Например, проект **Droidl.**

3. Выберите место, где будет храниться проект. Поскольку это новый проект, установите переключатель в положение **Create new project in workspace** (Создать в рабочей области новый проект). Установите флажок **Use default location** (Путь по умолчанию), если он снят.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Гели вы хотите расположить файлы проекта в другой папке, просто снимите флажок **Use default location** (Путь по умолчанию) и укажите желаемую папку.

4. Выберите версию платформы для нового приложения. В большинстве случаев следует использовать версию Android, устанавливаемую на устройствах, используемых вашей целевой аудиторией, и отвечающую потребностям вашего приложения. Вели вы планируете использовать дополнения от Google (например, Google Maps), обращайте также внимание на версию Google API, которая соответствует выбранной платформе. Например, платформе Android версии 2.1 соответствует API Level 7.

5. Укажите имя приложения. Это имя будут видеть пользователи. В нашем случае приложение лучше назвать Droid #1.

6. Введите имя пакета, удовлетворяющее требованиям пространства имен стандартных пакетов Java. Поскольку весь код в этой книге соответствует требованиям пространства имен com.androidbook.*, можете указать имя com.androidbook.droidl.

7. Обратите внимание на флажок **Create Activity** (Создать деятельность). Он служит для создания класса запуска Activity по умолчанию. Назовите вашу деятельность DroidActivity.

ЧТО ТАКОЕ «ДЕЯТЕЛЬНОСТЬ»?

Деятельность — это центральный компонент платформы Android. Каждая деятельность представляет собой задачу, которую приложение должно выполнять, она часто связана с определенным экраном пользовательского интерфейса.

В приложении Droid #1 будет единственная деятельность — DroidActivity, выводящая на экран значения строкового ресурса. Более подробно о деятельности мы поговорим в часе 3.

Настройки нового проекта должны быть такими же, как показано на рис. 1.2.

8. Убедитесь, что значение поля ввода **Min SDK Version** (Минимальная версия SDK) корректно. В этом поле следует указывать минимально допустимую версию API целевой платформы по умолчанию (платформе Android версии 2.1 соответствует API Level 7). Измените значение этого поля, если хотите иметь поддержку более старых версий Android SDK.

Сейчас вы можете оставить это поле ввода без изменений.

9. Нажмите на кнопку **Next** (Далее).

10. Помимо приложений, мастер проектов Android позволяет создавать тестовые проекты. В нашем случае тестовый проект создавать не обязательно. Однако вы всегда можете добавить тестовый проект позже, нажав на соответствующую кнопку (с латинскими буквами «a», «J» и «u»: ), которая находится справа от кнопки вызова **Android Project Wizard** (Мастер проектов Android) на панели инструментов Eclipse. Подробную информацию о тестовых проектах вы можете найти в часе 22.

11. Нажмите на кнопку **Finish** (Завершить).

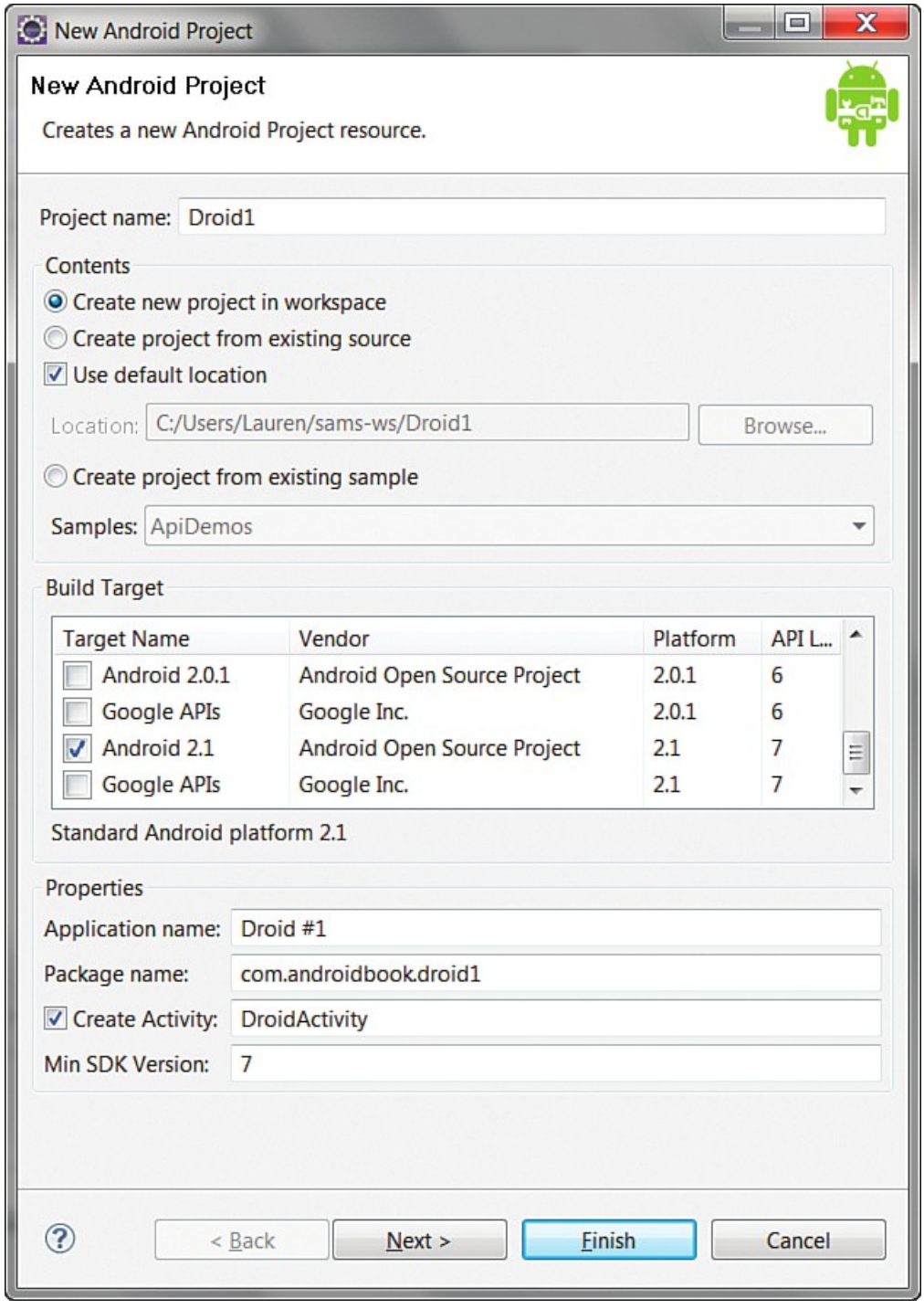

Рис. 1.2 Android Project Wizard (Мастер проектов Android) Eclipse

Структура проекта Android

Теперь на панели **File Explorer** (Проводник файлов) среды разработки Eclipse у вас есть новый проект Android — Droid 1. В дополнение к обычному JAR-файлу Android SDK, создаются другие важные файлы и папки:

- **AndroidManifest.xml** — основной конфигурационный файл приложения (файл манифеста);
- **default.properties** — сгенерированный файл сборки, используемый Eclipse и плагином ADT. Не редактируйте его;
- **/src** — папка, в которой хранится весь исходный код;
- **/src/com.androidbook.droidl/DroidActivity.java** — главная точка входа приложения — деятельность DroidActivity. Эта деятельность была определена как деятельность запуска по умолчанию в файле манифеста Android;
- **/gen/com.androidbook.droid1/R.java** — сгенерированный исходный файл управления ресурсами. Не редактируйте его; ;
- **/assets** — папка для хранения ^скомпилированных файлов ресурсов проекта;
- **/res** — папка, необходимая для управления всеми ресурсами приложения. Ресурсы приложения включают анимацию, графические объекты, файлы макетов, данные строкового и числового типа, а также подключаемые файлы;
- **/res/drawable** — графические ресурсы приложения (значки) разных размеров;
- **/res/layout/main.xml** — файл макета, используемый деятельностью DroidActivity для прорисовки экрана;
- **/res/values/strings.xml** — строковые ресурсы.

З НАЕТЕ ЛИ ВЫ, ЧТО...

С помощью мастера проектов Android вы можете добавить в Eclipse уже существующие проекты Android. Для этого установите переключатель в положение **Create project from existing source** (Создать проект на основе существующего источника] вместо **Create new project in workspace** (Создать в рабочей области новый проект) в диалоговом окне **New Android Project** (Новый проект Android), как на рис. 1.2. Несколько демонстрационных проектов, разработанных под разные версии платформы, вы можете найти в папке /samples Android SDK. В этой папке выберите любую из папок **/android-***, где символ * — это версия платформы.

Вы можете также установить переключатель в положение **Create project from existing sample** (Создать проект на основе существующего примера), название которого говорит само за себя. Но перед тем, как перейти к списку демонстрацион- ных проектов, убедитесь, что выбрана нужная версия платформы.

Редактирование ресурсов проекта

Каждый раз при создании нового проекта автоматически открывается редактор ресурсов файла манифеста Android. Если вы закрыли этот редактор, когда просматривали другие файлы проекта, щелкните дважды по файлу **AndroidManifest.xml,** чтобы вернуться в редактор (см. рис. 1.3).

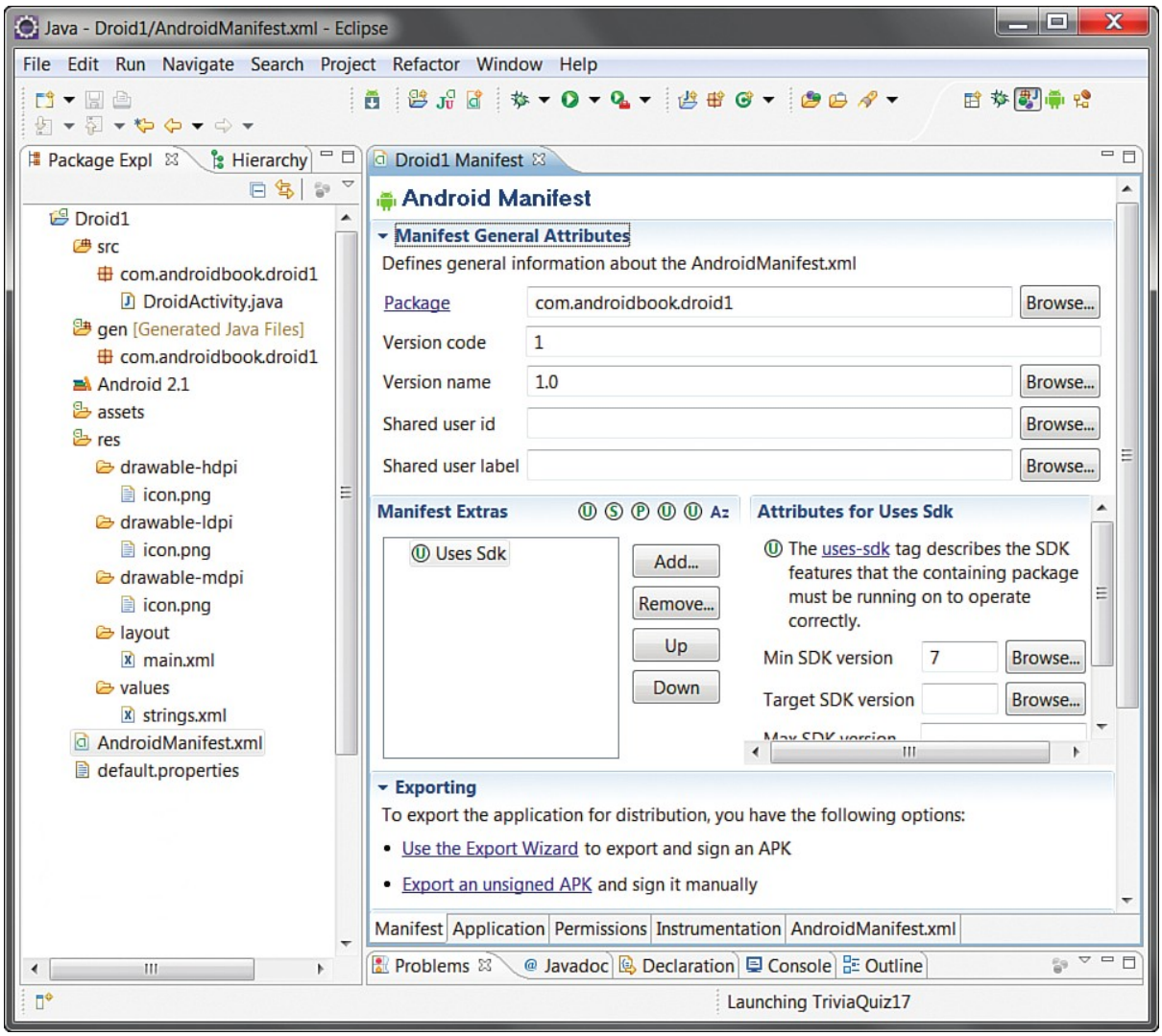

Рис. 1.3. Редактирование ресурсов Android в Eclipse

РЕДАКТИРОВАНИЕ ФАЙЛА МАНИФЕСТА ANDROID

Файл манифеста Android — это основной конфигурационный файл каждого приложения Android. Редактор распределяет информацию из этого файла по нескольким вкладкам.

- **Manifest** (Манифест) — на этой вкладке, показанной на рис. 1.3, определяются общие параметры приложения, такие как название пакета п информация о версии приложении (для установки и обновления).

- **Application** (Приложение) — на этой вкладке определяются такие данные о приложении, как имя и значок приложения, а также внутренние компоненты приложения, например, какие деятельности могут выполняться (включая деятельность запуска DroidActivity по умолчанию) и другие возможности и сервисы, обеспечиваемые приложением.

- **Permissions** (Разрешения) — это вкладка, где определяются права приложения. Например, если приложению необходима возможность чтения списка контактов

телефона, то в конфигурационный файл должен быть добавлен тип Uses-Permission с правом на чтение контактов (android, permission. READCONTACTS).

- **Instrumentation** (Инструментарий) — эта вкладка используется для тестирования компонентов с помощью различных классов инструментария, доступных в Android SDK.

- **AndroidManifest.xml** — эта вкладка представляет собой простой XML- редактор для редактирования файла манифеста вручную.

Если перейти на вкладку **AndroidManifest.xml,** вы увидите код файла манифеста, который выглядит примерно следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.androidbook.droidl"
    android: versionCode=" 1"
    android:versionName="1. 0">
    <appplication
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity
           android: name=". DroidActivity"
           android:label="@string/app_name">
            <intent-filter>
             <action
                android:name="android.intent.action.MAIN" />
             <category
                android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
        </activity>
    </application>
    <uses-sdk
android:minSdkVersion="7" />
</manifest>
```

ВНИМАНИЕ!
ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Все файлы ресурсов Android, включая файл манифеста Android, записываются в формате XML. Это означает, что вы можете редактировать их без специального редактора. Чтобы создать новый XML-файл Android, нажмите на кнопку его создания

с изображением листа бумаги, (буквой «а» и знаком «плюс» ) на панели инструментов Eclipse.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО
РЕДАКТИРОВАНИЕ КОНФИГУРАЦИОННОГО ФАЙЛА ANDROID

Попробуйте отредактировать файл манифеста Android. Вы не сможете проводить отладку приложения, пока не установите значение атрибута android:debuggable равным true. Для этого выполните следующие действия:

1. Откройте файл **AndroidManifest.xml** в редакторе ресурсов.
2. Перейдите на вкладку **Application** (Приложение).
3. Раскройте список атрибута debuggable и выберите пункт true.
4. Сохраните файл манифеста.

Теперь, если переключиться на вкладку **AndroidManifest.xml,** вы увидите, что у тега «application» появился атрибут отладки:

```
android:debuggable="true"
```

РЕДАКТИРОВАНИЕ ДРУГИХ ФАЙЛОВ РЕСУРСОВ

большинство ресурсов приложения Android хранятся в папке **/res.** Она содержит в себе подпапки:

- **/drawable-ldpi, /drawable-hdpi, /drawable-mdpi** — в этих подпапках хранятся графические файлы ресурсов для различной плотности размеще ния точек и разрешений экрана. Если вы просмотрите содержимое этих папок на панели **Project Explorer** (Проводник проектов), то найдете гра фический файл **icon.png** в каждой из них. Это значок приложения. По дробнее о различии между этими папками вы узнаете в часе 20.

- **/layout** — и этой подпапке хранятся файлы макета пользовательскою интерфейса. Здесь вы можете найти файл макета экрана **main.xml,** кото рый содержит описание пользовательского интерфейса для деятельно сти по умолчанию.

- **/values –** в этой подпапке различные типы ресурсов сгруппированы по типам, таким как строковые значения, значения цвета и другие примитивные типы. Здесь вы можете видеть файл ресурса **strings.xml,** который содержит все строковые ресурсы, используемые в приложении.

Щелкните дважды по какому-либо из файлов ресурсов, чтобы открыть его н редакторе.

Помните, что вы можете редактировать XML-данные непосредственно н текстовом редакторе.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО
РЕДАКТИРОВАНИЕ СТРОКОВЫХ РЕСУРСОВ

Если вы посмотрите на код файла макета **main.xml,** то увидите, что он выводит на экран простой интерфейс с единственным элементом пользовательского интерфейса — TextView. Этот элемент выводит на экран текстовую строку. В нашем случае отображаемый на экране текст определяется строковым ресурсом 0string/hello.

Для редактирования строкового ресурса 0string/hello редактором строковых ресурсов выполните следующие действия:

1. Откройте в редакторе ресурсов файл **strings.xml.**
2. Обратите внимание на строковый ресурс с именем hello и текстом Hello World, DroidActivity! в редакторе ресурсов.
3. В поле ввода **Value** (Значение) измените это значение на Hello, Dave.

4. Сохраните файл.

Теперь если вы переключитесь на вкладку **strings.xml** и посмотрите на код XML, то увидите, что в контейнерном теге <resources> содержится два строковых элемента:

```
<?xml version="l.0" encoding="utf-8"?>
<resources>
<string name="hello">Hello, Dave</string>
<string name="app_name">Droid #l</string>
</resources>
```

Первая строка соответствует ресурсу @string/hello. Вторая — ресурсу 0string/app_name, который содержит название приложения. Если вернуться к конфигурационному файлу Android, вы можете заметить, что ресурс ©string/ app_name используется в конфигурации приложения.

Намного более подробно о ресурсах проектов мы поговорим в часе 4. А теперь давайте перейдем к рассмотрению вопроса компиляции и выполнения приложений.

ВЫПОЛНЕНИЕ И ОТЛАДКА ПРИЛОЖЕНИЙ

Перед сборкой и отладкой приложений Android сначала вы должны настроить проект для отладки. Плагин ADT позволяет вам делать это непосредственно в среде разработки Eclipse. Вы должны сделать следующее:

- сконфигурировать AVD (Android Virtual Device, виртуальное устройство Android) для эмулятора;
- создать отладочную конфигурацию для своего проекта;
- собрать проект Android и запустить отладочную конфигурацию.

После выполнения этих действий Eclipse присоединит свой отладчик к эмулятору Android (или мобильному телефону), и вы сможете выполнять отладку приложения.

Управление виртуальными устройствами Android

Чтобы запустить приложение в эмуляторе Android, сначала вы должны сконфигурировать AVD. Профиль AVD описывает тип устройства, которое вы хотите имитировать в эмуляторе, включая версию поддерживаемой платформы. Вы можете указать различные размеры экранов и ориентации, а также имеет ли эмулятор SD-карту и, если она есть, ее емкость. В нашем случае AVD по умолчанию с поддержкой Android версии 2.1 будет достаточно. Ниже приведена пошаговая инструкция для создания типичного AVD.

1. Запустите Android SDK and AVD Manager в Eclipse, щелкнув по кнопке с логотипом Android и стрелкой () на панели инструментов. Вы можете также запустить этот менеджер командой **Window => Android SDK and AVD Manager** (Окно => Менеджер Android SDK и AVD).

2. Выберите пункт **Virtual Devices** (Виртуальные устройства) в левом меню. Сконфигурированные AVD будут выведены на экран в виде списка.

3. Нажмите на кнопку **New** (Новый), чтобы создать новый AVD.

4. Введите имя для AVD в поле ввода **Name** (Имя). Назовите этот AVD именем VanillaAVD.

5. Выберите версию платформы в раскрывающемся списке **Target** (Целевая версия). Например, чтобы иметь поддержку Android 2.1, выберите пункт **Android 2.1 - API Level 7.**

6. Выберите емкость SD-карты в килобайтах или мегабайтах. Образ этой SD-карты будет занимать пространство на жестком диске вашего компьютера, поэтому не задавайте слишком большой объем, желательно не больше 1024 Мбайт. (Минимальный объем составляет 9 Мбайт, но имей те в виду, что весь объем SD-карты хранится на вашем компьютере.)

7. Выберите соотношение сторон для обложки. Опция **Skin** (Обложка) влияет на визуальное отображение эмулятора. В нашем случае подойдет экран HVGA, который задан здесь по умолчанию. Он соответствует портретному режиму экрана. Все параметры создаваемого AVD должны быть такими же, как показано на рис. 1.4.

8. Нажмите на кнопку **Create AVD** (Создать AVD) и ждите завершения операции.

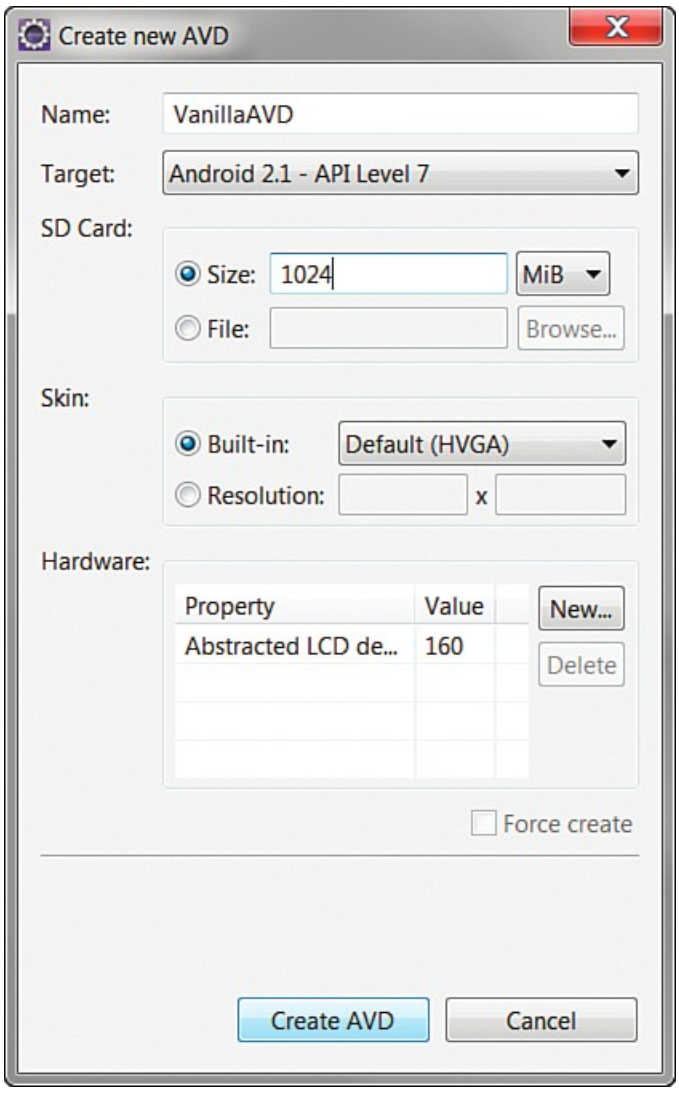

Рис. 1.4. Создание нового AVD в Eclipse

9. Закройте Android SDK и AVD Manager.

КСТАТИ

Поскольку Android SDK и AVD Manager форматирует память, выделенную для образов SD-карт, создание AVD с большим объемом SD-карты может занять некоторое время.

Создание отладочной конфигурации и конфигурации выполнения

Уже сейчас почти все готово для запуска вашего первого приложения. Для этого вам осталось выполнить последнее задание: вы должны создать отладочную конфигурацию (или конфигурацию выполнения) для вашего проекта.

Эта процедура состоит из следующих этапов:

1. Выполните команду **Run => Debug Configurations** (Выполнение => Конфигурации отладки).

2. Щелкните дважды по пункту **Android Application** (Приложение Android), чтобы создан, новую конфигурацию.

3. Созданная вами конфигурация называется **New_configuration.**

4. Измените это имя на DroidDebug в поле ввода **Name** (Имя).

5. Нажмите на кнопку **Browse** (Обзор) и выберите проект **Droid 1.**

6. На вкладке **Target** (Целевая версия) установите флажок напротив имени созданного вами AVD.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Если на вкладке **Target** (Целевая версия) вы выберите **Manual** (Вручную) вместо **Automatic** (Автоматически), то каждый раз при запуске этой конфигурации вам будет предлагаться выбор целевой версии платформы. Эта возможность может оказаться полезной, если вы собираетесь тестировать приложение на нескольких устройствах и конфигурациях эмулятора. Более подробную информацию об этом вы найдете в разделе «Запуск приложений Android на мобильном телефоне» этого часа.

7. Сохраните изменения, нажав на кнопку **Apply** (Применить). В результате диалоговое окно **Debug Configurations** (Конфигурации отладки) должно выглядеть так, как показано на рис. 1.5.

Запуск приложений Android в эмуляторе

Теперь все готово к первому запуску приложения. Чтобы запустить его, нажмите на кнопку **Debug** (Отладка) в окне **Debug Configurations** (Конфигурации отладки) или на кнопку с изображением зеленого жука () на панели инструментов Eclipse, а затем выберите конфигурацию отладки DroidDebug из списка.

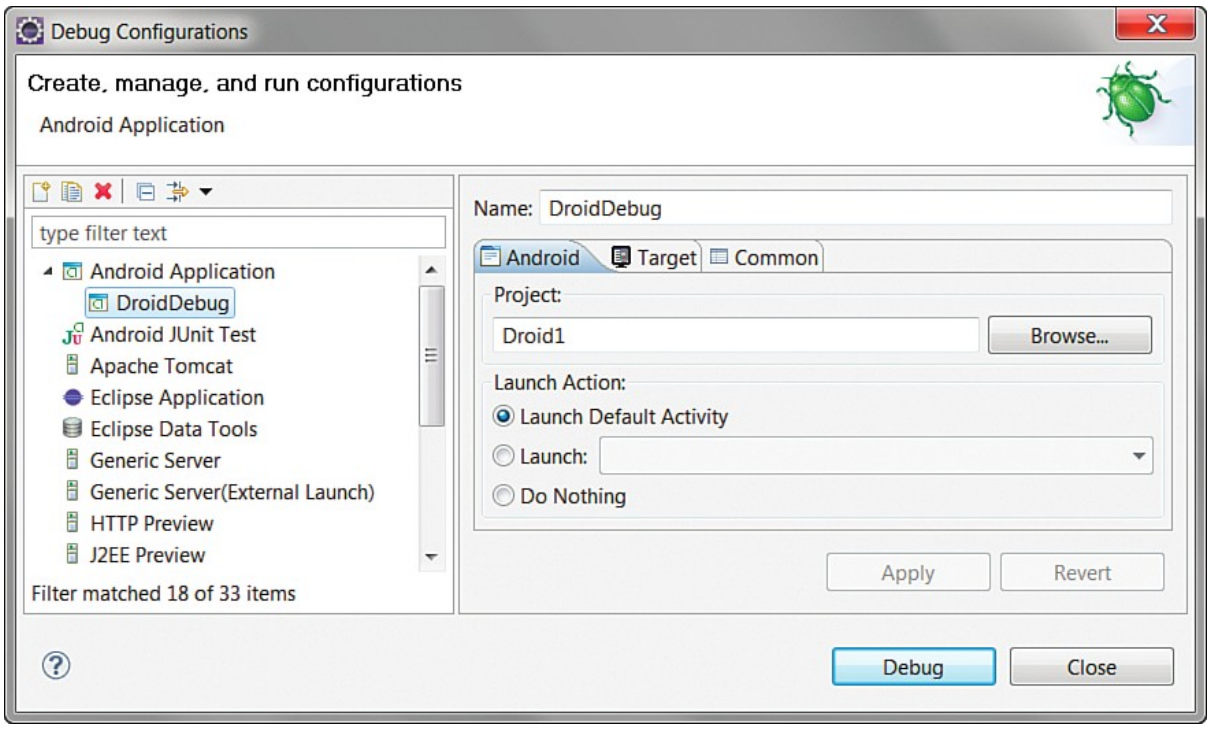

Рис. 1.5. Конфигурация отладки DroidDebug в Eclipse

КСТАТИ

При первой попытке выбрать конфигурацию отладки DroidDebug из меню отладки вам может быть предложено пройти через менеджер конфигурации. После этого конфигурация будет доступна непосредственно в меню.

После нажатия на кнопку отладки откроется окно эмулятора. Это может занять некоторое время, не торопитесь. Возможно, вам понадобится нажать кнопку **Menu** (Меню) в эмуляторе, чтобы перейти с экрана блокировки на необходимый нам экран (см. рис. 1.6).

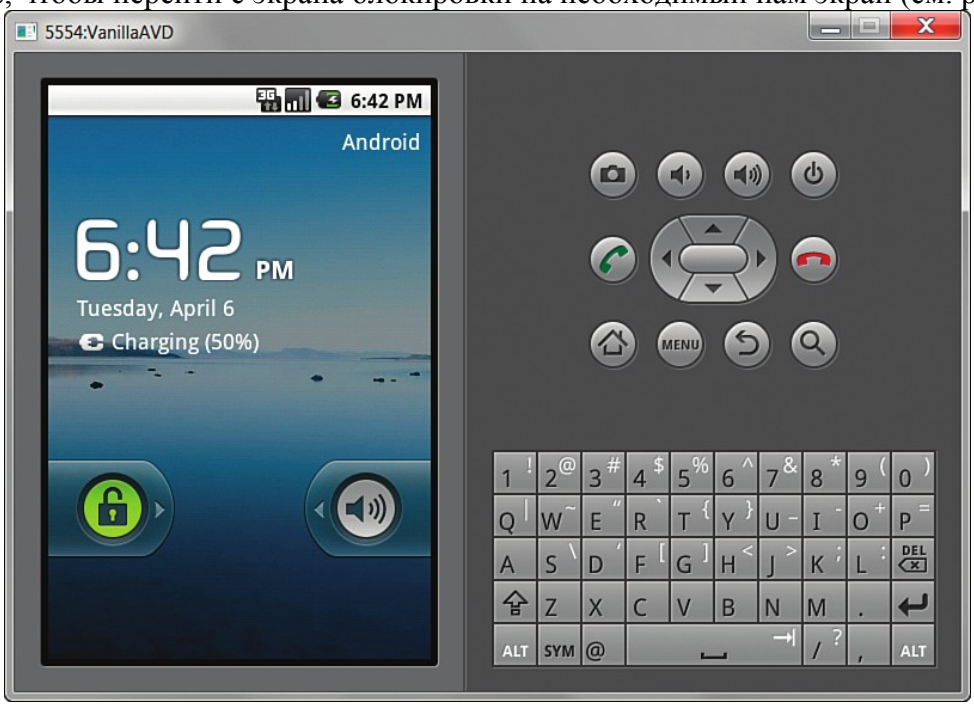

Рис. 1.6. Запуск эмулятора Android (вид экрана блокировки)

Теперь отладчик Eclipse присоединен, и ваше приложение выполняется, как показано на рис. 1.7.

Как вы можете видеть, это приложение очень простое. Оно выводит на экран текст единственного элемента пользовательского интерфейса I ext View. Никаких других действий приложение не выполняет.

Рис. 1.7. Приложение Android Droid #1, запущенное в эмуляторе

Отладка приложений Android с помощью DDMS

В дополнение к обычной перспективе отладки (так называется набор представлений), встроенной в Eclipse, плагин ADT располагает перспективой DDMS (Dalvik Debug Monitor Service). Во время выполнения приложения обратите внимание на эту перспективу в Eclipse. Открыть перспективу

DDMS (см. рис. 1.8) вы можете, щелкнув по значку **DDMS** ( ) в верхнем правом углу интерфейса Eclipse. Чтобы перейти обратно к панели **Project Explorer** (Проводник проектов), выберите перспективу **Java** в верхнем правом углу Eclipse.

КСТАТИ
Если кнопка перспективы DDMS не отображается в Eclipse, вы можете добавить ее на рабочее пространство, щелкнув по кнопке **Open Perspective** (Открыть перспективу) в верхнем правом углу рядом с кнопками других перспектив (или выполнив команду **Window => Open Perspective** (Окно =» Открыть перспективу)). Затем в открывшемся меню **Open Perspective** (Открыть перспективу) выберите пункт **Other** (Другие), чтобы получить полный список доступных перспектив. Выберите перспективу **DDMS** и нажмите на кнопку **ОК**

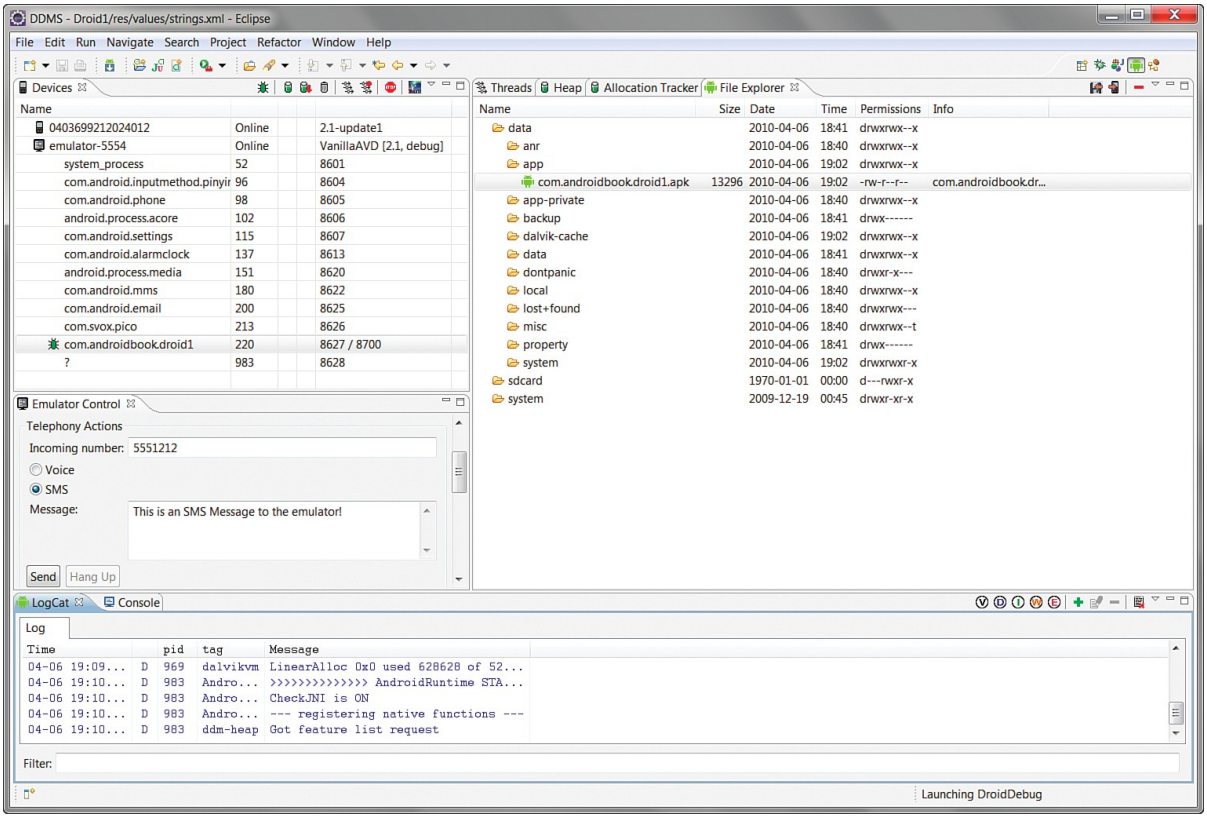

Рис. 1.8. Перспектива DDMS в Eclipse

Перспектива DDMS может использоваться для мониторинга процессов приложения и непосредственного взаимодействия с эмулятором. Вы можете имитировать голосовые вызовы и отправлять SMS на эмулятор. Вы можете также задавать ложное местоположение эмулятору для тестирования геолокационных служб. Подробнее о DDMS и других инструментах, доступных разработчикам Android, вы узнаете в часе 2.

Средство ведения журнала LogCat по умолчанию присутствует как на перспективе DDMS, так и на перспективе отладки. Оно служит для отображения накопленной информации с эмулятора или подключенного мобильного телефона.

Запуск приложений Android на мобильном телефоне

Пришло время для загрузки вашего приложения на настоящий мобильный телефон. Для этого вы должны подключить телефон к компьютеру через USB-кабель передачи данных.

Но сначала убедитесь, что все параметры отладки заданы верно:

1. Выполните команду Run => Debug Configurations (Выполнение =*t*> Конфигурации отладки).

2. Выберите конфигурацию отладки **DroidDebug.**

3. На вкладке **Target** (Целевая версия) в разделе **Deployment Target Selection Mode** (Режим выбора версии платформы) установите переключател!. в положение **Manual** (Вручную). Вы можете в любое время вернуть автоматический режим, но сейчас вам нужен режим выбора вручную, чтобы каждый раз при запуске отладки иметь возможность выбора ее запуска в эмуляторе (и соответственно выбрать AVD) или на одном из подключенных мобильных устройств.

4. Сохраните изменения, щелкнув по кнопке **Apply** (Применить).

5. Подключите устройство Android к компьютеру с помощью USB-кабеля.

6. Затем в Eclipse нажмите на кнопку **Debug** (Отладка).
Появится диалоговое окно (рис. 1.9), отображающее все доступные кон фигурации для выполнения и отладки вашего приложения. Здесь вы видите список всех физических и виртуальных устройств. Вы можете также запускать новые экземпляры эмуляторов из созданных вами AVI).

7. Щелкните дважды по одному из подключенных устройств Android в списке. Здесь каждому из подключенных мобильных телефонов (вклю чая эмуляторы) соответствует один пункт. Если в этом списке вы не видите подключенный вами мобильный телефон, проверьте USB- кабель и убедитесь, что все необходимые драйверы установлены, как описано в приложении А.

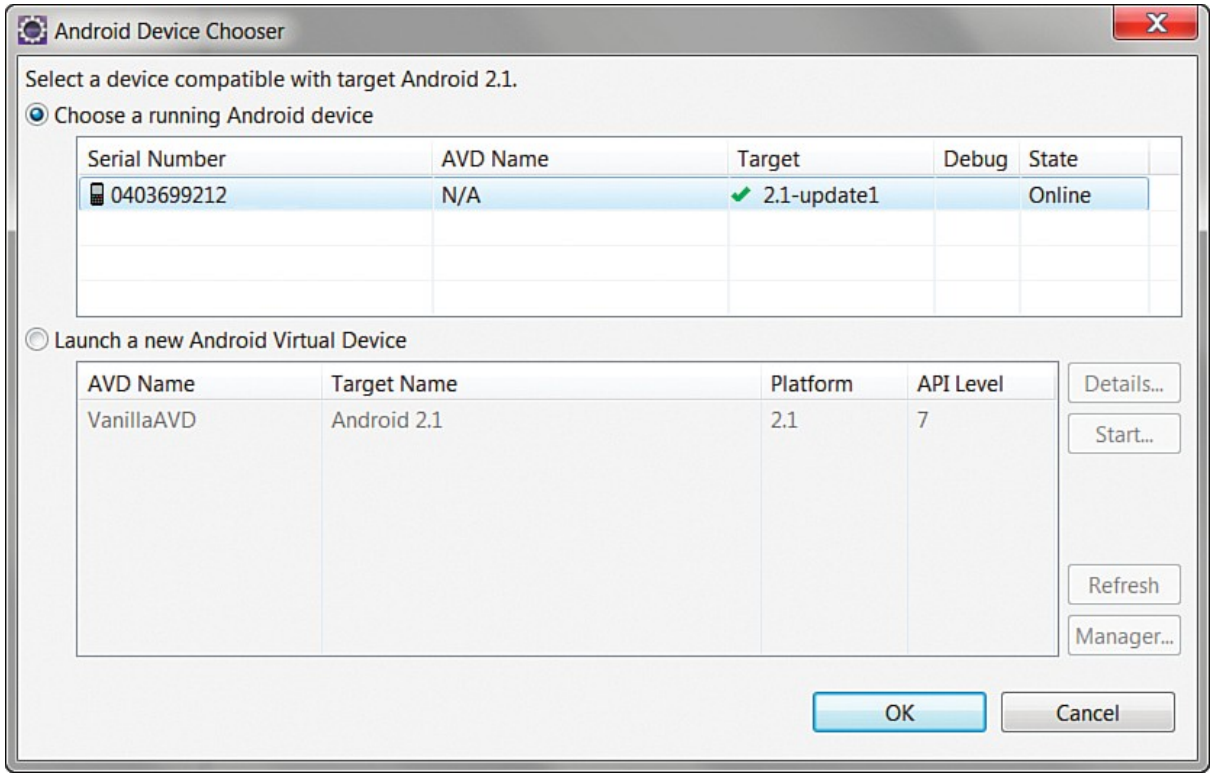

Рис. 1.9. Диалоговое окно Eclipse выбора устройства для развертывания приложения

После этого Eclipse выполнит установку приложения Android на мобильный И'игфоп, присоединит отладчик и запустит приложение. На экране вашего Могншыюго телефона должно появиться изображение, похожее на то, которое мы видели в эмуляторе. В перспективе DDMS вы можете видеть всю до- fiymiyio журналируемую информацию, а также множество возможностей Перспективы DDMS для работы с физическими телефонами и эмуляторами.

ПЛОХО ВЛАДЕЕТЕ ECLIPSE?

Если вы все еще чувствуете себя неуверенно в среде разработки Eclipse, можете обратиться к приложению Б.

ИТОГИ

По правляем! Теперь вы можете считать себя Android-разработчиком. Но ШUK;I ивы еще только начинаете свой путь разработки приложений Android J среде Eclipse. Вы создали свой первый проект. Вы рассмотрели и скомпилировали работающий код Android. Наконец, вы запустили созданное приложение в эмуляторе и на настоящем устройстве Android.

ВОПРОСЫ И ОТВЕТЫ

Вопрос. Какие языки программирования поддерживаются для разработки приложений Android?

Ошет. На сегодняшний день Java — единственный язык программирова**нии,** полностью поддерживаемый Android. Другие языки, такие как C++, Могут быть добавлены в будущем. Несмотря на то, что приложения должны быть на языке Java, языки C и C++ все же могут использоваться для реше**ния** задач, в которых необходима более высокая производительность, в этом случае применяется Android NDK. Более подробную информацию об использовании Android NDK вы можете найти по адресу **developer.android. com/sdk/ndk.**

Вопрос. Зачем создавать AVD для платформы Android версии 1.1 (или любой более старой версии прошивки), если существуют новые версии Android **SDK?**

Ответ. Несмотря на то, что прошивку большинства телефонов можно легко обновить, не каждое устройство на базе платформы Android будет поддерживай» новые версии. Поэтому перед выбором версии платформы, поддерживаемой приложением, узнавайте о прошивке, поддерживаемой телефонами целевой аудитории.

Вопрос. Редакторы ресурсов Android могут быть неудобны для обработки большого объема данных, таких как строковые ресурсы. Есть ли какой-либо способ решения этой проблемы?

Ответ. Файлы проектов Android, такие как файл манифеста, файлы макетов и ресурсов (например, **/res/values/strings.xml)** хранятся в виде специальных XML-файлов. Вы можете редактировать эти файлы вручную, щелкнув по вкладке редактирования кода XML редактора ресурсов. Более подробную информацию о формате XML вы узнаете в часе 4.

ПРАКТИКУМ
Контрольные вопросы

1. Кто входит в состав Альянса разработчиков открытых стандартов мобильных устройств (Open Handset Alliance)?

   A. Производители мобильных устройств.

   B. Операторы беспроводной связи и поставщики услуг.

   C. Разработчики программного обеспечения для мобильных устройств. Г. Все вышеупомянутые.

2. Сразу после установки SDK можно просто запустить эмулятор Android с настройками по умолчанию. Правда ли это?

3. Какая из ниже перечисленных IDE для разработки приложений Android самая популярная?

   A. Eclipse.
   B. IntelliJ.
   C. Emacs.

4. Вы можете использовать Eclipse для отладки приложений на мобильном устройстве. Правда ли это?

Ответы

1. Правильный ответ: Г. Альянс разработки открытых стандартов мобиль ных устройств — деловое сообщество, представляющее все сегменты рынка мобильных устройств.

2. Нет, это не так. Сначала вы должны создать AVD.

3. Правильный ответ: A. Eclipse — самая популярная IDE для разработки приложений Android. Другие IDE также могут использоваться, но они не поддерживают плагин ADT в отличие от Eclipse.

4. Да, это правда. Eclipse поддерживает отладку приложений как в эмуляторе, так и на мобильных устройствах.

Упражнения

1. Зайдите на сайт **developer.android.com** и ознакомьтесь с его содержанием. Обратите внимание на вкладку **Dev Guide** (Руководство разработчика) и ссылочные

материалы. Перейдите в раздел **Community** (Сообщество) на вкладке **Resources** (Ресурсы). Здесь вы можете зарегистрироваться в качестве начинающего разработчика Android.

2. Добавьте больше текста для отображения к созданному вами приложению **Droid #1.** Для этого добавьте еще один строковый ресурс в файл **strings.xml** и сохраните его. Затем с помощью редактора ресурсов измените файл **main.xml,** чтобы добавить второй элемент управления TextView. Присвойте атрибуту android:text элемента пользовательского интерфейса TextView значение, соответствующее только что созданному строковому ресурсу. Наконец, перезапустите приложение в эмуляторе, чтобы оценить изменения.

3. Откройте на рабочем пространстве Eclipse один из демонстрационных проектов Android, идущих в комплекте с Android SDK. Просмотрите файлы проекта, создайте конфигурацию выполнения и запустите демонстрационное приложение в эмуляторе.

Час 2. ОСВОЕНИЕ ИНСТРУМЕНТОВ РАЗРАБОТКИ ANDROID

Вопросы, рассматриваемые в этом часе:

- **использование документации Android;**
- **выполнение отладки приложений с помощью DDMS;**
- **работа с эмулятором Android;**
- **использование Интерфейса отладки Android (ADB, Android Debug Bridge);**
- **работа с виртуальными устройствами Android.**

На сегодняшний день в распоряжении разработчиков Android находится около десятка средств разработки, облегчающих процесс создания качественных приложений. Приоритетная задача на ранней стадии освоения Android — получить представление о доступных инструментах и о том, как они могут использоваться. Когда вы сталкиваетесь с какой-либо проблемой, у вас уже будет некоторое представление о том, какие инструменты могут вам помочь в ее решении. Средства разработки Android находятся в подпапке **/tools** папки установки Android SDK. В этом часе вы узнаете о большинстве самых важных инструментов, доступных для свободного использования. Эта информация поможет вам разрабатывать приложения Android быстрее и легче.

ДОКУМЕНТАЦИЯ ПО ПЛАТФОРМЕ ANDROID

Хотя сама по себе документация Android — не инструмент, это ресурс первостепенной важности для каждого разработчика Android. Ее HTML-вари ант можно найти в подпапке **/docs** Android SDK, обращайтесь к нему при возникновении каких-либо затруднений. Последнюю редакцию справочной документации можно найти на веб-сайте разработчиков Android: **http:// developer.android.com.**

Документация Android состоит из шести разделов (см. рис. 2.1):

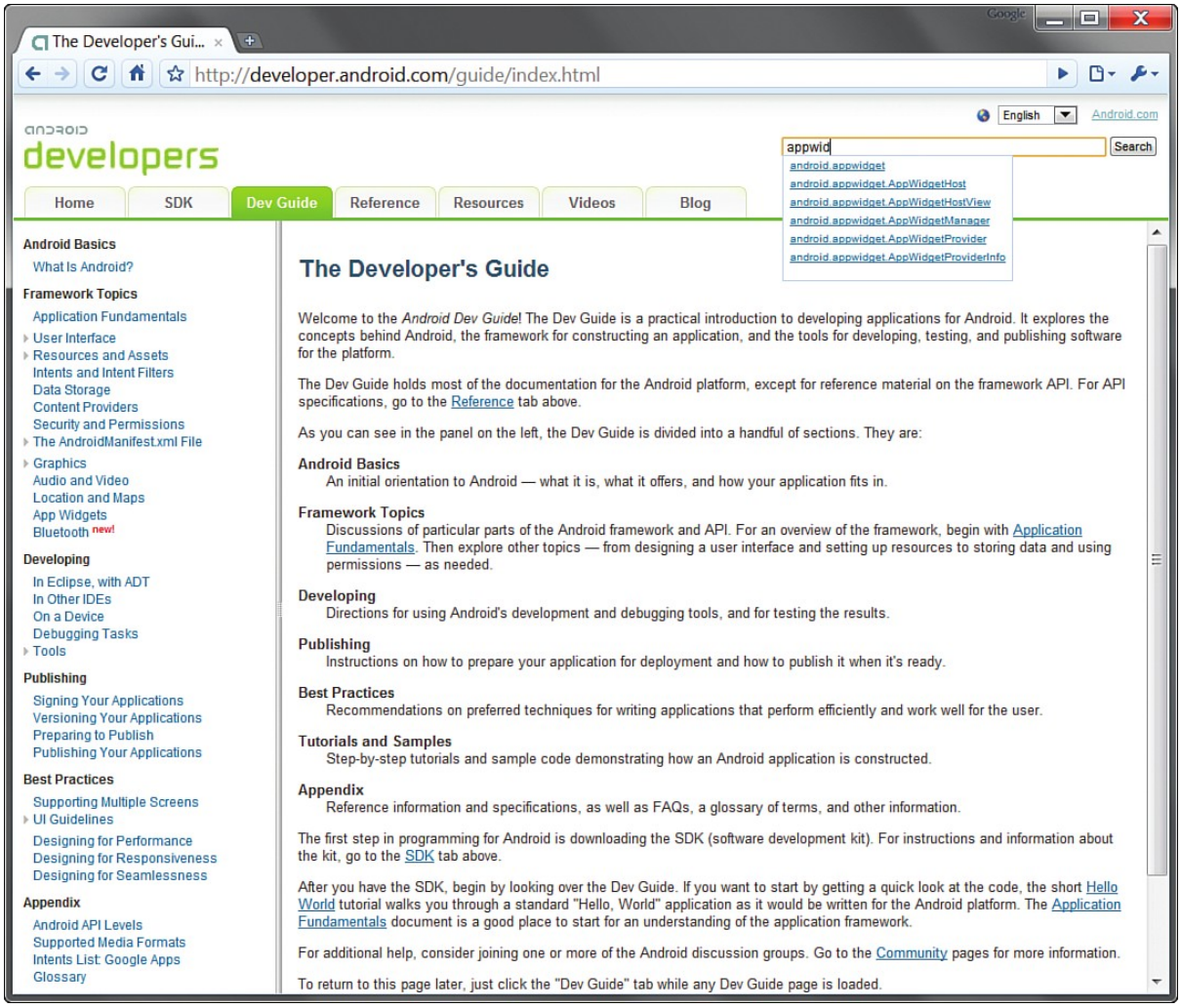

Рис. 2.1. Документация разработчиков Android (онлайн-версия)

- **SDK** важная информация о **SDK,** установленном на вашем компьютере. Здесь имеются сведения о версиях и описание всех известных проблем каждой из них. Эта информация может вам пригодиться, если онлайн-справка была обновлена для новой версии, но вы хотите разрабатывать приложение на более старой версии SDK.

- **Dev Guide** (Руководство разработчика) — эта вкладка содержит руководство разработчика Android, включая ответы на часто задаваемые вопросы, а также пошаговые примеры и глоссарий терминов Android для начинающих.

- **Reference** (Справка) — на этой вкладке представлен поиск по списку пакетов и классов всех программных интерфейсов Android.

- **Blog** (Блог) — официальный блог разработчиков Android. Здесь вы можете найти последние новости и объявления. Это прекрасное место для поиска примеров с практическими рекомендациями, обучения оптимизации приложений Android и получения информации о новых версиях SDK и конкурсах для разработчиков.

- **Videos** (Видео) — обучающий видеоматериал, который можно посмотреть онлайн. Здесь вы найдете видеоролики о платформе Android с советами по разработке, а также записи конференций Google на английском языке.

- **Resources** (Ресурсы). Раздел **Community** (Сообщество) этой вкладки — I ваша отправная точка для обсуждения вопросов с другими разработчиками Android. Кроме того, вы можете вступить в одну или несколько тематических групп Google по вашему желанию.

Рекомендуем вам провести первое ознакомление с документацией Android SDK сейчас. Но перед тем, как перейти к онлайн-документации, ознакомьтесь с ее локальным вариантом.

ОТЛАДКА ПРИЛОЖЕНИЙ С ПОМОЩЬЮ DDMS

Мониторинговый сервис отладки Dalvik (DDMS, Dalvik Debug Monitor Service) — утилита отладки, интегрируемая в Eclipse через перспективу DDMS. Перспектива DDMS предоставляет несколько полезных возможностей взаимодействия с мобильными телефонами и эмуляторами (см. рис. 2.2).

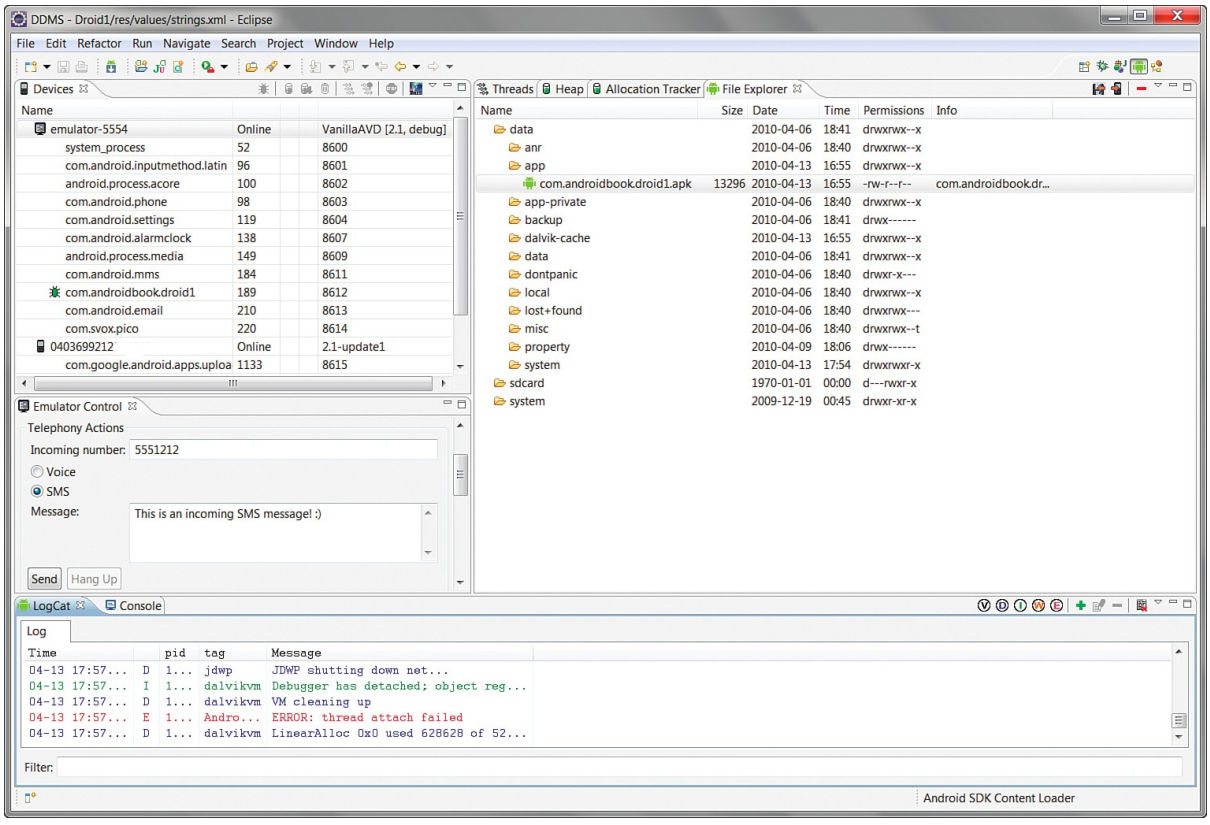

Рис. 2.2. Перспектива DDMS с одним эмулятором и одним устройством Android

Возможности DDMS можно грубо разделить на пять категорий:

- управление задачами;
- управление файлами;
- взаимодействие с эмулятором;
- журналирование;
- создание скриншотов.

DDMS и перспектива DDMS — основные инструменты отладки. Рассмотрим их возможности более детально.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Инструмент DDMS может быть запущен отдельно от Eclipse. Его исполняемый файл находится в папке Android SDK **/tools.**

Управление задачами

На панели **Devices** (Устройства) перспективы DDMS приводится список (Мониторов и мобильных телефонов, подключенных в настоящий момент. Вы можете выбрать отдельные экземпляры, просмотреть текущие процессы и потки. Чтобы просмотреть поток, щелкните мышью по интересующему процессу устройства, например **com.androidbook.droid1,** затем на кнопке обновления потока (), как показано на рис. 2.3. Вы можете также запросить сбор лишних данных на процессе и затем посмотреть обновления стека, щелкнув по кнопке с изображением зеленого цилиндра (). Наконец, можно остановить процесс, щелкнув по кнопке с изображением знака «stop» ().

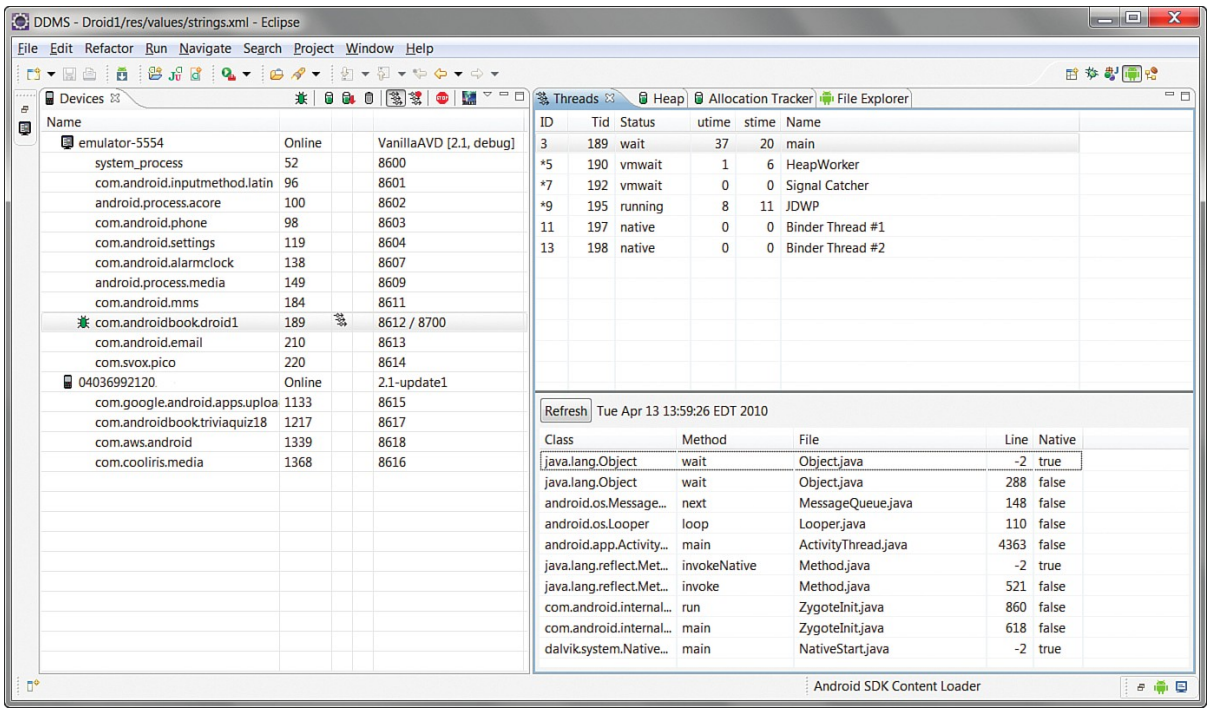

Рис. 2.3. Использование DDMS для просмотра активности потоков приложения Droid1

ОТЛАДКА В ПЕРСПЕКТИВЕ DDMS

В пределах перспективы DDMS вы можете выбрать определенный процесс на эмуляторе или мобильном телефоне и затем нажатием на кнопку отладки с изображением зеленого жука присоединить отладчик к этому процессу. Для использования этой возможности на вашем рабочем пространстве Eclipse должен быть исходный код. Это относится только к Eclipse, в автономной версии DDMS этого делать нельзя.

Обзор файловой системы Android

Для просмотра файлов и папок в эмуляторе или на устройстве вы можете воспользоваться панелью **File Explorer** (Проводник файлов) в перспективе DDMS (см. рис. 2.4). Вы можете копировать файлы между файловой системой Android и вашим компьютером с помощью кнопок перемещения ( ) и ( ).

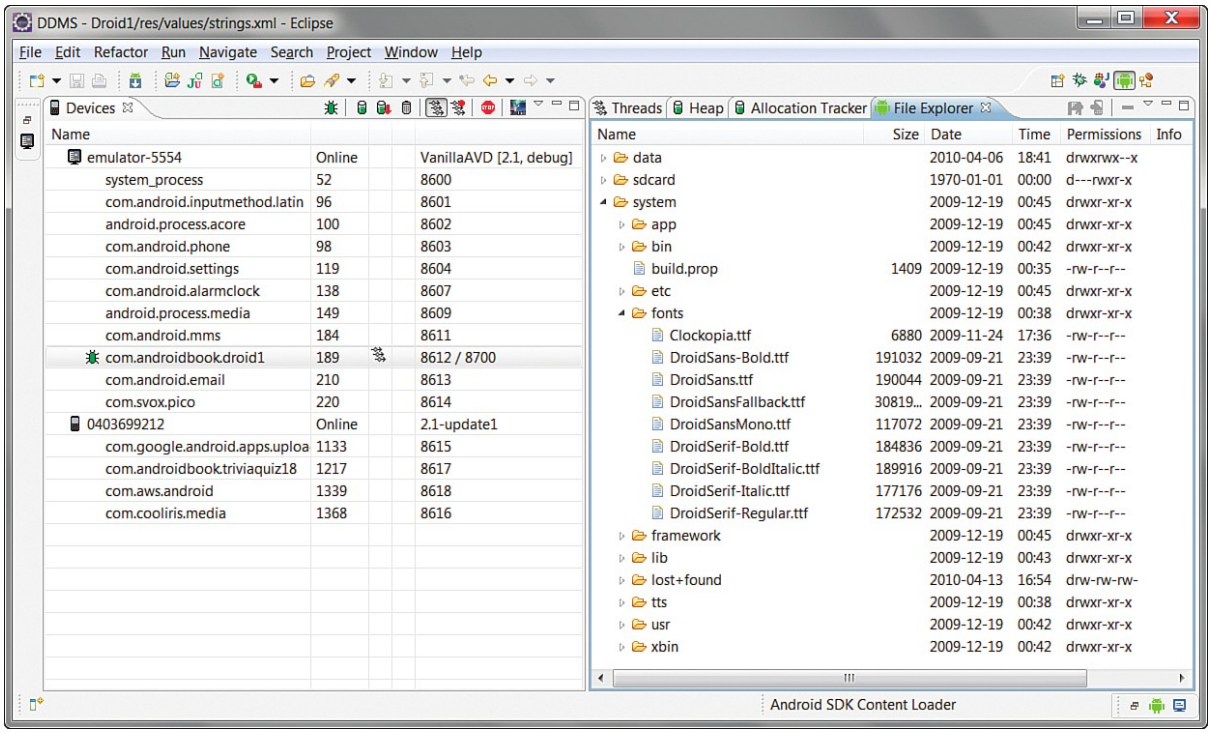

Рис. 2.4. Просмотр доступных шрифтов для мобильного телефона на панели **File Explorer** (Проводник файлов) и перспективе DDMS

Удалять файлы и папки вы можете кнопкой с изображением знака «минус» ( ) или клавишей **Delete.** При этом не будет выведено сообщение с предложением о подтверждении удаления и отмены этого действия.

Взаимодействие с эмуляторами

DDMS может инициировать множество событий, таких как голосовые вызовы, отправка и прием SMS и запрос геолокационных координат на определенных экземплярах эмулятора. Эта возможность доступна на панели **Emulator Control** (Управление эмулятором) в DDMS. Все события могут быть инициированы только в одну сторону — из DDMS к устройству.

КСТАТИ

Эти возможности доступны только для эмуляторов. В случае с мобильными телефонами вы должны использовать реальные вызовы и реальные сообщения.

ИМИТАЦИЯ ВХОДЯЩИХ ГОЛОСОВЫХ ВЫЗОВОВ НА ЭМУЛЯТОРЕ

Входящие голосовые вызовы инициируются на панели **Emulator Control** (Управление эмулятором) в DDMS (см. рис. 2.5). Так как это не настоящий звонок, никакие данные (звукового или другого типа) между вызывающей стороной и адресатом не передаются.

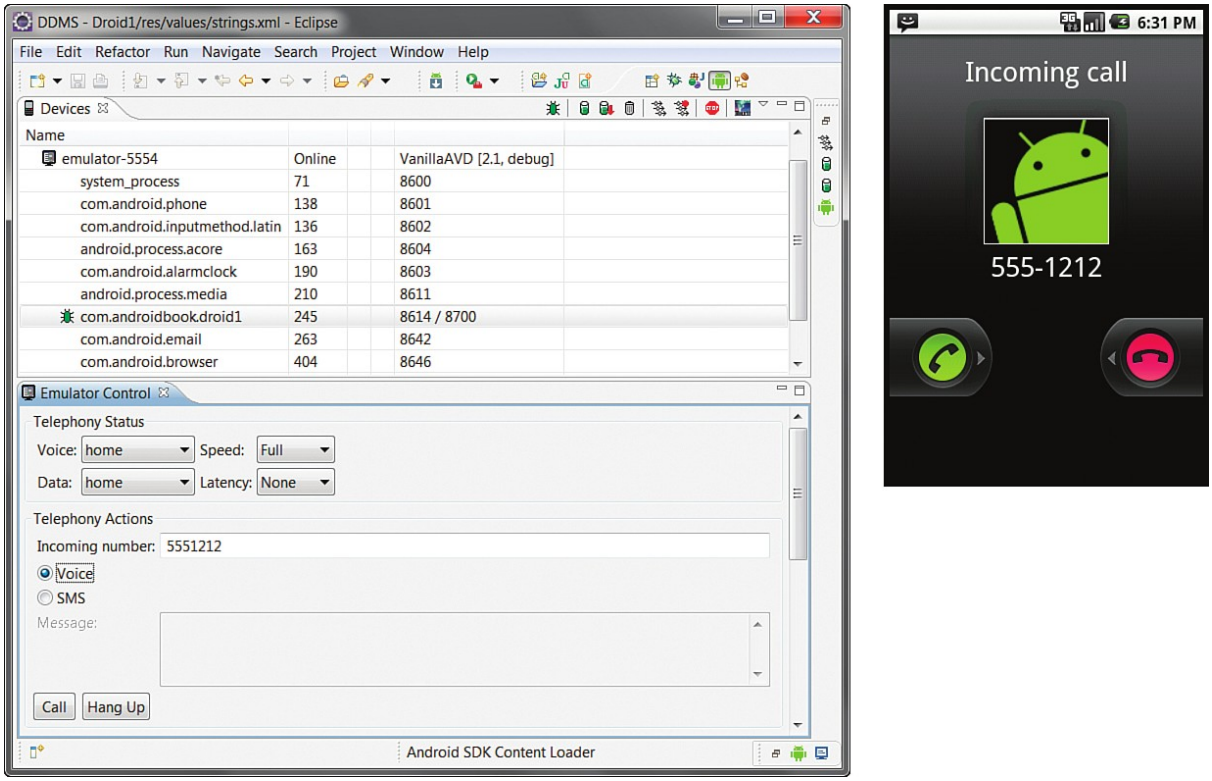

Рис. 2.5. Имитация голосового вызова к эмулятору (справа) на панели **Emulator Control** (Управление эмулятором) в DDMS (слева)

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО
ИМИТАЦИЯ ГОЛОСОВОГО ВЫЗОВА К ЭМУЛЯТОРУ

1. Чтобы инициировать входящий вызов к эмулятору, работающему на вашем компьютере, выполните следующие шаги:
2. В DDMS выберите эмулятор, к которому вы хотите совершить вызов.
   На панели Emulator Control (Управление эмулятором) в разделе Telephony Actions (Управление телефоном) введите номер телефона адресата (например, 5551212).
3. Установите переключатель в положение Voice (Голосовой вызов).
4. Нажмите на кнопку Call (Вызов).
5. В эмуляторе вы должны получить входящий вызов. Ответьте на вызов, щелкнув по кнопке Send (Отправить) в эмуляторе.

6. Вы можете завершить вызов в любое время, щелкнув по кнопке End (Завершить) в эмуляторе или кнопке Hang Up (Сброс) на панели Emulator Control [Управление эмулятором] в DDMS.

ИМИТАЦИЯ ВХОДЯЩИХ SMS НА ЭМУЛЯТОРЕ

На панели Emulator Control (Управление эмулятором) в DDMS вы можете также имитировать отправку SMS к эмулятору (см. рис. 2.6). Отправка SMS осуществляется по такому же принципу, что и голосовой вызов.

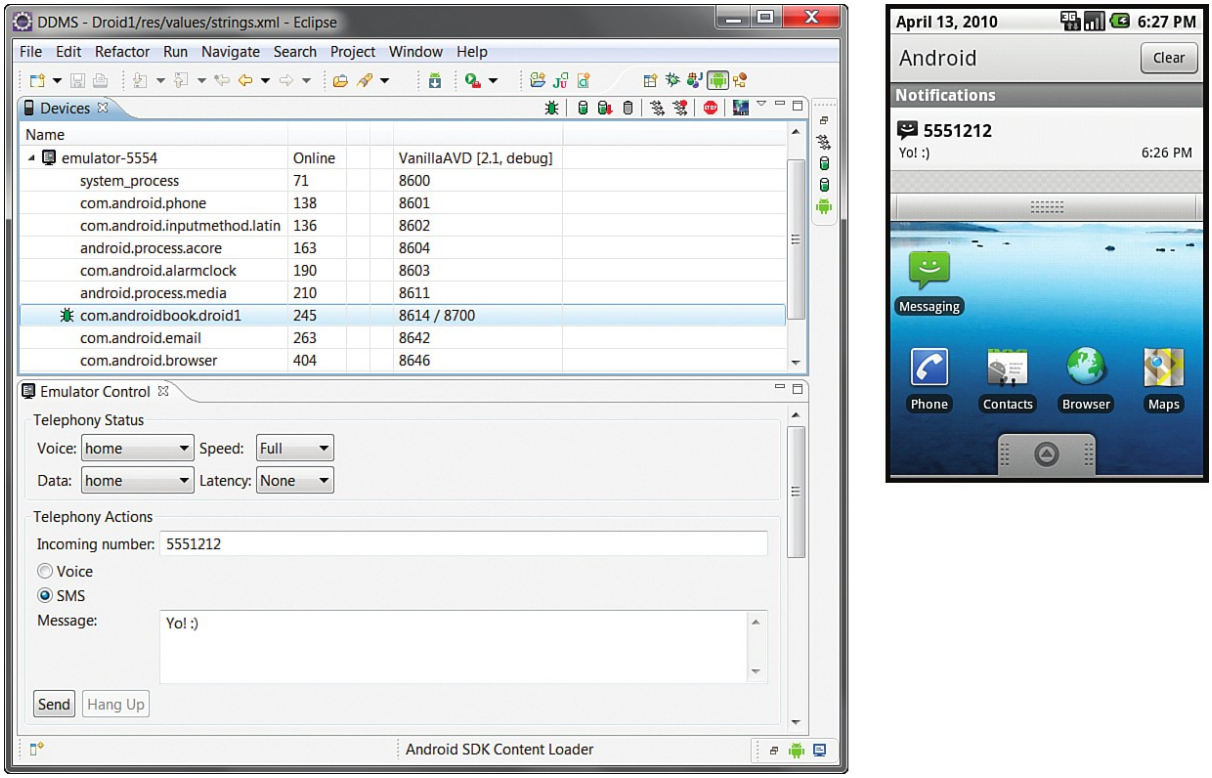

Рис. 2.6. Имитация отправки **SMS** к эмулятору (справа) с помощью панели **Emulator Control** (Управление эмулятором) в DDMS (слева)

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО
ИМИТАЦИЯ ОТПРАВКИ SMS К ЭМУЛЯТОРУ

Чтобы отправить SMS к эмулятору, работающему на вашем компьютере, выполните следующие шаги:

1. В DDMS выберите эмулятор, на который вы хотите отправить SMS.
2. На вкладке Emulator Control (Управление эмулятором) введите номер телефона (например, 5551212) в разделе Telephony Actions (Управление телефоном).
3. Установите переключатель в положение SMS.
4. Введите текст SMS.
5. Нажмите на кнопку Send (Отправить). В эмуляторе вы должны увидеть уведомление о входящем SMS.

Создание скриншотов с эмулятора или мобильного телефона

Одна из возможностей, которая может пригодиться при отладке приложения как на мобильном телефоне, так и на эмуляторе, — это создание скриншота текущего экрана (см. рис. 2.7).

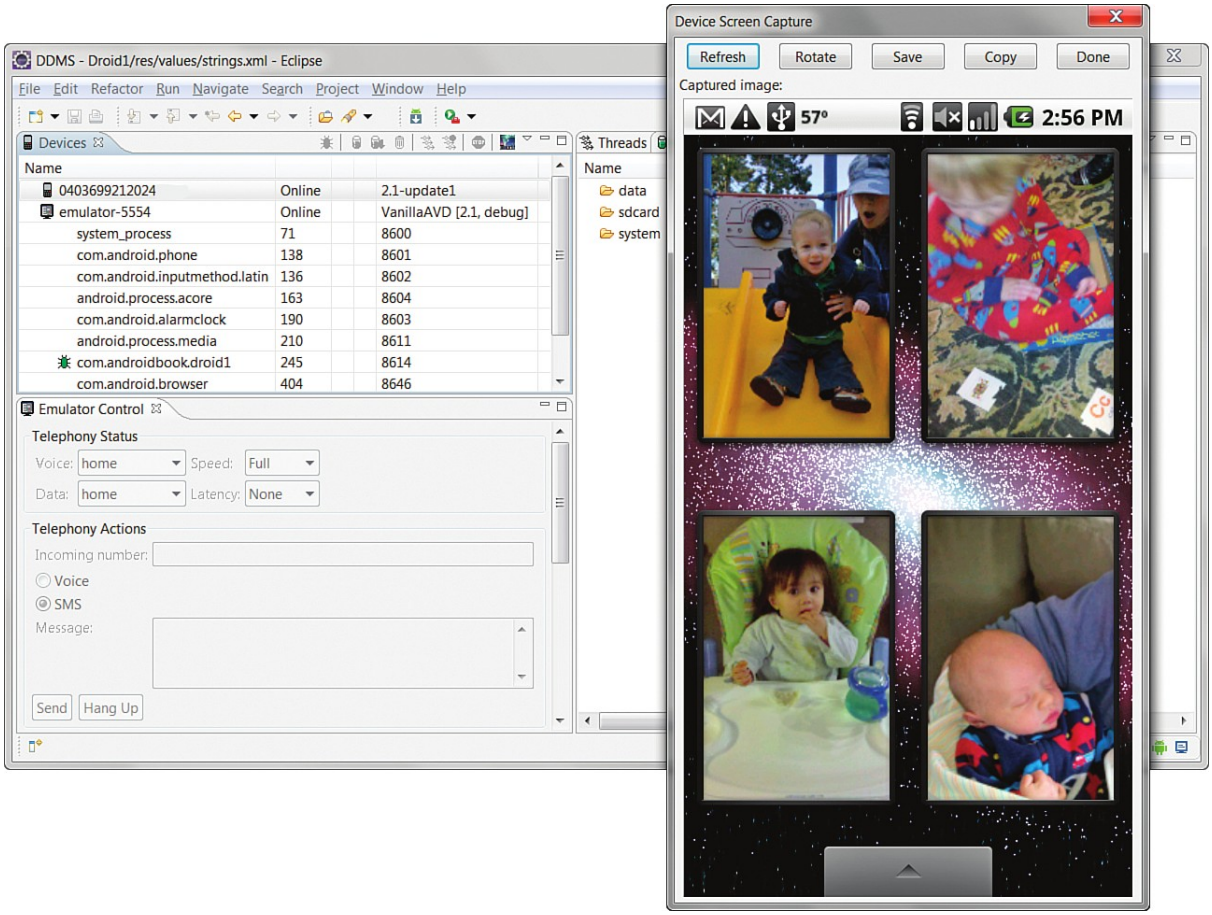

Рис. 2.7. С помощью кнопки **Screen Capture** (Создать скриншот) в DDMS вы можете сделать скриншот

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО СОЗДАНИЕ СКРИНШОТА

Возможность создания скриншота особенно полезна при работе с реальными мобильными телефонами. Чтобы создать скриншот, выполните следующие шаги:

1. Откройте DDMS, выберите устройство (или эмулятор), с которого вы хотите создать скриншот.
2. На выбранном устройстве или эмуляторе выведите изображение экрана, скриншот которого вы хотите создать. Настройте внешний вид, если это необходимо.
3. Нажмите на кнопку с изображением разноцветного квадрата (). Откроется диалоговое окно создания скриншота.
4. В этом диалоговом окне нажмите на кнопку Save (Сохранить), чтобы сохранить скриншот на жесткий диск.

Просмотр журнала

Встроенная в DDMS утилита журналирования LogCat позволяет просматривать консоль журналирования Android. Возможно, вы уже обращали внимание на панель LogCat и диагностическую информацию, представленную на некоторых рисунках этого часа. Более подробно о настройке параметром журналирования конкретных приложений мы поговорим в часе 3.

ФИЛЬТРАЦИЯ ЖУРНАЛИРУЕМОЙ ИНФОРМАЦИИ

В Eclipse вы можете фильтровать данные журнала по необходимым критериям. С помощью тегов вы можете создавать собственные фильтры (подробнее об этом • в приложении Б).

РАБОТА С ЭМУЛЯТОРОМ ANDROID

Эмулятор Android — один из самых мощных инструментов, который есть в распоряжении разработчика. Поэтому каждый разработчик должен умен, им пользоваться и знать его возможности. Эмулятор Android интегрируется в Eclipse IDE с помощью плагина ADT.

ОГРАНИЧЕНИЯ ЭМУЛЯТОРА

Эмулятор Android — очень удобный инструмент, но у него есть некоторые огр.ши чения:

- Эмулятор — не полноценное устройство. Он моделирует лишь общее пов*« дение мобильного телефона, поэтому специфические аппаратные функции не могут быть доступны.
- Такие данные, как геолокационная информация со спутника, информация nfi уровне заряда батареи питания и качестве связи, симулируются компьютером
- Дополнительные возможности устройства, такие как управление камерой, ограни- ченны.
- Телефонные звонки имитируются, поэтому они не могут быть приняты или от- правлены в настоящей сети. SMS также имитируются и не пересылаются по настоящей сети.
- Поддержка USB и Bluetooth отсутствует.

Помните, эмулятор Android не заменяет при тестировании реальное устройство.

Устройства ввода в эмуляторе

Как разработчик, вы можете вводить данные в эмуляторе несколькими способами.

- Компьютерная мышь позволяет щелкать по элементам управления, прокручивать и перетаскивать их (например, ползунковый регулятор урони! громкости).

- Мышь позволяет также имитировать нажатие отдельных клавиш на программной или реальной клавиатуре.

- Клавиатура используется для ввода текста в элементах управления а также для управления определенными состояниями эмулятора.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО

Протестируйте некоторые из способов управления эмулятором:

1. Запустите в Eclipse приложение Droid 1, созданное вами в часе 1.
2. Во время выполнения приложения нажмите сочетание клавиш **Ctrl+F11** или **Ctrl+F12**, чтобы сменить ориентацию экрана эмулятора с портретной на альбомную и наоборот.
3. Нажмите сочетание клавиш **Alt+Enter**, чтобы перейти в полноэкранный режим эмулятора. Затем нажмите **Alt+Enter** еще раз, чтобы вернуться в оконный режим

Помимо этих в вашем распоряжении есть еще целый набор доступных команд Полный их список вы можете найти в официальной документации к эмулятору. Онлайн-версия доступна по адресу: **developer.android.com/guide/developing/tools, emulator.html**.

Обзор системы Android

Исли у вас небольшой опыт работы с устройствами Android, то сейчас подходящее время для изучения системы Android с точки зрения пользователя **И** габл. 2.1 представлены некоторые встроенные возможности Android.

Типы системных экранов Android и их свойства

Тип экрана	Описание	Внешний вид
Экран по умолча-нию	Это основной экран. На нем отображаются основные элементы интерфейса и пункты меню	
Экран набора номера	Это встроенное приложение для совершения телефонных звонков. Помните, что эмулятор обладает ограниченным набором возможно-стей	

Экран обмена текстовыми сообщениями	Встроенное приложения для отправки и получения SMS. Помните, что возможности по обмену сообщениями ограничены
Веб-браузер	Встроенный веб-браузер. Обратите внимание,что эмулятор использует реальное интернет-соединение на вашем компьютере
Список контактов	Вся информация о контактах
Каталог приложений	Здесь отображаются все установленные приложения. Чтобы перейти к нему, вернитесь к экрану по умолчанию и переместите серый ползунок вверх
Экран настройки параметров	Это встроенное приложение для настройки широкого диапазона параметров эмулятора, таких как управление приложениями, настройки звука и экрана, а также региональные и языковые настройки
Инструменты Разработки	Встроенное приложение для настройки Параметров инструментов разработки

Использование обложек (skins) в эмуляторе

Возможности эмулятора, такие как размер экрана, его ориентация, а также то, имеет ли эмулятор физическую или виртуальную клавиатуру, определяются обложкой эмулятора.

Android SDK поддерживает множество различных обложек, которые имитируют различные разрешения телефонных экранов (по умолчанию — HVGA). Совместимость обложки зависит от версии целевой платформы. Подбор подходящей обложки — часть процесса конфигурации AVD.

Использование образа SD-карты в эмуляторе

Для сохранения данных в эмуляторе необходим сконфигурированный образ SD-карты. Хорошо сконфигурированный образ SD-карты пригодится вам, например, для хранения таких медиа-файлов, как графические данные с камеры и звуковые файлы.
Создавать образ SD-карты удобнее всего одновременно с созданием AVD, как это было описано в часе 1. При этом образ SD-карты должен быть не меньше 9 мегабайт.

ДРУГИЕ ИНСТРУМЕНТЫ ANDROID

Несмотря на то, что все наиболее важные инструменты уже были рассмо грены, существует еще несколько специализированных утилит, которые входят в состав Android SDK.

Android Hierarchy Viewer (Просмотр иерархии Android) позволяет раз работникам проверять компоненты пользовательского интерфейса, такие как View Properties (Свойства представления), во время выполнения приложения.

Draw 9-Patch tool (Инструмент растяжения NinePatch) — служит для разработки растягиваемых PNG-файлов.

AIDL Compiler (Компилятор AIDL) — служит для создания удаленных интерфейсов для упрощения взаимодействия между процессами (IPC) на платформе Android.

Утилита командной строки mksdeard позволяет разработчикам создавать автономные образы SD-карт для использования в AVD и эмуляторе.

РАЗРАБОТКА ПРИЛОЖЕНИЙ ANDROID ВНЕ ECLIPSE

Для разработки приложений Android предпочтительнее пользоваться средой разработки Eclipse, но без нее можно обойтись. Плагин ADT для Eclipse предоставляет удобную возможность для выполнения многих задач разработки: создания, отладки, сжатия и лицензирования приложений Android.
Разработчики, не использующие Eclipse, или те, кому нужны более мощные средства отладки, недоступные в плагине ADT, могут использовать эти базовые инструменты непосредственно в командной строке. Все инструменты, перечисленные ниже, находятся в папке **/tools** Android SDK.

- android — служит для создания файлов проекта Android и управления AVD.
- aapt [Android Asset Packaging Tool, инструмент сжатия данных Android) — служит для сжатия файлов проекта Android в APK-файлы для последующей установки на

эмуляторе или мобильном телефоне.
- ddms — имеет собственный пользовательский интерфейс, который напоминает перспективу DDMS в Eclipse.
- adb (Android Debug Bridge) — имеет интерфейс командной строки для взаимодействия с эмулятором или устройством.

ИТОГИ

И питав Android SDK входит множество мощных инструментов для решения общих задач при разработке приложений Android. Документация Android — главный источник информации для каждого разработчика Средство отладки DDMS, интегрированное в среду разработки Eclipse Используется для наблюдения за работой приложения на эмуляторе или физическом устройстве. Эмулятор Android может использоваться для выКопнения и отладки приложений Android почти без необходимости использования реального устройства. Существует также множество других инструментов для взаимодействия с мобильными телефонами и эмуляторами в интерфейсе командной строки, а также специальные утилиты для разработки пользовательских интерфейсов Android и сжатия приложений.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Документация Android, постявляемая с Android SDK, и документация на вебсайте **developer.android.com** одинаковы?

Ответ. Нет. Вариант документации, входящей в состав SDK, был актуален лишь на время релиза SDK, поэтому он больше подходит для утсановленной версии Android SDK. Документация по Android SDK, доступная в онлайне, - самый свежий вариант. Если у вас нет доступа в сеть или интернет-соединение слишком медленное, вы можете воспользоватся локальным вариантом. В остальных случаях мы рекомендуем вам онлайн-документацию.

Вопрос: Влияет ли обложка эмулятора на его возможности?

Ответ. Да. Обложка эмулятора задает размер экрана и его ориентацию. У каждой из обложек есть также вспомогательная клавиатура и кнопки. Одни обложки используют реальную клавиатуру, другие виртуальную.

Вопрос: Достаточно ли проведения тестирования приложения на эмуляторе?

Ответ. Нет. Эмулятор Android лишь имитирует функциональность реального устройства, что может сохранить ваше время и деньги при разработке проектов Android. Это удобный инструмент для тестирования, но он не может полностью заменить реальное устройство. С эмулятора вы не можете определять реальные геолокационные координаты или совершать телефонные звонки. Кроме того, эмулятор — это универсальное устройство, оно не способно моделировать все характеристики определенного мобильного телефона. Поэтому, даже если ваше приложение хорошо работает на эмуляторе. это не гарантиря того, что оно будет работать на реальном устройстве.

ПРАКТИКУМ

Контрольные вопросы

1. Какие возможности доступны в перспективе DDMS?
A. Сохранение скриншотов экрана эмулятора и мобильного телефона.
B. Просмотр файловой системы эмулятора или мобильного телефона.
C. Контроль потока и информации о стеке в системе Android.
D. Остановка процессов.
E. Моделирование входящих телефонных звонков и SMS к эмуляторам.
F. Все вышеперечисленное.

2. Достаточно ли использования эмулятора Android для отладки приложений?

3. Для каких версии платформы могут быть написаны приложения Android?

4. Эмулятор Android обладает ограниченными возможностями и поддерживаег только одну конфигурацию экрана. Правда ли это?

Ответы

1. Правильный ответ F . Перспектива DDMS может применяться для отслеживания, обзора и взаимодействия с эмуляторами и мобильными телефонами.
2. Нет. Эмулятор Android позволяет производить отладку, но кроме этою вы должны также произвести отладку на реальном устройстве, чтобы быть уверенным в работоспособности приложения.

3. Существует множество версий платформ и их число увеличивается с каждым новым выпуском SDK. Одни из самых распространенных версий платформы на данный момент — Android 1.1, Android 1.5. Android 1.6, Android 2.0, Android 2.0.1 и Android 2.1. В платформы Android версии выше 1.1 по необходимости может быть включен Google API. Версия платформы влияет на профиль AVD, который вам необходимо создать для запуска приложений на эмуляторе Android.

4. Нет, это не так. Несмотря на то, что у эмулятора Android ограниченные возможности, он может поддерживать несколько различных обложек. Полный список поддерживаемых обложек вы можете найти в Android SDK and AVD Manager.

Упражнения

1. Запустите эмулятор Android и обратите внимание на доступные настройки. Измените настройки языка. Попробуйте удалить приложение.

2. Запустите эмулятор Android и настройте экран по умолчанию. Измените фоновое изображение. Установите виджет. Понаблюдайте за тем, как эмулятор имитирует мобильный телефон. Обратите внимание на ограничения, такие как совершение голосового вызова.

3. Попробуйте запустить инструмент **Hierarchy Viewer** (Просмотр иерархии) в проекте Droidl, созданном вами в часе 1. Обратите внимание на созданные вами элементы пользовательского интерфейса `TextView`.

ЧАС 3. СОЗДАНИЕ ПРИЛОЖЕНИ ANDROID

Вопросы, рассматриваемые в этом часе:

- **проектирование типичного приложеиня Android;**
- **использование контекста приложения;**
- **работа с деятельностями, интентами и диалоговыми окнами;**
- **журналирование информации.**

Для описания компонентов приложения каждой поатформы употребляется различная терминология. Три самых выжных класса на платформе Android это: Context (Контекст), Activity (Деятельность), и Intent (Интент). Кроме этих компонентов, есть другие, также доступные разработчикам, но именно эти три компонента входят в состав практически каждого приложения Android. В этом часе мы сосредоточим ваше внимание на понимании общих черт приложений Android. Мы также рассмотрим некоторые служебные классы, которые могут помочь разработчикам в отладке приложений.

ПРОЕКТИРОВКА ТИПИЧНОГО ПРИЛОЖЕНИЯ ANDROID

Приложение Android можно представить как набор задач, каждая из которых называется деятельностью (activity). Каждая деятельность приложения имеет определенное назначение и пользовательский интерфейс. Чтобы лучше понять этот принцип на конкретном примере, составим план небольшого игрового приложения «Chippy's Revenge».

Проектирование возможностей приложения

Проект приложения «Chippy's Revenge» очень прост. В этой игре будет пять экранов.

- **Экран-заставка —** обычная заставка с изображением логотипа и ее версии при загрузке. Во время процесса загрузки может воспроизводится фоновая музыка.

- **Меню** — на этом экране пользователь может переходить по пунктам меню, например **Play, Scores** и **Help**.

- **Play** —экран, где происходит основной игровой процесс.

- **Scores** — на этом экране отображается статистика игровых достижений (включая достижения других игроков), что дает стимул для улучшения собственного рекорда.

- **Help** — на этом экране приводятся инструкции по игре, включая описание управления, целей игры, игровой тактики, а также подсказки и советы.

Знакомое описание? Это типичный проект любого мобильною приложения для любой платформы, в данном случае игры.

КСТАТИ

Разумеется, вы можете разработать собственный пользовательский интерфейс, какой пожелаете. Каких-либо строгих требований к интерфейсу на платформе Android нет, но при этом приложение должно быть стабильным, интерактивным и уметь взаимодействовать с системой Android. Однако все успешные и наиболее популярные приложения привлекают интерес пользователей в первую очередь удобным пользовательским интерфейсом. Поэтому, вместо того, чтобы придумывать новый интерфейс, если есть возможность, лучше следовать приведенному выше плану. Таким образом, пользователю вашего приложения не придется тратить время и силы на его изучение.

Требования к деятельности приложения

- Вы должны реализовать пять классов деятельности — по одному для каждой возможности игры.

- `SplashActivity` — деятельность запуска по умолчанию. Она выводит на экран простую информацию о запускаемой игре (это может быть изображение логотипа игры) с фоновым музыкальным сопровождением в течение нескольких секунд, после которого запускается деятельность `MenuActivity`.

- `MenuActivity` — назначение этой деятельности можно понять из се названия. В ее макете несколько кнопок, каждая из которых связана с определенной возможностью приложения. Деятельность, связанная с каждой из кнопок, запускается обработчиком события `onClick()`.

- `PlayActivity` — основная часть логики приложения реализуется этой деятельностью. Эта деятельность выводит на экран графику во время игрового процесса, обрабатывает данные, полученные от пользователя, ведет подсчет заработанных очков и следит за всей игровой динамикой.

- `ScoresActivity` — эта деятельность такая же простая, как и `SplashActivity`. В основном она занимается тем, что загружает данные достижении игроков в элемент пользовательского интерфейса `TextView` в пределах собственного макета.

- `HelpActivity` — эта деятельность практически идентична деятельности `ScoreActivity`, за исключением того, что вместо вывода на экран информации об игровых достижениях она выводит на экран текст справки. Здесь можно добавить прокрутку элемента `TextView`.

У каждого класса деятельности должен быть собственный, ассоциированный с ним файл макета, хранящийся в ресурсах приложения. Вы можете использовать один файл макета деятельности ScoresActivity и HelpActivity, но обычно в этом нет необходимости. Тем не менее, если создадите один макет для двух деятельностей, то вы можете разместить фоновое изображение и текст в элементе `Textview` вместо файла макета.

На рис. 3.1 изображен проект игры «Chippy's Revenge» версии 0.0.1 в виде блок-схемы.

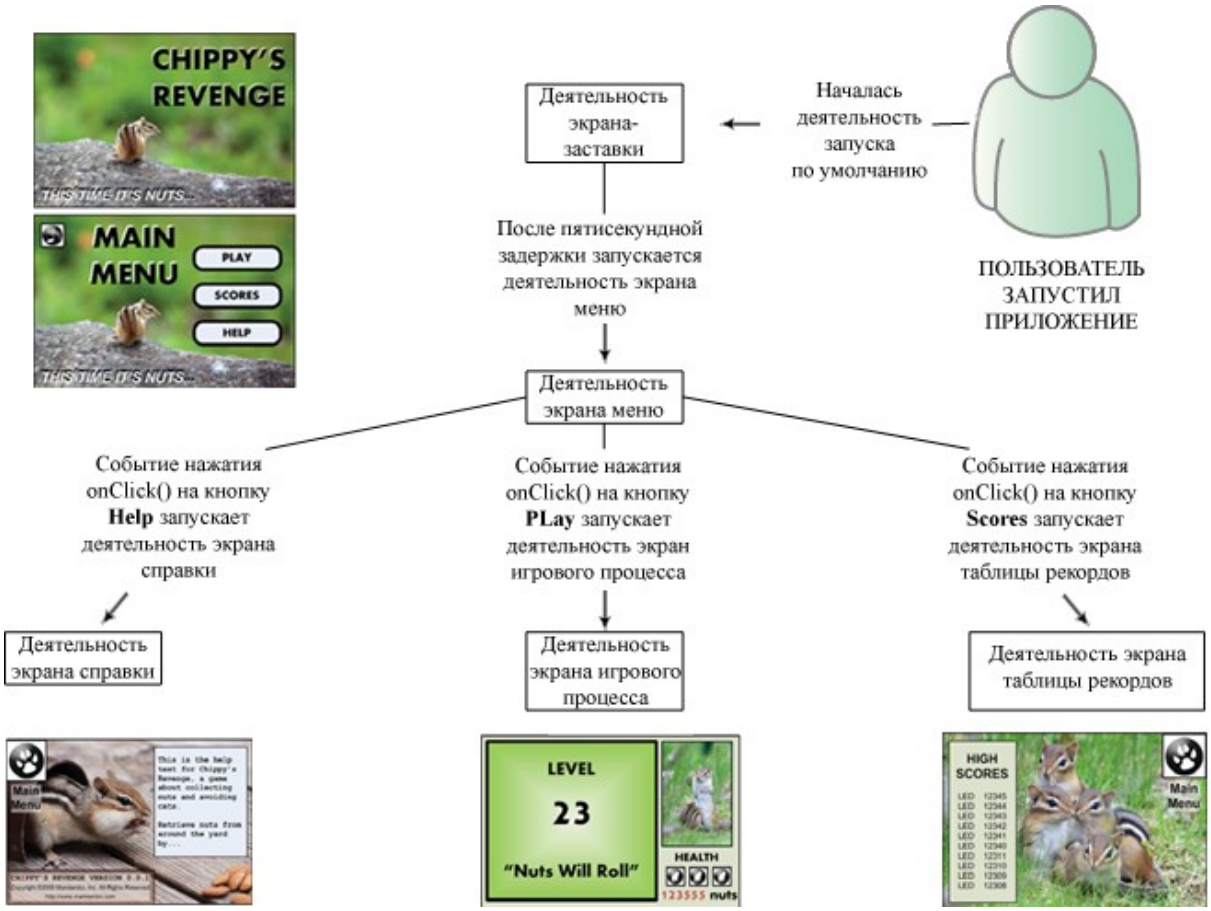

Рис. 3.1. Проект простого приложения Android («Chippy's Revenge»)

Реализация функциональности приложения

Теперь, когда вы понимаете, как выглядит проект типичного приложения Android, вы, возможно, задались вопросом, как функционирует такой проект на программном уровне.

Мы говорили о том, что каждая деятельность имеет собственный пользовательский интерфейс, описанный в отдельном фарше ресурса макета. Возникает несколько вопросов:

- Как управлять состоянием приложения?
- Как сохраняются настройки?

- Как запустить определенную деятельность?

Имен примерное представление об игровом приложении, теперь настало время рассмотреть процесс создания приложения Android более детально. Начнем с контекста приложения.

КОНТЕКСТ ПРИЛОЖЕНИЯ

Контекст приложения — это основа всей высокоуровневой функциональности приложения. Контекст приложения служит для получения доступа к настройкам и ресурсам, используемым несколькими экземплярам деятельности.

Получить доступ к контексту приложения текущего процесса вы можете методом `getApplicationContext ()`, например:

```
Context context = getApplicationContext();
```

Поскольку класс Activity происходит из класса Context, вы можете ис пользовать оператор this вместо явного указания контекста приложения.

ВНИМАНИЕ!
Контекст `Activity` не рекомендуется использовать во всех случаях - это можем привести к утечкам памяти. Обсуждение данной проблемы выходит далеко рамки книги, поэтому исчерпывающую информацию по этой теме вы можете поискать в официальном блоге разработчиков Android по адресу: **android-developers. blogspot.com/2009/01/avoiding-memory-leaks.html.**

Как только необходимый контекст приложения был успешно получен, его можно использовать для доступа к возможностям и службам приложения.

Доступ к ресурсам приложения

Доступ к ресурсам приложения осуществляется методом `getResources ( )` контекста приложения. Самый прямой способ доступа к ресурсам - через URI, определенный **в** автоматически сгенерированном классе `R.java`. В следующей строке кода к экземпляру строкового типа из ресурсов приложения по его идентификатору `hello`:

```
String greeting = getResources ().getString (R.string,hello);
```

Доступ к параметрам приложения

Получить доступ к общедоступным параметрам приложения вы сможете методом контекста приложения `getSharedPreferences()`.Класс `SharedPreferences` используется для хранения данных приложения, таких как настройки конфигурации. Каждому объекту `SharedPreferences` распределять настройки по категориям или хранить все настройки вместе одним комплектом. Допустим, вы пользователей и игровой статистики, например количество жизней или уровень здоровья игрового персонажа. В следующем фрагменте кода создается набор общедоступных параметров `GemePref`, и некоторые из них сохраняются:

```
SharedPreferences settings = getSharedPreferences("GamePrefs",
MODE_PRIVATE);
SharedPreferences.Editor prefEditor = settings.edit();
prefEditor.putString("UserName", "Spunky");
prefEditor.putBoolean("HasCredits", true);
prefEditor.commit();
```

Таким образом, получение доступа к параметрам настройки осуществляется через объект `SharedPreferences:`

```
SharedPreferences settings = getSharedPreferences("GamePrefs",
MODE_PRIVATE);
String userName = settings.getString("UserName", "Chippy Jr.
(Default)");
```

Доступ к другим возможностям приложения с помощью контекстов

Контекст приложения обеспечивает доступ ко многим высокоуровневым возможностям приложения. Вот задачи, которые вы можете выполнять с помощью контекста приложения:

- запускать экземпляры Activity;
- получать доступ к активам приложения;
- отправлять запрос провайдеру на системном уровне (например, службе позиционирования);
- управлять собственными файлами приложения, папками и базами данных,
- просматривать и изменять права приложения.

Первый элемент из этого списка, запуск экземпляров Activity, возможно, самое распространенное применение контекста приложения.

РАБОТА С ДЕЯТЕЛЬНОСТЬЮ

Класс Activity — это ключевой класс каждого приложения Android. Большую часть времени вы будете определять и реализовывать деятельность для экранов вашего приложения.

В вашем игровом приложении «Chippy's Revenge» необходимо реализовать пять различных классов Activity. В процессе игры пользователь переходит от одной деятельности к другой, взаимодействуя со средствами управления каждой деятельности.

Запуск деятельности

Существует несколько способов запуска деятельности, среди которых:

- назначение деятельности запуска в файле манифеста;

. запуск деятельности с помощью контекста приложения;

- запуск дочерней деятельности из родительской деятельности.

НАЗНАЧЕНИЕ ДЕЯТЕЛЬНОСТИ ЗАПУСКА В ФАЙЛЕ МАНИФЕСТА

Каждое приложение Android должно иметь деятельность по умолчанию в пределах файла манифеста Android. Если вы посмотрите на файл манифеста проекта **Droid1**, то заметите, что деятельность `DroidActivity` назначена по умолчанию.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Другие классы `Activity` могут быть назначены для запуска в определенных случаях. Эти вторичные точки входа конфигурируются в файле манифеста Android специальными фильтрами.

В игре «Chippy's Revenge» логично предположить, что деятельность `SplashActivity` будет деятельностью по умолчанию.

ЗАПУСК ДЕЯТЕЛЬНОСТИ С ПОМОЩЬЮ КОНТЕКСТА ПРИЛОЖЕНИЯ

Наиболее распространенный способ запуска деятельности — методом контекста приложения `startActivity()`. Этот метод принимает единственный параметр, называемый интентом. Мы еще вернемся к обсуждению это го термина, но сначала рассмотрим метод `startActivity()`.

В следующем фрагменте происходит вызов места startActivity () с явно указанным интентом:

```
startActivity(new Intent(getApplicationContext(),
MenuActivity.class));
```

Здесь интент запрашивает запуск деятельности `MenuActivity` по ее классу. Этот класс должен быть реализован в любом месте пакета.

Поскольку класс `MenuActivity` пределен в пределах пакета данного приложения , он должен быть зарегистрирован как деятельность в пределах файла манифеста Android. Фактически вы можете использовать этот метод для запуска любой деятельности вашего приложения, однако это лишь один из нескольких способов запуска деятельности .

ЗАПУСК РОДСТВЕННОЙ ДЕЯТЕЛЬНОСТИ

Вместо запуска одной конкретной деятельности иногда может возникнуть необходимость запуска родственных деятельностей для получения требуемого результата. В этом случае используется

метод `Activity. startActivityForResult ()`. Результат вычисления будет возвращен параметром интента метода вызываемой деятельности `onActivityResult ()`. Далее мы поговорим о передаче данных с помощью интента более подробно.

Управление состоянием деятельности

Приложения могут быть прерваны при возникновении важных событий, таких как входящий телефонный вызов. Так как одновременно может быть активным только одно приложение , в любой момент времени может выполняться только одна деятельность приложения.

Приложения Android кроме памяти, ресурсов и данных управляют также своим состоянием. Операционная система Android может прекратить приостановленную деятельность, находящуюся в ожидании , при низком уровне свободной памяти. Другими словами, любая развернутая деятельность может быть принудительно завершена. Это значит, что приложение Android должны запоминать свое состояние и быть готовыми к остановке или завершению в любой момент времени.

ОБРАБОТКА СОБЫТИЙ ДЕЯТЕЛЬНОСТИ

У класса `Activity` есть несколько обработчиков событий, которые позволяют деятельности реагировать на такие события, как приостановка и возобновление. В табл. 3.1 представлены наиболее важные методы событий.

Метод обработки событий	Описание	Рекомендации
onCreate()	Вызывается после запуска или перезапуска деятельности	Инициализирует статические данные деятельности. Связывает с необходимыми данными или ресурсами. Задает макет методом `setContextView()`
onResume()	Вызывается при восста-Новлении деятельности	Получает монопольные ресурсы. Запускает воспроизведение любой анимации, аудио- или видеоконтента
onPause()	Вызывается при свертывании деятельности	Сохраняет незафиксированные данные. Деактивирует и выпускает монопольные ресурсы. Останавливает воспроизведение всех видео-, аудиофайлов и анимации
onDestroy()	Вызывается при завершения	Удаляет все статические данные деятельности. Отдает все используемые ресурсы

Основной поток часто называют потоком пользовательского интерфейса или UI-потоком, т. к. именно здесь происходит внутренняя прорисовка пользовательского интерфейса. Деятельность должна обрабатывать события очень быстро, чтобы не блокировать основной поток. Если основной UI- поток будет заблокирован слишком долго, система

Android завершит деятельность. Особенно важно отвечать как можно быстрее во время вызова метода onPause (), когда задача с более высоким приоритетом (например, входящий телефонный звонок) приводит к свертыванию приложения.

На рис. 3.2 представлен порядок, в котором совершается обработка событий деятельности.

СОХРАНЕНИЕ СОСТОЯНИЯ ДЕЯТЕЛЬНОСТИ

Каждая деятельность может иметь собственные параметры — так же, как общедоступные параметры приложений. Вы можете получить доступ к этим параметрам с помощью метода деятельности `getPreferences()`. Этот механизм может быть полезен для сохранения информации о состоянии. Например, деятельность `PlayActivity` вашего игрового приложения может использовать эти параметры для отслеживания текущего уровня, счета, здоровья игрового персонажа и других статистических данных.

Завершение деятельностей

Чтобы завершить деятельность, вам необходимо сделать вызов метода `finish()`. Существует несколько различных версий этого метода в зависимости от того, завершает ли деятельность саму себя или завершает работу Другой деятельности.

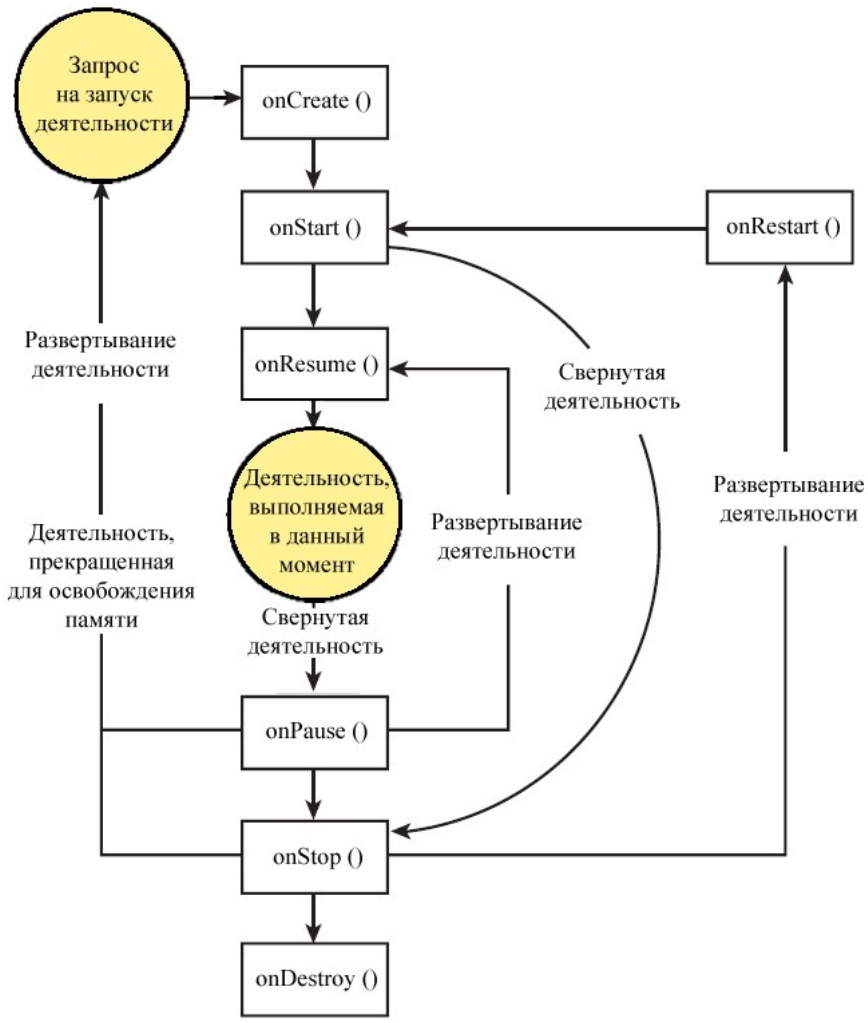

Рис. 3.2. Наиболее важные методы обработки событий деятельности

В вашем приложении вы можете возвращаться с экранов `Scores`, `Play` и `Help` к экрану Меню завершением деятельностей `ScoresActivity`, `PlayActivity` или `HelpActivity`.

РАБОТА С ИНТЕНТАМИ

Объект Intent заключает в себе запрос задачи, используемый операционной системой Android. Когда метод startActivity() вызывается с параметром интента, система Android сравнивает действие Inentent с соответствующей деятельностью в системе. После этого деятельность запускается.

Система Android обрабатывает все разрешения интентов. Интент может быть строго определенным и включать запрос на определенную деятельность, подлежащую запуску. Или менее конкретным — в этом случае может быть запущена любая деятельность, соответствующая определенным критериям. Более подробную информацию о разрешении интентов вы можете найти в документации по Android.

Передача данных с помощью интентов

Интенты могут использоваться для передачи данных между деятельностями. Для этого вы должны включить дополнительные данные в пределах интента.

Для внедрения фрагментов дополнительных данных в интент используется метод `putExtra ()` с тем типом объекта, который вы хотите включить. Согласно программным требованиям Android, каждый фрагмент дополнительных данных должен начинаться с префикса пакета (например, `com. androidbook.chippy.NameOfExtra`).

В следующем примере интент включает в себя дополнительную информацию — значение текущего игрового уровня, являющееся целым числом:

```
Intent intent=new Intent(getApplicationContext(),HelpActivi ty.class);
intent.putExtra("com.androidbook.chippy.LEVEL", 23) ;
startActivity(intent);
```

После запуска класса `HelpActivity` метод `getintent ()` может использоваться для доступа к интенту. Дополнительные данные могут быть извлечены с помощью соответствующих методов. Например:

```
Intent callinglntent=getlntent();
int helpLevel=callingIntent.getlntExtra("com.androidbook.
chippy.LEVEL",1);
```

Этот небольшой фрагмент данных может использоваться для получения специальных подсказок по прохождению определенного игрового уровня.

Для родительской деятельности, запустившей дочернюю деятельность методом `startActivityForResult()`, результат как параметр будет передан методу `onActivityResult()` со значением Intent. Данные интента могут быть извлечены и использоваться родительской деятельностью.

Использование интентов для запуска других приложений

Первоначально приложение может запускать только те классы деятельности, которые определены в его собственном пакете. Однако если на это есть соответствующие права, приложение может также запускать внешние классы деятельности сторонних приложений.

Существуют четко определенные действия интента для многих пользовательских задач. Например, вы можете создать действия интента для выполнения таких задач как:

- запуск встроенного веб-браузера и переход по URL;
- запуск веб-браузера с выводом строки поиска,
- запуск встроенного приложения для набора номера с указанием номера телефона;
- запуск встроенного приложения Maps о указанием координат;
- запуск сервиса Google Street View с указанием места расположения;
- запуск встроенного приложения для работы с камерой в обычном или видеорежиме;
- запуск средства выбора рингтона для мобильного телефона;
- запись звука.

Ниже приведен пример создания простого интента с предопределенным действием (`ACTION_VIEW`) для запуска веб-браузера и перехода на указанный URL:

```
Uri address = Uri.parse("http://www.perlgurl.org");
Intent surf = new Intent(Intent.ACTION_VIEW, address);
startActivity(surf);
```

В этом примере создается интент с предопределенным действием и некоторыми данными. Действие в этом случае означает просмотр чего-либо. В роли данных выступает URL, по котором необходимо перейти для вывода веб-страницы.

Здесь происходит запуск деятельности браузера и ее развертывание, при этом исходная деятельность вызова приостанавливается и переходит в фоновый режим. Когда пользователь заканчивает работу с браузером, кнопкой **Back** он может вернутся к исходной деятельности.

Приложения могут также создавать свои собственные типы интентов и позволять другим приложениям вызывать их, включая тесно интегрированные наборы приложений.

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

На сайте openintents.org вы можете найти список действий интента по адресу: **www.openintents.org/en/intentstable**. В этот список входят действия, встроенные в Android, а также приложения сторонних разработчиков.

РАБОТА С ДИАЛОГОВЫМИ ОКНАМИ

Экраны мобильных телефонов обычно очень малы, поэтому каждый пиксель пользовательского интерфейса имеет большое значение. Иногда вам может понадобиться обработать простое взаимодействие с пользователем — для этого нет необходимости создавать новую деятельность. В таких случаях создание диалогового окна деятельности может быть наиболее оптимальным решением. Диалоговые окна могут быть полезными при создании простого пользовательского интерфейса, не требующего нового свободного экрана или деятельности для функционирования. Вместо этого вызываемая деятельность создает диалоговое окно, которое может иметь собственный макет, и пользовательский интерфейс с кнопками и элементами ввода.

В табл. 3.2 представлены наиболее важные методы создания и управления диалоговыми окнами деятельности.

Некоторые методы диалоговых окон класса Activity

Метод	Назначение
Activity.showDialog()	Отображает диалоговое окно по необходимости
Activity.onCreateDialog()	Обработка события первичного создания диалогового окна и добавления к группе деятельностей
Activity.onPrepareDialog()	Обработка события непрерывного обновления диалогового окна. Диалоговые окна создаются один раз и могут быть использованы деятельностью многократно. Этот метод позволяет диалоговым окнам обновляться непосредственно перед показом для каждого вызова метода `showDialog()`
Activity.ondismissDialog()	Отзывает диалоговое окно и возвращает к деятельности. Диалоговое окно остается доступным для повторного использования вызова метода `showDialog()`
Activity.removeDialog()	Полностью удаляет диалоговое окно из пула диалогового окна деятельности

Классы деятельности могут включать более одного диалогового окна. Каждое диалоговое окно может быть создано один раз и использоваться многократно.

Существует довольно много готовых типов диалоговых окон, доступных для использования, в дополнение к основному диалоговому окну. Вот некоторые из них: `AlertDialog`, `CharacterPickerDialog`, `DatePickerDialog`, `rogressDialog` и `TimePickerDialog`.

Вы можете также создать полностью переделанное диалоговое окно с помощью метода `Dialog.setContentView ()` и XML-файла макета. Для получения доступа к средствам управления диалогового окна используется метод `Dialog.findViewById ()`.

ЖУРНАЛИРОВАНИЕ ДАННЫХ ПРИЛОЖЕНИЯ

Android предоставляет полезный класс утилиты журналирования `android.util.Log`. Записи журнала сортируются по релевантности (и подробности) с наиболее серьезными ошибками. В табл. 3.3 перечислены некоторые из часто используемых методов журналирования класса `Log`.

Таблица 3.3

Наиболее часто используемые методы журналирования

Метод	Назначение
Log.e()	Журналирование ошибок
Log.w()	Журналирование предупреждений
Log.i()	Журналирование информационных сообщений
Log.d()	Журналирование отладки
Log.v()	Журналирование подробных записей

ВНИМАНИЕ!

Злоупотребление утилитой Log может привести к уменьшению производительности приложения. Отладка и подробное журналирование должны использоваться только при разработке и должны быть удалены перед публикацией приложения.

Первый параметр каждого метода Log — это строка, называемая тегом. Часто при программировании в Android определяют глобальную статическую переменную строкового типа для представления всего приложения или определенной деятельности приложения, чтобы фильтры журналирования могли ограничивать объем журнала только необходимыми данными.

Например, вы можете определить строковую переменную TAG следующим образом:

```
private static final String TAG="MyApp";
```

Теперь каждый раз при использовании метода Log, вы указываете этот тег. Запись журнала должна выглядеть примерно следующим образом:

```
Log.i(TAG, "In onCreate() callback method");
```

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

ВЫ можете использовать утилиту LogCat, входящую в состав Eclipse, для фильтрации сообщений журнала по тегу. За более подробной информацией обратитесь к приложению Б.

ИТОГИ

В этом часе вы узнали, как различные приложения Android могут быть спроектированы с помощью трех компонентов приложения: Context, Activity и Intent. Каждое приложение Android включает одну или более деятельностей. Высокоуровневая функциональность приложения доступна посредством контекста приложения. Каждая деятельность выполняет определенную функцию и (обычно) имеет собственный макет или пользовательский интерфейс. Деятельность запускается, когда система Android сравнивает объект интента с наиболее подходящей деятельностью приложения, основанную на действии и информации, заданных в интенте. Интенты могут использоваться также для передачи данных от одной деятельности к другой.

В дополнение к изучению основ взаимодействия приложений Android вы научились также использовать полезные классы утилит Android, такие как журналирование приложений, которое может помочь оптимизировать разработку приложений Android и их отладку.

ВОПРОСЫ И ОТВЕТЫ

Вопрос. Как спроектировать интерактивное приложение, которое не будет завершаться при низком уровне свободной памяти?

Ответ. Приложения могут уменьшать (но не исключать) риск непредвиденного завершения при низком уровне свободной памяти путем управления состоянием деятельности. Это значит, что приложение должно использовать соответствующие обратные вызовы деятельности согласно приведенным рекомендациям. Очень важно, чтобы приложения получали ресурсы только при необходимости и высвобождали их как можно быстрее, насколько это возможно.

Вопрос. Как можно спроектировать форму ввода для приложения Android?

Ответ. Мобильные приложения должны быть готовы к внезапной остановке и возобновлению в любое время. Типичная веб-форма состоит из различных полей ввода и кнопок Submit (Отправить), Clear (Сброс), Cancel (Отмена), что не очень хорошо подходит для мобильного приложения. Вместо этого рассмотрите возможность непосредственного приема данных по мере их ввода пользователем. Это сократит использование аппаратных ресурсов для обслуживания данных к минимуму, поскольку состояние деятельности изменяется без привлечения внимания пользователя.

ПРАКТИКУМ
Контрольные вопросы

1. Какой из этих экранов следует показывать пользователю при запуске приложения?
   A. Экран меню.
   B. Экран-заставка.
   C. Экран игры.

2. Платформа Android обладает простым методом для сохранения настрой приложения. Правда ли это?

3. Какой из рекомендуемых способов получения экземпляра контекста требуется многими вызовами Android?

```
A.   Context context=(Context) this;
B.   Context context=getAndroidObject (CONTEXT) ;
C.   Context context=getApplicationContext () ;
```

4. Класс android.util.Log поддерживает шесть типов журналирования. Правда ли это?

Ответы

1. Правильный ответ: Б. Перед началом игры пользователь должен увидеть экран-заставку.

2. Да, это так. Для сохранения простых настроек используется класс `SharedPreferences`.

3. Правильный ответ: В. Таким способом вы получаете контекст, привязанный к приложению. Использование контекста деятельности, как в варианте А, также допустимо, но не рекомендуется.

4. Нет. Класс Log поддерживает пять типов журналирования: ошибочный, предупреждающий, информационный, отладочный и подробный.

Упражнения

Добавьте тег журналирования в проект Droid 1. В методе обработки события onCreate () добавьте сообщение журналирования информационного типа методом Log. i (). Запустите приложение и просмотрите результаты журналирования.

Реализуйте несколько методов обработки событий Activity в дополнение к методу onCreate(), например onStart(), onRestartf), onResume (), onPause (), onStop () и onDestroy (). Добавьте сообщение журнала к каждому методу обработки событий и запустите приложение. Просмотрите полученный журнал и проследите за деятельностью приложения. Затем попробуйте некоторые другие ситуации, такие как приостановка или переход в ждущий режим приложения и затем воз-обновите процесс. Сымитируйте входящий голосовой вызов. Откройте журнал приложения и обратите внимание на то, как деятельность реагирует на такие события.

Час 4. УПРАВЛЕНИЕ РЕСУРСАМИ ПРИЛОЖЕНИЯ

Вопросы, рассматриваемые в этом часе:

- **использование ресурсов приложения и системные ресурсы;**
- **работа с простыми ресурсами;**
- **работа с рисунками;**
- **работа с макетами;**
- **работа с файлами;**
- **работа с другими типами ресурсов.**

При генерировании пользовательского интерфейса приложениями Android используются строковые, графические и другие типы ресурсов. Ресурсы могут быть включены в проект Android согласно четко определенной иерархии. В этом часе мы рассмотрим наиболее распространенные типы ресурсов, используемые в приложениях Android, вы узнаете как они хранятся и как к ним получить доступ в программном коде.

РЕСУРСЫ ПРИЛОЖЕНИЯ И СИСТЕМНЫЕ РЕСУРСЫ

Ресурсы бывают двух типов: ресурсы приложения и системные ресурсы. Ресурсы приложения определяются разработчиками в файлах проектов Android отдельно для каждого приложения. Системные ресурсы — это общие ресурсы, определенные платформой Android, они доступны для всех приложений через Android SDK.

При выполнении приложения вы можете получать доступ к обоим типам ресурсов. Вы можете также получать доступ к ресурсам из других скомпилированных ресурсов, таких как XML-файлы макетов для определения атрибутов элементов пользовательского интерфейса.

Ресурсы приложения

Ресурсы приложения создаются и хранятся в файлах проектов Android, в папке **/res.** Согласно строго определенной, но гибкой структуре каталогов ресурсы сортируются, определяются и компилируются с пакетом приложения. В большинстве случаев ресурсы приложения не используются системой Android.

ХРАНЕНИЕ РЕСУРСОВ ПРИЛОЖЕНИЯ

В процессе программирования данные, используемые приложением, обычно представлены и качестве ресурсов. Объединение ресурсом и компиляция в единый пакет приложения имеет рад преимуществ:

- Код становится более понятным и легким для чтения, что уменьшает количество возможных ошибок.
- Ресурсы сортируются по типу и поэтому не повторяются.
- Ресурсы расположены удобно для обеспечения гибкости настройки телефона.
- Процесс локализации и интернационализации приложения очень прост.

Платформа Android поддерживает множество типов ресурсов (см. рис. 4.1), с помощью которых могут быть созданы самые разные приложения.
В приложении Android может использоваться множество различных видов ресурсов. Ниже приведен список некоторых из наиболее распространенных типов:

- строки, цвета и размерности;
- графические файлы рисунков;
- файлы макета;
- подключаемые файлы всех типов.

Все ресурсы распределяются по специальным папкам проекта, а тип ресурса определяется XML-тегами. Некоторые папки, такие как **/res/drawable**, **/res/layout** и **/res/values**, создаются автоматически для каждого нового проекта Android, кроме них разработчик может также добавить свои папки.

К файлам ресурсов, хранящимся в папке /res, применяются следующие требования:

- Символы имени файла ресурса должны быть в нижнем регистре.
- Имя файла ресурса может содержать только латинские буквы, числа, символ подчеркивания и точки.
- Имена файлов ресурса (и XML-атрибуты `name`) должны быть уникальными.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

При компилировании ресурсов, имена соответствующих им переменных берутся из имени файла. Например, графическому файлу **mypic.jpg**, сохраненному в папке **/drawable**, соответствует ресурс `@drawable/mypic`. Поэтому будьте внимательны во время присвоения имен файлам ресурсов.

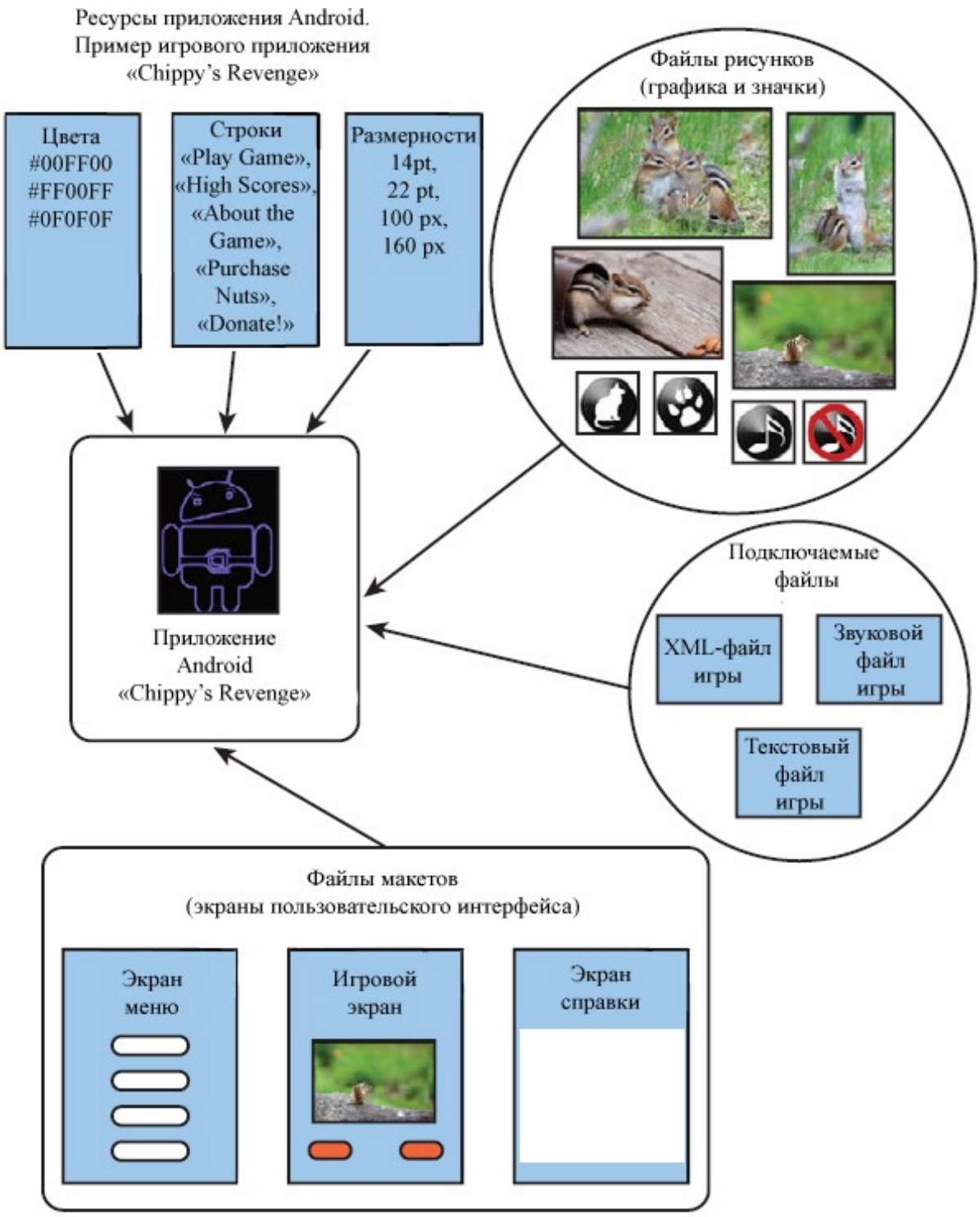

Рис. 4.1. В приложениях Android могут использоваться различные типы ресурсов

Более подробную информацию об именовании папок и файлов проекта вы можете найти в документации по Android.

ОБРАЩЕНИЕ К РЕСУРСАМ ПРИЛОЖЕНИЯ

Все ресурсы приложения сохраняются в папке /res проекта и компилируются в проект во время сборки. Ресурсы приложения могут использоваться программно, и к ним может происходить обращение из других ресурсов приложения.

ВНИМАНИЕ!

Каждый раз, когда вы сохраняете файл ресурса (те. копируйте файл ресурса, например мафический файл, в соответствующую папку) в Eclipse, файл класса `R.java` перекомпилируется для учета изменений. Если имя какого-либо файла не соответствует принятым соглашениям, компилятор сообщит об ошибке на панели Problems (Проблемы) в Eclipse.

К ресурсам приложения можно получить доступ программно через сгенерированный класс `R.iava`. Чтобы обратиться к ресурсу, вы должны получить доступ к объекту приложения `Resources` с помощью метода `getResources ()` и затем сделать вызов этого метода, основанный на типе ресурса, который вы хотите получить.

Например, чтобы получить строковый ресурс `hello`, определенный в файле **strings.xml,** используется следующий вызов метода:

```
String greeting=getResources {) .getString (R. string.hello) ;
```

Более подробно о получении доступа к различным типам ресурсов мы поговорим чуть позже.

Для обращения к ресурсу приложения из другого скомпилированного ресурса, такого как файл макета, используется следующий синтаксис:

```
@тип_ресурса/имя_ресурса
```

Например, к строке, представленной в предыдущем примере, вы можете обратиться следующим образом:

```
@string/hello
```

Более подробно об обращении к ресурсам мы поговорим в этом часе далее, когда речь пойдет о файлах макета.

Работа с системными ресурсами

Кроме ресурсов приложения, разработчики могут получать доступ к системным ресурсам Android. У вас есть доступ к большинству системных ресурсов, распределяемых между несколькими приложениями, что поможет создать общий стиль и восприятие. Например, класс системных строковых ресурсов Android (`android.R.string`) содержит стандартные слова, такие как **OK**, **Cancel** (Отмена), **Yes** (Да), **No** (Нет). **Cut** (Вырезать), **Copy** (Копировать), **Paste** (Вставить).

КСТАТИ

Дли поддержания малого объема вашего приложения и эффективного использования ресурсов перед добавлением новых ресурсов в проект проверяйте системные ресурсы — возможно, нужный вам ресурс, уже существует.

Все системные ресурсы находятся в пакете `android`. В нем содержатся классы для каждого из основных типов ресурсов.

Например, класс `android.R.string` содержит строковые системные ресурсы. Чтобы получить системный строковый ресурс со значением `ok`, например, сначала вы должны воспользоваться статичным методом `getSystem()` класса `Resources`, обеспечивающим доступ к глобальному системному объекту `Resource`, а затем методом `getString()`, указав полное имя строкового ресурса, например:

```
String confirm = Resources.getSystem().
getString(android.R.string.ok);
```

Для обращения к системному ресурсу из скомпилированного ресурса, такого как файл макета, используется следующий синтаксис:

```
@android:[тип_ресурса]/[имя_ресурса]
```

Например, вы можете обратиться к системному строковому ресурсу **ok**, установив соответствующий атрибут следующим образом:

```
@android:string/ok
```

РАБОТА С ПРОСТЫМИ РЕСУРСАМИ

Простые ресурсы, такие как строка, цвет и размерность, должны быть определены в XML-файлах папки **/res/values** проекта. В этих файлах используются специальные XML-теги, представляющие собой пары имя/значение. Такие ресурсы компилируются в пакет приложения во время сборки.

КСТАТИ

Управление ресурсами строк, цвета и размерностей производится в редакторе ресурсов Eclipse, но вы можете отредактировать XMl-файл в любом текстовом редакторе.

Работа со строками

Строковые ресурсы обычно используются тогда, когда в вашем приложении требуется произвести вывод какого-либо текста на экран. Строковые ресурсы обозначаются тегом `<string>` и хранятся в файле ресурсов **/res/values/ strings.xml**.

Ниже приведен пример файла строкового ресурса:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <string name="app_name">Name this App</string>
      <string name="hello">Hello</string>
</resources>
```

К строковым ресурсам можно применять форматирование. В табл. 4.1 представлены некоторые простые примеры строковых значений с различным форматированием.

КСТАТИ

Строковые ресурсы, содержащие апострофы (одинарные прямые кавычки), должны быть заключены в двойные прямые кавычки.

Таблица 4.1

Примеры форматирования строковых ресурсов

Значение строкового ресурса	Результат
Привет, мир	Привет, мир
«Привет, мир»	Привет, мир
\'Привет, мир\'	'Привет, мир'
Он сказал \»Нет\»	Он сказал «Нет»

Существует несколько способов доступа к строковому ресурсу на программном уровне. Самый простой заключается в использовании метода **getstring();**

```
String greeting = getResources().getString(R.string.hello);
```

Работа с цветами

Вы можете применять цветовые ресурсы к различным элементам пользовательского интерфейса. Цветовые ресурсы обозначаются тегом `<color>` в файле **/res/values/co(ors.xml**. Этот XML-файл ресурса не создается автоматически, поэтому при необходимости должен быть создан вручную.

КСТАТИ

Вы можете добавить новый XML-файл, такой как этот, в специальном диалоговом окне, которое открывается командой **File => New => Android XML File** (Файл => Новый => XML-файл Android*)*. Таким образом, предполагаемая папка и тип файла ресурса для проекта Android, будут заданы автоматически.

Ниже приведен пример цветового файла ресурса:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <color name="background_color">#006400</color>
      <color name="app_text_color">#FFE4C4</color>
</resources>
```

Система Android поддерживает 12-разрядные и 24-разрядные цвета в цветовой модели RGB. В табл. 4.2 представлен список форматов, поддерживаемых платформой Android.

Таблица 4.2

Цветовые форматы, поддерживаемые платформой Android

Формат	Описание	Пример
#RGB	12-разрядный цвет	#00F (синий)
#ARGB	12-разрядный цвет с поддержкой	#800F (синий, прозрачность

	прозрачности	50%)
#RRGGBB	24-разрядный цвет	#FF00FF (пурпурный)
#AARRGGBB	24-разрядный цвет с поддержкой прозрачности	#80FF00FF (пурпурный, прозрачность 50%)

В следующем фрагменте кода происходит получение цветового ресурса `app_text_color` методом `getColor()`:

```
int textColor = getResources().getColor(R.color.app_text_color);
```

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Существует множество способов подбора цвета. Например, на сайте **html-color-codes.info** вы найдете простую таблицу цветов, щелкая по ячейкам которой вы можете выбрать подходящий цвет.

Работа с размерностями

Размер элементов пользовательского интерфейса, таких как кнопки или элемент `Textview`, определяется различными видами размерностей. Ресурсы размерности обозначаются тегом `<dimen>` в файле ресурса **/res/values/ dimens.xml.** Этот XML-файл не создается автоматически и должен быть создан вручную.

Пример файла ресурса размерности:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
     <dimen name="thumbDim">100px</dimen>
</resources>
```

Для каждого значения ресурса размерности должна быть указана единица измерения. В табл. 4.3 перечислены единицы измерения, поддерживаемые платформой Android.

Таблица 4.3

Единицы измерения размерностей, поддерживаемые Android

Единица измерения	Описание	Условное обозначение
Пиксели	Экранные пиксели	px
Дюймы	Физическая величина	in
Миллиметры	Физическая величина	mm
Точки	Применяется в основном для измерения шрифтов	pt
Пиксели (независимо от плотности)	Пиксели относительно плотности 160 dpi	dp
Пиксели (независимо от масштаба)	Применяются для масштабирования шрифтов	sp

В следующем фрагменте кода происходит получение ресурса размерности методом getDimension():

```
float thumbnailDim = getResources().getDimension(R.dimen.thumbDim);
```

РАБОТА С РЕСУРСАМИ РИСУНКОВ

Ресурсы рисунков, такие как файлы изображений, хранятся в папке **/res/ drawable** проекта. Во время сборки эти ресурсы будут скомпилированы в пакет приложения, чтобы быть доступными приложению.

КСТАТИ

Вы можете перетаскивать файлы изображений непосредственно в папку **/res/ drawable** через панель **Project Explorer** (Проводник проектов) в Eclipse. Не забывайте, что имена файлов должны быть в нижнем регистре и должны содержать только латинские буквы, числа и символы подчеркивания.

Работа с изображениями

Наиболее распространенные графические ресурсы, используемые в приложениях, это растровые изображения, такие как PNG и JPG-файлы. Эти файлы часто используются в качестве значков приложений и кнопок, а также элементов пользовательского интерфейса.

Как вы можете видеть в табл. 4.4, Android поддерживает несколько распространенных форматов изображения.

Таблица 4.4

Графические форматы изображения, поддерживаемые платформой Android

Формат изображения	Описание	Расширение файла
PNG	Рекомендуемый формат (без потерь)	**.png**
PNG с поддержкой растягивания NinePatch	Рекомендуемый формат (без потерь)	**.9.png**
JPEG	Допустимый формат (с потерями)	**.jpg**
GIF	Не рекомендуется к использованию	**.gif**

ИСПОЛЬЗОВАНИЕ РЕСУРСОВ ИЗОБРАЖЕНИЯ В ПРОГРАММНОМ КОДЕ

Ресурсы изображений содержатся в классе `BitmapDrawable`. Чтобы получить доступ к графическому файлу ресурса, например **/res/drawable/logo.png,** используйте метод `getDrawable()`:

```
BitmapDrawable logoBitmap =
(BitmapDrawable)getResources().getDrawable(R.drawable.logo);
```

Однако в большинстве случаев нет необходимости загружать графику напрямую. Вместо этого вы можете использовать идентификатор ресурса в качестве атрибута элемента пользовательского интерфейса, например `ImageView`. В следующем фрагменте кода файл **logo.png** устанавливается и загружается в элемент интерфейса `ImageView` с именем `LogolmageView`, который должен быть определен в макете:

```
ImageView logoView = (ImageView)findViewById(R.id.LogoImageView);
logoView.setImageResource(R.drawable.logo);
```

Работа с другими типами рисунков

В дополнение к графическим файлам вы можете также создать специальные XML-файлы для описания других подклассов `Drawable`, таких как `ShapeDrawable`. Вы можете использовать класс `ShapeDrawable` для описания различных геометрических фигур, таких как прямоугольники и овалы. Более подробную информацию вы можете найти в документации Android, пакет `android.graphics.drawable`.

РАБОТА С МАКЕТАМИ

Большинство элементов пользовательского интерфейса приложений Android описывается специальными XML-файлами, называемыми файлами макета. Файлы ресурсов макетов находятся в папке **/res/layout.**

Также, как и другие ресурсы, файлы макета компилируются вместе с приложением.

ЗНАЕТЕ ЛИ ВЫ, ЧТО…
Очень часто файлы макета описывают весь экран и ассоциируются с определенной деятельностью, но это не единственный верный подход. Ресурсы макетов могут также описывать часть экрана и быть составной частью другого макета.

Далее приведен пример фала ресурса макета:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
        android:text="@string/hello" />
</LinearLayout>
```

Возможно, вы уже узнали код этого файла — это файл макета **main.xml** по умолчанию, который автоматически создается для каждого нового приложения Android. Этот файл макета описывает пользовательский интерфейс деятельности в пределах приложения. Он содержит элемент `LinearLayout`, используемый в качестве контейнера для всех элементов пользовательского интерфейса — в нашем случае элемента `TextView`.

В этом примере файл макета main.xml содержит ссылку на другой ресурс — строковый — `@string/hello`, который определен в файле ресурсов **strings.xml**.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Макеты могут также создаваться, редактироваться и использоваться во время работы приложения. Однако в большинстве случаев XML-файлы макетов значительно упрощают код и его использование.

Существует два способа форматирования ресурсов макета. Самый простой способ заключается в использовании редактора ресурсов Eclipse для разработки и предварительного просмотра файлов макета. Вы также можете редактировать XML-файлы макета в любом текстовом редакторе.

Проектирование макетов в редакторе ресурсов

Вы можете проектировать и просматривать макет в Eclipse с помощью редактора ресурсов макета (см. рис. 4.2). Если выбрать файл проекта **/res/ layout/main.xml** в Eclipse, вы увидите панель **Layout** (Макет), на которой в режиме предварительного просмотра отображается интерфейс, сформированный этим макетом. Добавлять и удалять элементы макета вы можете на панели **Outline** (Структура). Свойства и атрибуты для каждого элемента задаются на панели **Properties** (Свойства).

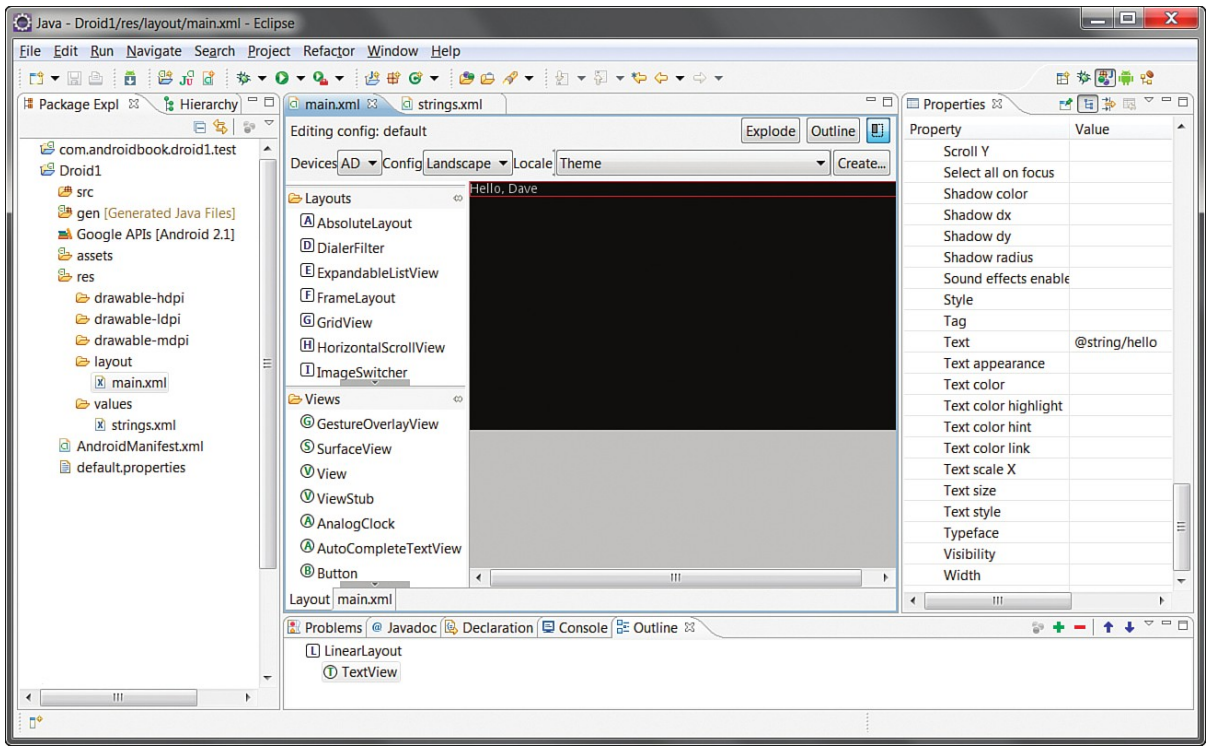

Рис. 4.2. Макет в редакторе ресурсов Eclipse

Как большинство других инструментов проектировки интерфейса, редактор ресурсов макета подходит для редактирования простого макета. Однако редактор ресурсов макета поддерживает элементы пользовательского интерфейса не полностью. Для создания более сложных элементов придется редактировать XML-файл вручную. После добавления новых элементов интерфейса в макет вы можете потерять возможность предварительного просмотра. Однако вы по-прежнему можете видеть макет интерфейса в эмуляторе или на мобильном телефоне.

Помните, что одна из главных задач разработчика — корректное отображение приложения на мобильном телефоне, а не в редакторе макета Eclipse.

Проектировка макетов в коде XML

Вы можете редактировать код необработанного XML-файла макета вручную. Если вы выберете файл **/res/layout/main.xml** и перейдете на вкладку **main.xml**, то увидите XML-код файла макета.

КСТАТИ

По мере приобретения опыта в разработке макетов, вы должны лучше ознакомится с принципами составления XML-файлов. Время от времени переходите на вкладку редактирования кода XML и обращайте внимание на код, сгенерированный для каждого типа элемента пользовательского интерфейса. Не полагайтесь на один только редактор ресурсов макета Eclipse.

Выполните самостоятельно

Ознакомьтесь с редактором ресурсов макета Eclipse:

1. Откройте проект Droid 1, созданный вами ранее.

2. Найдите файл макета **/res/layout/main.xml** и откройте его в редакторе ресурсов макета Eclipse двойным нажатием.
3. Переключитесь на вкладку Layout (Макет), на ней вы увидите макет главного экрана в режиме предварительного просмотра.
4. Перейдите на панель **Outline** (Структура). Здесь представлена иерархия XML-файла макета. Сейчас вы можете видеть элемент интерфейса LinearLayout, который содержит *в* себе элемент интерфейса TextView.
5. Выберите элемент интерфейса TextView на панели **Outline** (Структура). Теперь этот элемент интерфейса в предварительном просмотре макета выделен цветом.
6. Перейдите на вкладку **Properties** (Свойства). Здесь приведены все свойства и атрибуты элемента пользовательского интерфейса TextView, которые вы можете конфигурировать. Найдите в этом списке свойство Text и обратите внимание, что его значением является строковый ресурс `@string/hello`.
7. Щелкните по значению `@string/hello` этого свойства на панели **Properties** (Свойства), чтобы изменить его. Каждый раз, когда вы меняете значение, обращайте также внимание на изменения в предварительном просмотре макета. Вы можете ввести строку непосредственно, указать другой строковый ресурс (например, `@string/app_name`) или, щелкнув по кнопке с многоточием, выбрать подходящий ресурс из списка доступных для вашего приложения.
8. Перейдите на вкладку main.xml и обратите внимание на XML-код, Если вы сохраните и запустите свой проект в эмуляторе, то все изменения, которые были видны в предварительном просмотре, будут применены в эмуляторе.

Не бойтесь экспериментировать с Eclipse. Вы можете добавить несколько новых элементов интерфейса, таких как ImageView или еще один элемент TextView в ваш макет. Проектировку макетов мы рассмотрим более детально в этой книге позже.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Наиболее важные панели в редакторе ресурсов макета — **Outline** (Структура) и **Properties** (Свойства). Если разместить панель **Properties** (Свойства) в верхнем правом углу перспективы **Java**, то вы можете одновременно видеть основной редактор ресурсов макета в середине экрана, панель **Project Explorer** (Проводник проекта) слева, панель **Properties** (Свойства) справа (чтобы полностью использовать вертикальное пространство), а панель **Outline** (Структура) — внизу (на ней обычно меньше информации, чем на панели **Properties** (Свойства)).

ИСПОЛЬЗОВАНИЕ РЕСУРСОВ МАКЕТА В ПРОГРАММНОМ КОДЕ

Для доступа к макету программно вы можете воспользоваться классом `LayoutInflater`, чтобы расширить файл макета к объекту `View`, как в еле дующем примере:

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.main, null);
```

В большинстве случаев нет необходимости загружать весь макет, так как нам нужны лишь отдельные элементы, такие как `TextView` в файле макета main.xml. Содержание макета, независимо от того, какой это элемент интерфейса (`Button`, `ImageView` или `TextView`), происходит из класса `View`.

Файл макета по умолчанию, созданный для проекта Droidl, содержит элемент интерфейса `TextView`. Однако элемент интерфейса `TextView` по умолчанию не имеет атрибута name. Самый простой способ получить доступ к необходимому элементу интерфейса `View` — по его имени, полому вам необходимо задать атрибут id элементу интерфейса `TextView` в редакторе ресурсов макета. Назовите его `@+id/TextView01`.

Ниже приведен пример доступа к объекту `TextView` с именем `TextView01`, который был определен в файле ресурса макета:

```
TextView txt = (TextView)findViewById(R.id.TextView01);
```

РАБОТА С ФАЙЛАМИ

Кроме ресурсов строк, графических ресурсов и ресурсов макетов, в проектах Android могут содержаться файловые ресурсы. Такие файлы могут быть в любом формате. Тем не менее некоторые форматы более удобны и используются чаще.

Работа с XML-файлами

Формат XML полностью поддерживается платформой Android. Любые XML-файлы могут быть включены в качестве ресурсов, именно их рекомендуется использовать для хранения упорядоченных данных.

Структура XML-файла может быть любой — на ваше усмотрение. Множество утилит XML (табл. 4.5) входят в состав платформы Android.

Таблица 4.5

Пакеты утилит XML

Пакет	Описание
android.sax*	Средство для написания стандартных обработчиков SAX
android.util.Xml.*	Утилиты XML, включая XMLPullParser
org.xml.sax.*	Основные возможности SAX (см. **www.saxproject.org**)
javax.xml.*	SAX и ограничения DOM с поддержкой ядра 2-го уровня
org.w3c.dom	Интерфейсы для DOM, ядро 2-го уровня
org.xmlpull.*	Интерфейсы XmlPullParser и XMLSerializer (см. **www.xmlpull.org**)

Для доступа к XML-файлу /res/xml/default_values.xml программно вы можете воспользоваться методом `getXml()`:

```
XmlResourceParser defaultDataConfig = getResources().getXml(R.xml.default_values);
```

Работа с подключаемыми файлами

В приложениях Android в качестве ресурсов могут использоваться подключаемые необработанные файлы (raw files). Подключаемыми могут быть аудио- и видеофайлы, а также множество других форматов. Все подключаемые файлы ресурсов должны содержаться в папке **/res/raw**.

Не существует никаких правил или ограничений на содержание подключаемых файлов (кроме имен файлов ресурсов, о которых говорилось ранее). Но перед интегрированием каких-либо медиафайлов в проект обратитесь к документации по платформе Android, чтобы узнать какие именно форматы и колировки поддерживаются на *mex* мобильных телефонах, где вы планируете распространять ваше приложение. То же самое касается любого другого формата файла, который вы будете использовать в качестве ресурса приложения. Если формат файла, который вы планируете использовать, не поддерживается системой Android, то ваше приложение должно производить всю работу по обработке файлов собственными силами.

Для доступа к подключаемым файлам ресурсов программно можете воспользоваться методом `openRawResource()`. Например, в следующем фрагменте кода создается объект inputstream для доступа к файлу ресурса `/res/raw/file1.txt`:

```
InputStream iFile = getResources().openRawResource(R.raw.file1);
```

КСТАТИ

Каждому подключаемому файлу ресурса должно быть присвоено уникальное имя, исключая суффикс имени файла, например одновременное использование имен файлов **file1.txt** и **file1.dat** недопустимо.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Бывают случаи, когда вам может быть необходимо использовать файлы в приложении, но при этом не компилировать их вместе с остальными ресурсами. Для этой цели в Android предусмотрена специальная папка проекта — **/assets.** Эта папка находится в той же самой папке, что и **/res.** Все файлы, хранящиеся в этой папке, представлены в бинарном виде, с пакетом установки приложения, и они не компилируются.
Некомпилированые файлы, называемые активами приложения, не доступны методом `getResources()`. В этом случае для доступа к файлам а папке **/assets** используется AssetManager.

РАБОТА С ДРУГИМИ ТИПАМИ РЕСУРСОВ

Мы рассмотрели все наиболее часто используемые типы ресурсов, которые нам могут понадобиться в приложении. Но, кроме рассмотренных, существует еше несколько типов

ресурсов. Эти типы используются редко, и их применение на практике может вызвать затруднения. Они рассчитаны на очень мощные приложения. Ниже перечислены некоторые из таких типов ресурсов:

- строковые массивы;
- меню;
- анимации;
- геометрические фигуры;
- стили и темы;
- собственные элементы пользовательского интерфейса.

Если вы все же планируете использовать эти типы ресурсов, ознакомьтесь с информацией по данной теме в документации Android.

ИТОГИ

В приложениях Android может использоваться множество различных типов ресурсов, включая узкоспециальные типы ресурсов и системные ресурсы. Редактор ресурсов Eclipse упрощает управление ресурсами, но код XML- файлов ресурсов должен редактироваться вручную. После определения к ресурсам можно получить доступ непосредственно или по имени из других ресурсов. Строковые ресурсы, ресурсы цвета и размерности хранятся в специальных XML-файлах, а графические изображения хранятся в виде отдельных файлов. Пользовательские интерфейсы определяются XML- файлами макетов. Подключаемые файлы, содержащие данные в различных форматах, также могут использованы в качестве ресурсов. Кроме этого, приложения могут включать нескомпилированные бинарные файлы, называемые активами приложения.

ВОПРОСЫ И ОТВЕТЫ

Вопрос. Могу ли я узнать, что представляет из себя определенный системный ресурс по его имени?

Ответ. Не всегда. В официальной документации Android нет подробного описания всех системных ресурсов. Если вы не уверены в том, что представляет из себя тот или иной системный ресурс или для чего он служит, можете поэкспериментировать с ним самостоятельно и попробовать выяснить его значение. Описание системных ресурсов Android можно найти в папке **/toots/lib/res/default**.

Вопрос. Должны ли строковые ресурсы, ресурсы цвета и размерности хранится в отдельных XML-файлах?

Ответ. Технически - нет. Однако это настоятельно рекомендуется. Например, строковые ресурсы для различных локалей и языков будут отличаться, но ресурсы размерностей останутся теми же самыми. Поэтому распределение ресурсов по типам позволяет организовывать более гибкую структуру.

Вопрос. Какой XML-парсер лучше всего использовать?

Ответ. Наши тесты показали, что парсер SAX — самый эффективный синтаксический анализатор XML (сопровождаемый XMLPullParser), и в большинстве случаев мы рекомендуем именно его. Однако выбор всегда остается за вами, и вы должны сами выбрать лучший парсер исходя из особенностей вашего приложения.

Вопрос. В чем различие между активами приложения и ресурсами проекта?

Ответ. Ресурсы проекта компилируются в приложение и могут быть доступны метолом `getResources()`. Активы приложения используются менее часто для хранения скомпилированных файлов в файле пакета приложения, который устанавливается на мобильный телефон. Доступ к активам
выполняется методом `getAssets()`.

ПРАКТИКУМ
Контрольные вопросы

1. Какую разрядность цвета поддерживают цветовые ресурсы?
   A. 12-разрядный цвет.
   B. 24-разрядный цвет.
   C. 64-разрядный цвет.

2. Вы можете использовать в качестве ресурсов файлы любого формата. Правда ли это?

3. Какие графические форматы поддерживаются и рекомендуются к использованию в Android?
   A. JPG.
   B. PNG.
   C. GIF.
   D. PNG с поддержкой NinePatch-растягивания.

4. Символы имен файлов ресурсов могут быть в верхнем регистре. Правда ли это?

5. Имя ресурса может быть произвольным. Правда ли это?

Ответы

Правильные ответы: А и Б. Поддерживается как 12-разрялные, так и 24-разрядные цветовые форматы.

Правда. Для этого вам необходимо интегрировать файл ресурса в качестве подключаемого.

Правильные ответы: Б и Г. Все четыре формата поддерживаются, но предпочтительнее использовать графические PNG-файлы, включая файлы с поддержкой NinePatch-растягивания, потому что сжатие информации в них происходит без потерь. Внутренняя компрессия в файлах JPG происходит с потерями, а формат GIF считается устаревшим.

Нет, это не так. Имена файлов ресурсов могут содержать буквы, числа, символы подчеркивания и обязательно должны быть в нижнем регистре.

Нет, это не так. Имена ресурсов влияют на имена переменных в коде, используемых для ссылки на ресурсы.

Упражнения

Добавьте новый цветовой ресурс со значением `#00ff00` к вашему проекту Droid 1. В файле макета измените значение атрибута `TextColor` элемента интерфейса `TextView` на значение нового цветового ресурса. Перезапустите приложение и обратите внимание на изменения.

Добавьте новый ресурс размерности со значением `22pt` в проект Droidl. В файле макета измените значение атрибута `textSize` элемента интерфейса `TextView` на значение нового ресурса размерности. Перезапустите приложение и обратите внимание на изменения.

Добавьте новый файл графического ресурса в проект Droidl. В файле макета добавьте новый элемент интерфейса `ImageView` и измените значение атрибута `src`, чтобы он соответствовал новому графическому ресурсу. Перезапустите приложение и обратите внимание на изменения.

Добавьте ресурс подключаемого текстового файла в проект Droidl. Методом `openRawResource()` создайте объект `InputStream` и выполните прочтение файла. Выведите содержание файла в журнал методом `Log.v()`. Перезапустите приложение и обратите внимание на изменения.

Час 5. КОНФИГУРИРОВАНИЕ ФАЙЛА МАНИФЕСТА ANDROID

Вопросы, рассматриваемые в этом часе:

- **структура файла манифеста Android;**
- **конфигурирование основных настроек приложения;**
- **определение деятельности;**
- **управление правами приложения;**
- **управление другими настройками приложения.**

Каждый проект Android должен содержать специальный файл, называемый файлом манифеста. Из этого файла система получает настройки конфигурации приложения, включая идентификационную информацию, а также права, требуемые приложению для нормальной работы. В этом часе мы рассмотрим файл манифеста Android более подробно, и вы узнаете, как он используется в различных приложениях.

ИССЛЕДОВАНИЕ ФАЙЛА МАНИФЕСТА ANDROID

Файл манифеста Android — **AndroidManifest.xml —** один из важнейших файлов любого проекта Android. В системе Android этот файл служит для нескольких целей:

- установки и обновления пакетов приложений;
- предоставления информации о приложении пользователю;
- запуска деятельностей приложения;
- управления правами приложения;
- обработки множества других дополнительных конфигураций приложения, включая работу в качестве провайдера услуг или контент-провайдера.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Если вы пользуйтесь Eclipse с плагином ADT, то мастер проектов Android автоматически создавать готовый файл **AndroidManifest.xml** со значениями по умолчанию для наиболее важных настроек конфигурации.

Вы можете редактирован, файл манифеста Android и редакторе ресурсов Eclipse или в любом другом текстовом редакторе.

Редактор ресурсов Eclipse распределяет данные файла манифеста по нескольким категориям, представленным на пяти вкладках:

- **Manifest** (Манифест);
- **Application** (Приложение);
- **Permissions** (Права);
- **Instrumentation** (Инструментарий);
- **AndroidManifest.xml.**

Вкладка Manifest (Манифест)

На вкладке **Manifest** (Манифест) (см. рис. 5.1) содержатся настройки всего пакета, включая имя пакета, информацию о версии приложения и минимально допустимой версии Android SDK. Здесь вы можете также задать аппаратные требования.

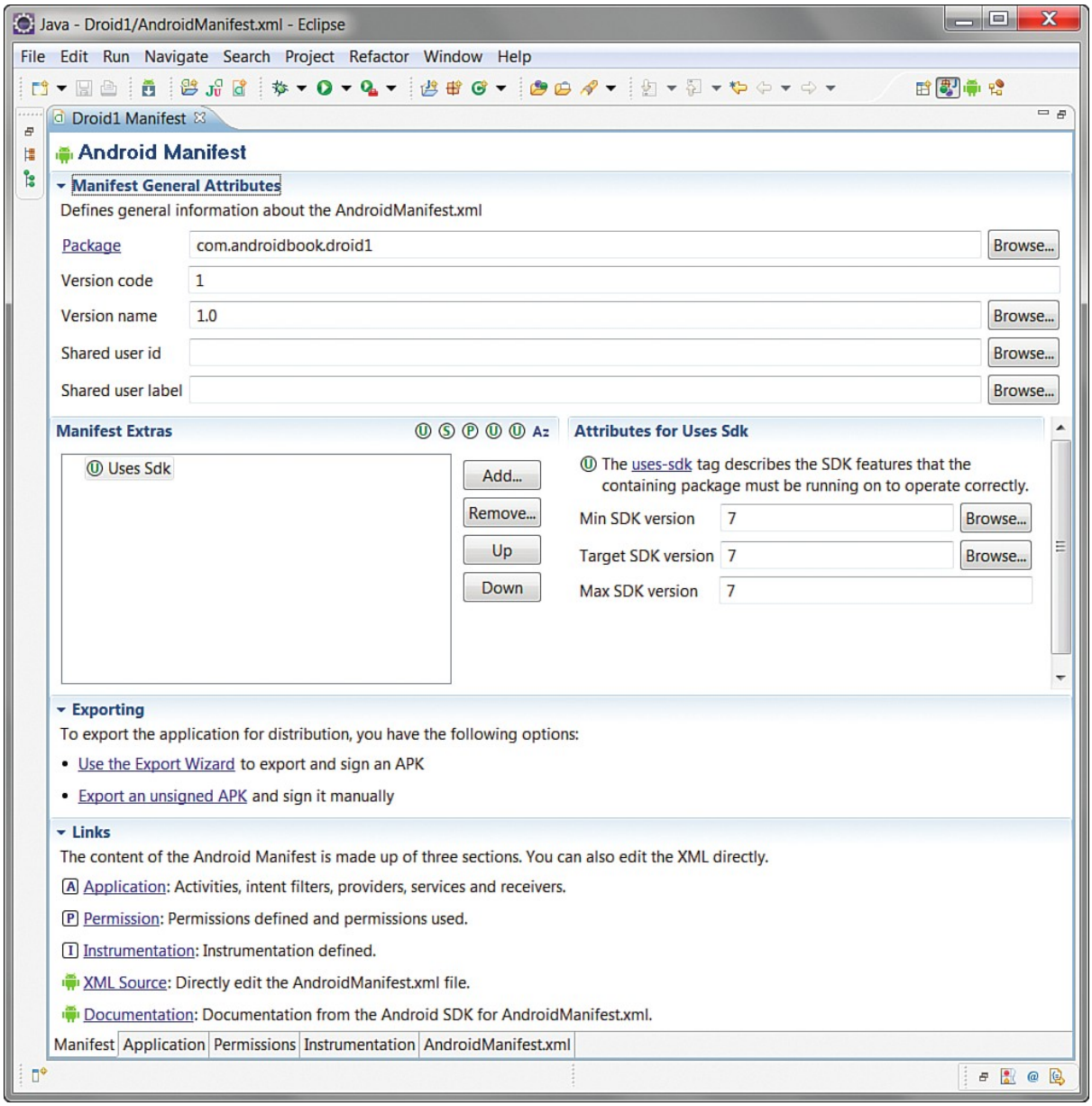

Рис. 5.1. Вкладка **Manifest** (Манифест) в редакторе Eclipse

Вкладка Application (Приложение)

На вкладках Application (Приложение) (см. рис. 5.2) содержатся настройки всего приложения, включая метку приложения и значок, а также информацию о компонентах приложения, таких как деятельности, интент-фильтры, и другой функциональности приложения, включая конфигурацию провайдера услуг и контент-провайдера.

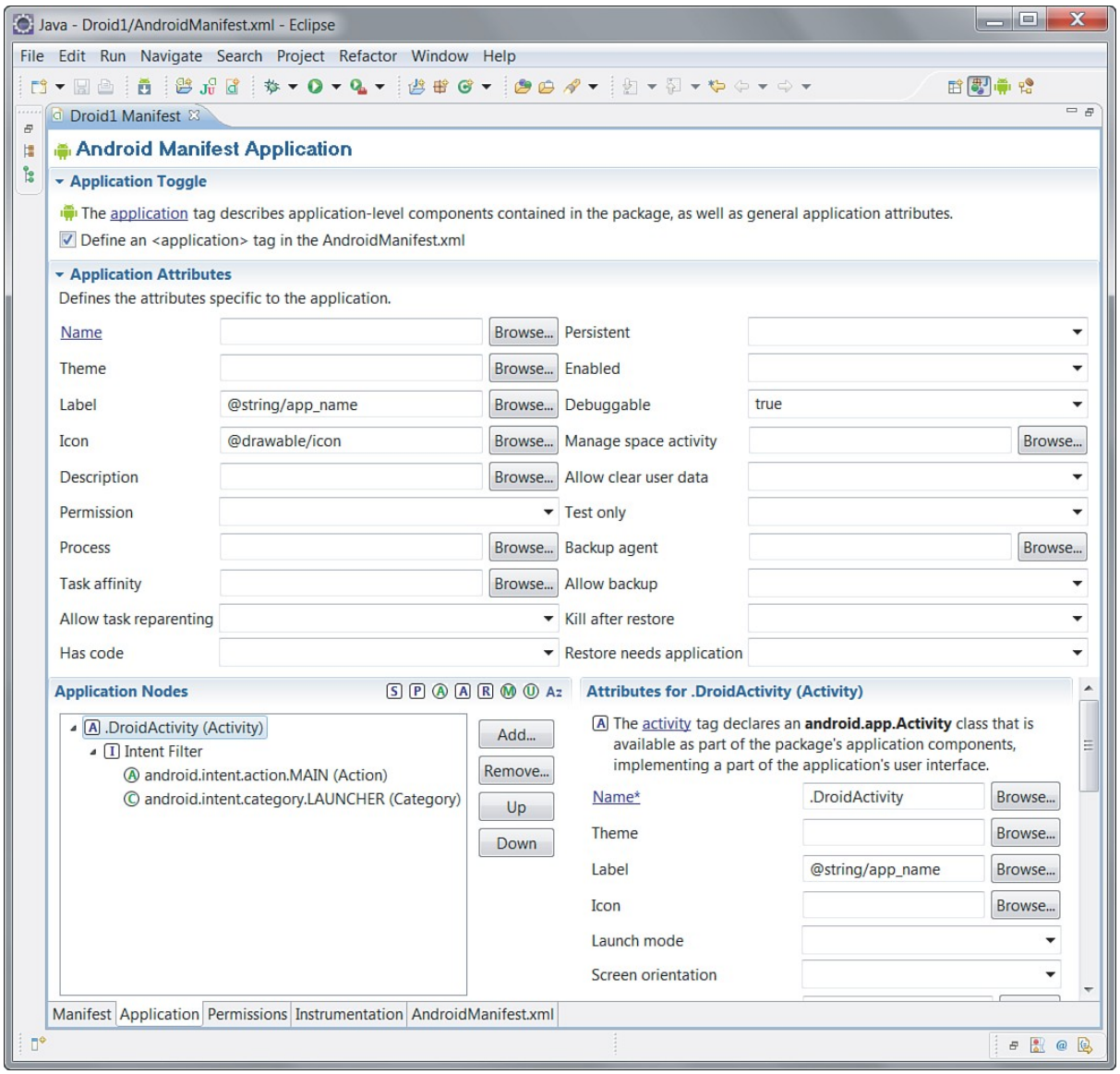

Рис. 5.2. Вкладка **Application** (Приложение) в редакторе ресурсов Eclipse

Вкладка Permissions (Права)

На вкладке Permissions (Права) (см. рис. 5.3) описаны права, предоставляемые приложению. Здесь могут быть также указаны специальные права, которые требуются приложению для работы.

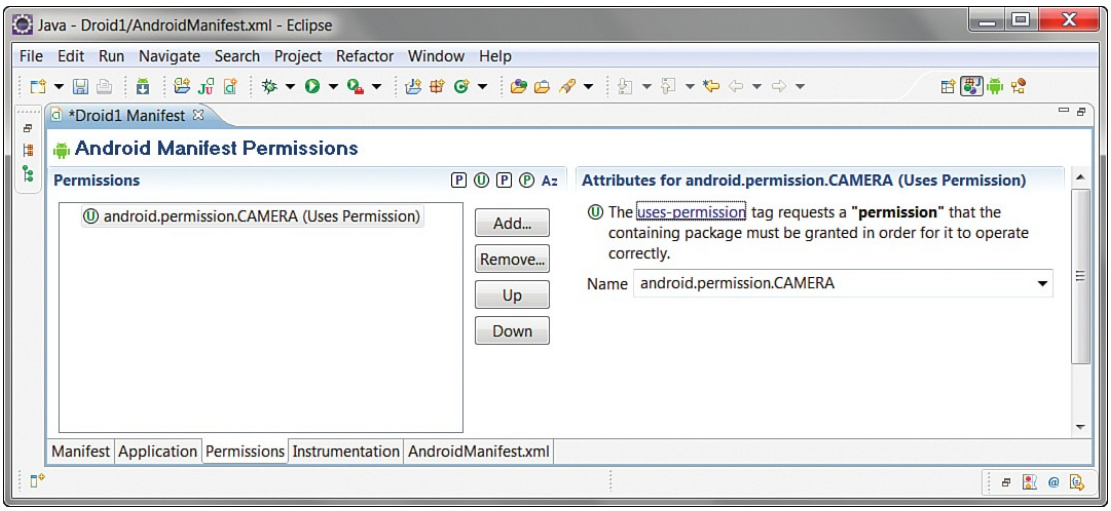

Рис. 5.3. Вкладка **Permissions** (Права) в редакторе Eclipse

ВНИМАНИЕ!

Не путайте раскрывающийся список **Permission** (Права! вкладки **Application** -Приложение) с возможностями вкладки **Permissions** (Права). На вкладке **Permissions** (Права) указываются права, требуемые приложению для доступа к необходимым ресурсам или API. В раскрывающемся списке **Permission (**Права) определяются права, требуемые другими приложениями для доступа к открытым ресурсам и API вашего приложения.

Вкладка Instrumentation (Инструментарий)

На вкладке **Instrumentation** (Инструментарий) (см. рис. 5.4) указываются классы инструментария для мониторинга за приложением. В поле **Name** (Имя) вводится полное имя класса подкласса instrumentation вашего приложения, а в поле **Target Package** (Целевой пакет) вы указываете имя пакета, файл которого содержит <application> для контролируемого приложения. Более подробно об инструментарии и тестировании мы поговорим в следующем часе.

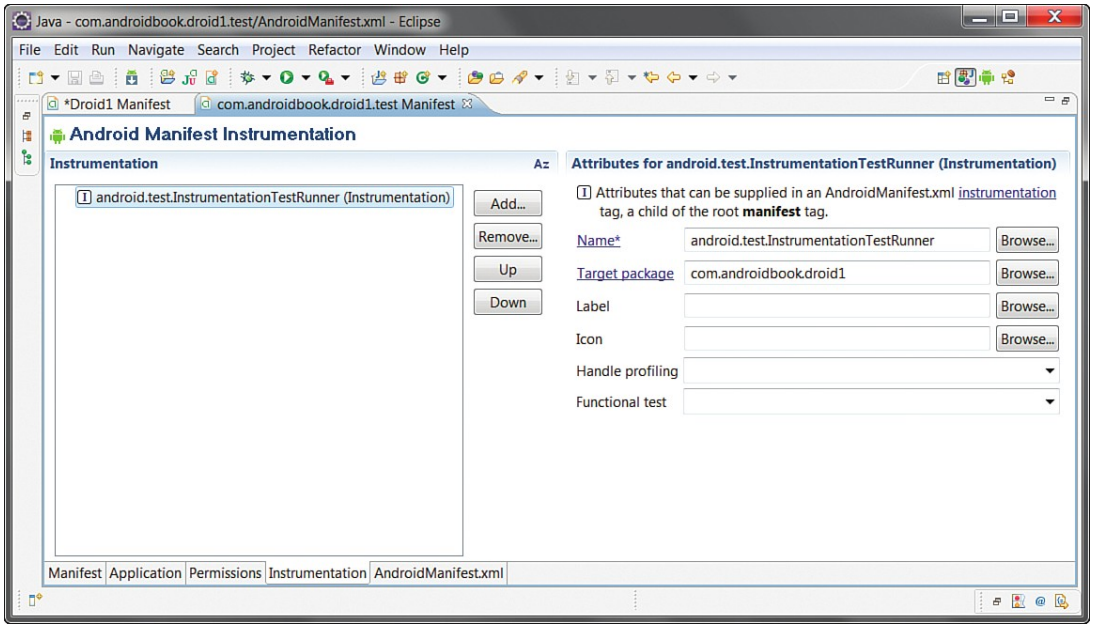

Рис. 5.4. Вкладка **Instrumentation** (Инструментарий) в редакторе Eclipse

Вкладка AndroidManifest.xml

Файл манифеста Android представляет собой XML-файл. Вы можете редактировать код XML вручную на вкладке AndroidManifest.xml редактора ресурсов (см. рис. 5.5).

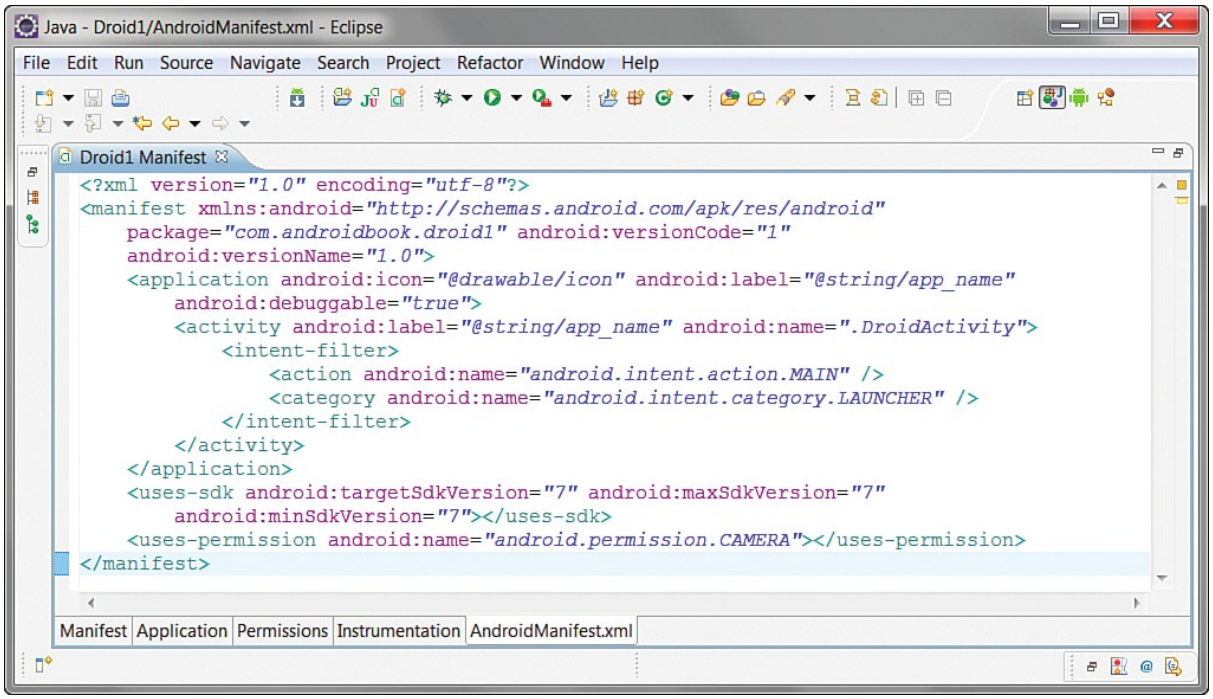

Рис. 5.5. Вкладка **AndroidManifest.xml** редактора ресурсов Eclipse

На рис. 5.5 приведен код файла манифеста проекта Droidl, созданного вами в первом часе. Пока этот XML-код довольно прост.

Обратите внимание на то, что в файле содержится единственный тег `<manifest>`, в котором содержатся настройки для всего пакета. В этом теге содержится другой тег — `<application>`, который служит для идентификации приложения, с единственной деятельностью `DroidActivity` и интент-фильтром. Кроме этого здесь присутствует тег `<uses-sdk>`, который содержит три атрибута версий.

Теперь давайте поговорим о каждой из этих настроек более детально.

КОНФИГУРИРОВАНИЕ ОСНОВНЫХ НАСТРОЕК ПРИЛОЖЕНИЯ

Большинство самых важных настроек приложения задаются атрибутами и дочерними тегами блоков `<manifest>` и `<application>`.

Пакет определяется в теге <manifest> файла манифеста Android с помощью атрибута package, например:

```
<manifest
    xmlns:android=”http://schemas.android.com/apk/res/android”
    package=”com.androidbook.droid1”
    android:versionCode=”1”
    android:versionName=”1.0”>
```

КСТАТИ

Если для создания проектов вы пользуйтесь мастером проектов Android, то вы должны указывать имя пакета. Это имя будет использоваться в файле манифеста Android по умолчанию.

Подключение вспомогательных библиотек

С помощью тега `<uses-library>` приложения могут взаимодействовать с другими библиотеками в дополнение к стандартным пакетам Android. Эта возможность доступна на вкладке **Application** (Приложение) редактора ресурсов. Например:

```
<uses-library
    android:name="com.company.sharedutilities" />
```

Управление версиями приложений

Информация о версии манифеста используется в двух случаях:

- для организации и отслеживания возможностей приложения;
- управления обновлениями приложения.

По этой причине у тега <manifest> есть два отдельных атрибута версии: имя версии и код версии.

ОПРЕДЕЛЕНИЕ ИМЕНИ ВЕРСИИ

Имя версии представляет собой информацию о версии, используемой для отслеживания сборок приложения. Продуманное управление версиями очень важно для публикации приложения и последующей поддержки. Атрибут `android:versionName` тега `<manifest>` хранит строковое значение для отслеживания номера сборки приложения. Например, проекту Droid 1 присвоено имя версии 1.0. Формат имени версии определяется разработчиком. Учтите, что это поле могут видеть пользователи.

УСТАНОВКА КОДА ВЕРСИИ

Код версии позволяет платформе Android программно обновлять приложения. Значение атрибута `android:versionCode` тега `<manifest>` может быть целое число, используемое платформой Android и рынками распространения для управления обновлением приложения. Значение атрибута `android:versionCode` обычно начинается с 1. Это значение должно постепенно увеличиваться с каждой новой версией приложения, готовой для опубликования. Поле кода версии невидимо для пользователей и не должно синхронизироваться с именем версии.

КСТАТИ

Код версии должен увеличиваться только для опубликованных приложений или в целях тестирования. Не изменяйте его каждый раз. когда вы развертываете приложение на устройстве для отладки.

Установка минимальной версии Android SDK

Приложения Android могут быть скомпилированы для совместимости с несколькими различными версиями SDK. В теге `<uses-sdk>` указывается минимально допустимая версия Android SDK для обеспечения нормальной работы приложения на мобильном телефоне. Атрибут `android:minSdkVersion` этого тега представляет минимально допустимую версию SDK в виде целого числа. В табл. 5.1 представлен список доступных версий.

Таблица 5.1

Доступные версии Android SDK

Версия Android sdk	Значение
Android 1.0 SDK 1	1
Android 1.1 SDK 2	2
Android 1.5 SDK 3	3
Android 1.6 SDK 4	4
Android 2.0 SDK 5	5
Android 2.0.1 SDK 6	6
Android 2.1 SDK 7	7

Например, проекте Droid1 вы уже указывали минимальную версию для Android SDK 2.1:

```
<uses-sdk
    android:minSdkVersion="7" />
```

ВНИМАНИЕ!
Номер версии для новых релизов Android SDK вы можете найти в сопроводительных комментариях. Во многих инструментах разработки этот номер часто упоминается как API Level (Уровень API), особенно в Android SDK and AVD Manager.

Именование приложения

Атрибут `android:label` тега `<application>` хранит строку, представляющую имя приложения. Вы можете определить это имя в виде фиксированной строки, как в следующем примере:

```
<application
android:label="My application name">
```

Как вариант, в атрибуте android: label вы можете укатать ссылку на строковый ресурс. В проекте Droid 1 имени приложения соответствует строковый ресурс:

```
<application
android:label="@string/app_name">
```

В этом случае, имя приложения будет хранить строковый ресурс **app_name** в файле **strings.xml.**

Присвоение значка приложению

Атрибут `android:icon` тега `<application>` хранит ссылку на строковый ресурс, представляющий собой значок приложения. В проекте Droidl вы можете задать значку приложения графический ресурс следующим образом:

```
<application
android:icon="@drawable/icon">
```

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО

Несмотря на то, что вы поместили единственную ссылку на значок в атрибуте `android:icon` тега `<application>`, вам необходимо создать три различных значка. Каждый из них предназначен для экрана с определенной плотностью распределения пикселов относительно физических размеров экрана: для экранов с низкой плотностью (1dpi). для экранов с высокой плотностью (hdpi) и экранов со средней плотностью (mdpi) (в версии Android 2.2 введено новое значение плотности — xhdpi, т.е. сверхвысокая плотность. — Прим. ред.). Система Android будет автоматически выбирать наиболее подходящее изображение исходя из характеристик устройства, на котором запущено приложение. Более подробную информацию по этой теме вы узнаете в часе 20. Сейчас вы можете просто создать одно графическое изображение и изменить его размеры для трех различных значений плотности распределения пикселей.

Чтобы создать собственный значок приложения, выполните следующие шаги:

1. Создайте изображение размером 48 x 48 точек в любом графическом редакторе, который вам нравится. Это изображение будет значком для экранов со средней плотностью.
2. Сохраните изображение в формате PNG с именем **myicon.png**.
3. Добавьте этот файл в папку **/res/drawable-mdpi** в качестве графического ресурса вашего приложения.
4. Повторите шаги 1-3, но теперь для размера 72x72 точки и поместите получившийся файл в папку **/res/drawable-hdpi**. Это значок для экранов с высокой плотностью.
5. Повторите шаги 1-3 для размеров 36x36 точек и поместите графический файл в папку **/res/drawable-ldpi**. Это значок для экранов с низкой плотностью точек.
6. Установите значение атрибута `android:icon`, чтобы оно соответствовало имени ресурса вашего значка – `@drawable/myicon`. Система сама выберет одно из трех графических изображений, исходя из плотности размещения точек на экране мобильного телефона (или эмулятора), на котором запущено приложение.

В результате будет выводиться один и тот же значок, но в трех различных разрешениях из трех различных ресурсных папок. Как правило, впоследствии изображение автоматически оптимизируется для каждого разрешения.

Описание приложения

Значение атрибута `android:description` тега `<application>` строка, содержащая краткое описание приложения. Вы можете также указать в этом атрибуте ссылку на строковый ресурс:

```
<application
android:label="My application name"
android:description="@string/app_desc">
```
Система Android и электронные системы дистрибуции используют это описание для отображения информации о приложении.

Установка отладочной информации для приложения

Атрибуту `android:debuggable` тега `<application>` присваивается булево значение, которое разрешает или запрещает отладку приложения отладчиком, например Eclipse, Вы не сможете проводить отладку приложения, если значение этого атрибута не равно `true`. Не забудьте перед публикацией приложения присвоить этому атрибуту значение `false`.

Установка других атрибутов приложения

На вкладке **Application** (Приложение) вы можете найти еще несколько дополнительных настроек, но они применяются довольно редко. Например, если вы хотите применить к приложению вместо оформления по умолчанию какое-либо другое. Здесь есть также настройки для управления тем, как приложение будет взаимодействовать с операционной системой Android- Для большинства приложений вы можете оставить эти настройки без изменений.

На вкладке **Application** (Приложение) вам предстоит выполнить множество действий в разделе **Application Nodes** (узлы приложения), где вы можете зарегистрировать компоненты приложения каждый раз, когда вы регистрируете новую деятельность.

ОПРЕДЕЛЕНИЕ ДЕЯТЕЛЬНОСТЕЙ

Как вы уже знаете, приложения Android включают множество различных деятельностей. Каждая деятельность перед использованием должна быть регистрирована в файле манифеста Android. Вы должны обновлять файл манифеста каждый раз при добавлении новой деятельности к приложению.

Каждая деятельность представляет собой определенную задачу для вытю.4- нения, часто связанную с определенным экраном. Деятельность может быть запущена различными способами с помощью интентов. Каждая деятельность может иметь свою собственную метку (имя) и значок, но по умолчанию метка и значок приложению уже назначены.

Регистрация деятельностей

Вы можете регистрировать новые деятельности в разделе **Application Nodes** (Узлы приложения) вкладки **Application** (Приложение). Каждая новая деятельность обладает собственным тегом `<activity>` в получившемся XML- файле. Например, в следующем фрагменте кода XML происходит определение класса деятельности `DroidActivity`:

```
<activity
android:name=".DroidActivity" />
```

Эта деятельность должна быть определена как класс пакета приложения.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО

Зарегистрируйте новую деятельность в проекте Droidl, выполнив следующие шаги:

1. Откройте проект Droidl в Eclipse.
2. Щелкните правой кнопкой по пакету **/com.androidbook.droidl** в папке /**src** и в открывшемся контекстном меню выберите команду **New** (Новый), затем **Class** (Класс). Откроется диалоговое окно **New Java Class** (Новый класс Java).
3. Назовите новый класс DroidActivity2.
4. Нажмите на кнопку **Browse** (Обзор) рядом с полем ввода **Superclass** (Суперкласс) и задайте суперкласс android.app.Activity.
5. Нажмите на кнопку **Finish** (Завершить). В результате в вашем проекте появится новый класс.
6. Сделайте копию файла макета **main.xml** в папке ресурсов **/res/layout** для созданной только что деятельности и назовите его **second.xml.** Сделайте пометку в макете, чтобы было понятно, что она предназначена для второй активности. Например, вы можете изменить отображаемый текст. Сохраните новый файл макета.
7. Откроите класс DroidActivity2. Щелкните правой кнопкой по названию класса и в открывшемся контекстном меню выберите команду Source (Источник), затем **Override/Implement Methods** (Замена/Применение методов).
8. Установите флажок рядом с методом onCreate(Bundle). В результате этот метод будет добавлен к вашему классу.
9. Затем е коде метода `onCreate()` задайте разметку для загрузки новой активности путем добавления и вызова метода `setContentview(R.layout.second)`. Сохраните файл класса.
10. Откройте файл манифеста Android и щелкните по вкладке **Application** (Приложение) в редакторе ресурсов.
11. В разделе **Application Nodes** (Узлы приложения) вкладки **Application** (Приложение) нажмите на кнопку **Add** (Добавить) и выберите элемент **Activity** (Деятельность). Справа должен появиться раздел **Atributes for Activity** (Атрибуты активности).
12. Нажмите на кнопку **Browse** (Обзор) справа от поля ввода **Name** (Имя) активности. Выберите деятельность DroidActivity2.
13. Сохраните файл манифеста. Перейдите на вкладку **AndroidManifest.xml**, чтобы увидеть изменения в коде XML.

Теперь у вас есть новая, полностью зарегистрированная деятельность DroidActivity2, которую вы можете использовать в вашем приложении.

ВНИМАНИЕ!

Если вы добавляете новый элемент при выделенной активности, то он будет добавлен в эту деятельность. Поэтому следите, чтобы не было выделенных деятельностей, или устанавливайте переключатель в положение Create a new element at the top level, in Application (Создать новый элемент на верхнем уровне, в приложении) вверху диалогового окна.

Определение активности запуска

С помощью интент-фильтра вы можете определить деятельность как основную точку входа приложения. Интент-фильтр запуска деятельности по умолчанию должен быть

сконфигурирован в теге <intent-filter> с типом действия MAIN и категорией LAUNCHER. В вашем проекте Droidl мастер проектов Android установил деятельность DroidActivity как основную точку запуска приложения:

```
<activity
android:name=".DroidActivity"
android:label="@string/app_name">
       <intent-filter>
            <action
android:name="android.intent.action.MAIN" />
            <category
android:name="android.intent.category.LAUNCHER" />
       </intent-filter>
</activity>
```

Таким образом, тег <intent-filter> покатывает системе Android что все запросы запуска приложения должны направляться деятельности DroidActivity.

УПРАВЛЕНИЕ ПРАВАМИ ПРИЛОЖЕНИЯ

Платформа Android основывается на ядре операционной системы Linux, известной своей встроенной системой безопасности, ставшей частью модели безопасности Android. Каждое приложение Android существует в собственной виртуальной машине и работает в собственной учетной записи пользователя Linux (см. рис. 5.6).

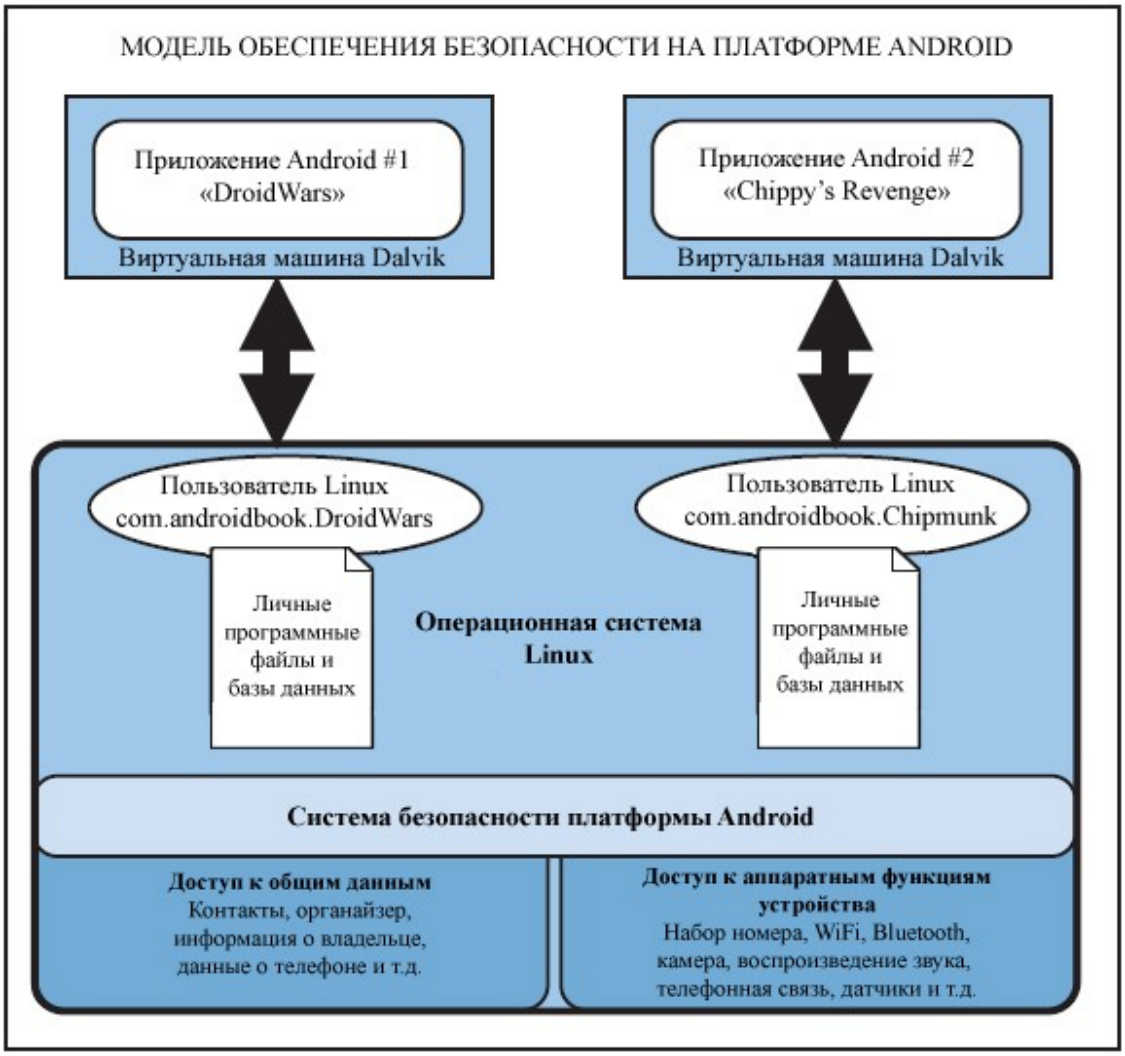

Рис. 5.6. Архитектура платформы Android в упрощенном виде с точки зрения безопасности

Для приложений, требующих доступа к совместно используемых или привилегированным ресурсам на мобильном телефоне, должны быть объявлены определенные права в файле манифеста Android. Такой принцип обеспечения безопасности гарантирует, что никакие приложение не может изменить свое поведение во время выполнения или выполнить какие-либо операции без разрешении пользователем.

КСТАТИ

Поскольку каждое приложение работает под индивидуальной учетной записью, у каждого приложения есть свои файлы и папки так же, как это реализовано для пользователей в операционной системе Linux.

Приложения Android могут получить доступ к собственным файлам и базам данных без наличия специальных прав. Однако, если приложению необходимо полнить доступ к совместно используемым или особым ресурсам, права, разрешающие такие действия, должны быть указаны в теге `<uses- permission>` файла манифеста Android. Управлять правами вы можете на вкладке **Permissions** (Права) в редакторе ресурсов файла манифеста Android.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО

Чтобы предоставить вашему приложению право доступа ко встроенной камере,

выполните следующие шаги:

7. Откройте проект Droidl в Eclipse.
8. Откройте файл манифеста Android и перейдите на вкладку **Permissions** (Права) ресурса редактор.
9. Нажмите кнопку **Add** (Добавить) и выберите пункт **Uses Permission** (Права пользования). Поле ввода атрибута **Name** (Имя) отображается в правой стороне экрана.
10. Введите имя `android.permission. CAMERA`.
11. Сохраните файл манифеста. Переключитесь на вкладку **AndroidManifest.xml,** чтобы увидеть изменения в коде XML.

Таким образом, вы зарегистрировали разрешение на пользование камерой. Теперь ваше приложение может получать доступ с соблюдением прав безопасности.

В табл. 5.2 перечислены некоторые из наиболее часто используемых прав приложениями Android.

Таблица 5.2

Права, требуемые приложениями Android наиболее часто

Категория прав	Права
Геолокационные сервисы	android.permission.ACCESS_COARSE_LOCATION android.permission.ACCESS_FINE_LOCATION
Доступ к базе данных контактов	android.permission.READ_CONTACTS android.permission.WRITE_CONTACTS
Доступ к органайзеру	android.permission.READ_CALENDAR android.permission.WRITE_CALENDAR
Изменение общих настроек мобильного телефона	android.permission.SET_ORIENTATION android.permission.SET_TIME_ZONE android.permission.SET_WALLPAPER
Голосовой вызов	android.permission.CALL_PHONE android.permission.CALL_PRIVILEGED
Отправка и прием сообщений	android.permission.READ_SMS android.permission.RECEIVE_MMS android.permission.RECEIVE_SMS android.permission.RECEIVE_WAP_PUSH android.permission.SEND_SMS android.permission.WRITE_SMS
Использование сетевых соединений	android.permission.INTERNET
Доступ к аудионастройкам	android.permission.RECORD_AUDIO android.permission.MODIFY_AUDIO_SETTINGS
Доступ к настройкам сети	android.permission.ACCESS_NETWORK_STATE android.permission.CHANGE_NETWORK_STATE
Доступ к настройкам WiFi	android.permission.ACCESS_WIFI_STATE android.permission.CHANGE_WIFI_STATE

Доступ ко встроенным устройствам Мобильного телефона	android.permission.BLUETOOTH android.permission.CAMERA android.permission.FLASHLIGHT android.permission.VIBRATE android.permission.BATTERY_STATS
Службы учетных записей	android.permission.GET_ACCOUNTS android.permission.MANAGE_ACCOUNTS
Синхронизация	android.permission.READ_SYNC_SETTINGS android.permission.READ_SYNC_STATS android.permission.WRITE_SYNC_SETTINGS

Полный список прав, используемых приложениями Android, вы можете найти в документации по классу android.Manifest.permission.

ВНИМАНИЕ!

Некоторые права еще не были реализованы в системе Android. Тем не менее приложения всегда должны запрашивать права на выполнение действий для совместимости с последующими версиями системы.

КСТАТИ

Приложения могут определять и осуществлять свои собственные права. Это может быть очень важно для определенных типов приложений, таких как банковская деятельность и коммерческие приложения.

УПРАВЛЕНИЕ ДРУГИМИ НАСТРОЙКАМИ ПРИЛОЖЕНИЯ

Помимо возможностей, о которых рассказано в этом часе, в файле манифест Android можно сконфигурировать многое другое. Например, если вашем приложению требуется аппаратная клавиатура или сенсорный экран, вы можете указать эти требования в файле манифеста.

Вы также должны объявлять здесь все используемые приложением компоненты — например, если ваша программа предоставляет доступ к каким- либо услугам, контенту или функциям приемника.

ИТОГИ

Файл манифеста Android **(AndroidManifest.xml)** присутствует в корневой папке каждого проекта. Это незаменимый компонент любого проекта. В файле манифеста Android используется простая XML-схема для описания самого приложения, его компонентов и прав. Эта информация используется платформой Android для управления приложением. В Eclipse есть удобный редактор ресурсов для управления файлами манифеста Android.

ВОПРОСЫ И ОТВЕТЫ

Вопрос. Существует ли механизм локализации имени приложения для различных языков?

Ответ. Да. Для этого достаточно присвоить атрибуту `android: label` строковый ресурс и создать файлы ресурса для каждого языка, поддержку которого вы хотите добавить. Подробнее о локализации ресурсов мы поговорим в одном из следующих часов.

Вопрос. Зачем добавлять в файл манифеста Android права, которые не используются системой?

Ответ. Хотя ваша программа может функционировать и без объявления этих прав, их все же нужно декларировать. С выходом новой версии системы Android эти права могут потребоваться, и, если они не будут объявлены, это может привести к сбоям в работе приложения.

Вопрос. Я добавил новый класс Activity в проект, и мое приложение стало работать нестабильно. Что я сделал не так?

Ответ. Возможно, вы забыли зарегистрировать деятельность в файле манифеста Android. В этом случае приложение может вообще не запуститься.

Вопрос. Зачем редактировать файл манифеста Android в текстовом редакторе, если можно использовать редактор ресурсов Eclipse?

Ответ. В большинстве случаев редактировать файл манифеста удобнее всего в редакторе ресурсов. Но большие изменения лучше вносить в текстовом редакторе.

Вопрос. Зачем объявлять право на чтение общедоступных ресурсов. как список контактов?

Ответ. Чтобы обеспечить защиту конфиденциальных данных пользователя, приложения всегда должны регистрировать право на чтение уязвимых данные. Несмотря на то, что приложение не может причинить вред системе Android чтением данных, информация, хранящаяся в списке контактов или органайзере, теоретически может использоваться в преступных целях. Поэтому приложение должно в явном виде получить право доступа к подобного рода информации.

ПРАКТИКУМ
Контрольные вопросы

1. Каждое приложение Android должно иметь собственный файл манифеста. Правда ли это?

2. Номер в значении атрибута `android:versionCode` должен соответствовать имени версии приложения, хранящемуся в атрибуте `android: versionName`. Правда ли это?

3. Какое из приведенных ниже выражений служит для предоставления доступа к использованию видеокамеры?

A. `android.permission.USE_CAMERA.`
B. `android.permission.CAMERA.`
C. `android.permission.hardware.CAMERA.`

4. Перед установкой приложения пользователь может видеть все права, объявленные в файле манифеста Android. Правда ли это?

Ответы

1. Правда. Файл манифеста Android — неотъемлемая часть каждого проекта Android. Этот файл определяет идентичность приложения, его настройки и права.

2. Нет, это не так. Версия приложения, хранящаяся в атрибуте `android:versionCode`, должна повышаться при каждом внесении изменений в приложение, она может обновляться. Этот номер не обязательно должен соответствовать значению атрибута `android:versionName`.

3. Правильный ответ: Б. Для получения доступа к видеокамере используется выражение `android.permission.CAMERA`.

4. Правда. Это делается для того, чтобы пользователь знал, какими правами обладает приложение, например создание снимков с видеокамеры или доступ к списку контактов пользователя.

Упражнения

1. Создайте еще один значок для проекта Droidl.

2. Создайте другой класс Activity. Измените какие-нибудь права приложения на ваш выбор. Кроме того, попытайтесь использовать какую-либо возможность системы Android, не запрашивая соответствующее право заранее, и оцените результат.

3. Измените файл манифеста с помощью редактора файла манифеста. Обратите внимание на изменения, произошедшие в коде XML. Теперь попробуйте сделать го же самое непосредственно в коде XML.

Час 6. РАЗРАБОТКА КАРКАСА ПРИЛОЖЕНИЯ

Вопросы, рассматриваемые в этом часе:

- **проектирование простого приложения Android;**
- **подготовка прототипа приложения к запуску;**
- **запуск прототипа игры.**

Настало время применить навыки, полученные в предыдущих часах. В этом часе вы разработаете прототип приложения Android — основной каркас, на котором к концу книги вы построите полноценное приложение.
По мере освоения книги вы реализуете в приложении множество интересных функций.

ПРОЕКТИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ ANDROID

Поскольку в последнее время большую популярность приобрели простые социальные игры, в качестве примера приложения мы разработаем одну из таких игр. В процессе нашей игры пользователю будут задаваться случайные вопросы о том, в каких путешествиях он побывал и какой получил опыт, например:

- Бывали ли вы когда-нибудь у египетских пирамид?
- Вы когда-нибудь доили корову?
- Занимались ли вы когда-нибудь дайвингом с белыми акулами?
- Вы когда-нибудь поднимались на высокую скалу?

Игрок, прошедший большую часть игры и достигший наилучших результатов, должен обладать самым высоким счетом. Игра будет называться «Been There, Done That!».

Определение высокоуровневых игровых функций

Попробуйте представить, какими возможностями должно обладать хорошее игровое приложение. Кроме игрового экрана с вопросами об игре вам понадобятся:

- экранная заставка;
- вывод на экран заработанных игроком очков;
- вывод текста с разъяснением игровых правил;
- возможность сохранения игровых настроек.

Вы также должны придумать способ перехода между различными экранами «провою интерфейса. Один из способов решения этой проблемы — создать экран главною меню, с помощью которою пользователь может перемешаться между экранами приложения.

Исходя из перечисленных требований, в игровом приложении «Been There, Done That!* нам может понадобиться шесть основных экранов:

- экран-заставка;
- экран главного меню;
- основной игровой экран;
- экран настроек;
- экран таблицы рекордов;
- экран справки.

Эти экраны составят основу пользовательского интерфейса приложения.

Требования, предъявляемые к деятельностям

У каждого экрана игрового приложения «Been There, Done That!» будет собственный класс Activity. На рис. 6.1 представлена схема шести необходимых деятельностей, по одной для каждого экрана.

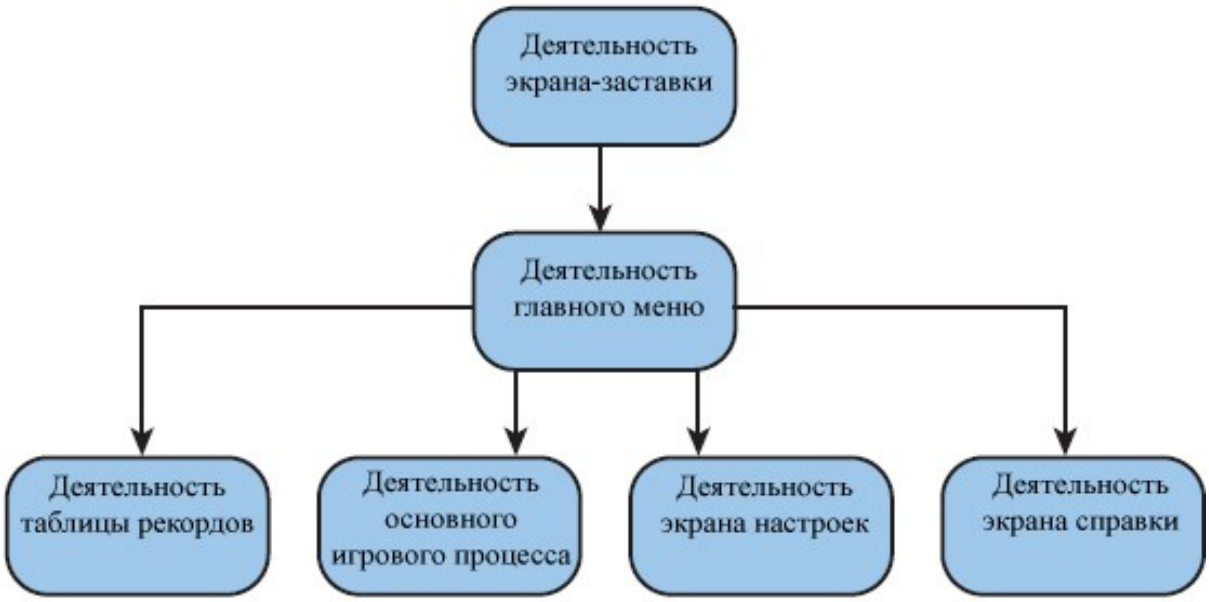

Рис. 6.1. Схема взаимосвязи деятельностей
в игровом приложении «Been Three, Done That!»

Практика показывает, что удобнеее реализовать базовый класс Activity с общедоступными компонентами. Он будет называться QuizActivity. Таким образом, будут определятся все деятельности, необходимые в приложении «Been There, Done That!»:

- Класс QuizActivity, полученный из android, app. Activity, будет основным классом. С его помощью вы определите параметры приложения и другие настройки и возможности приложения.

- Класс QuizSplashActivily, полученный из класса QuizActivity, представляет собой экран-заставку.

- Класс QuizMcnuActivity, полученный из класса QuizActivity, представляет собой экран главною меню.

- Класс QuizHelpActivity, полученный из класса QuizActivity, представляет собой экран справки.

- Класс QuizScoresActivity, полученный из класса QuizActivity, представляет собой экран таблицы рекордов.

- Класс QuizSettingsActivity, полученный из класса QuizActivity, представляет собой экран настроек.

- Класс QuizGameActivity, полученный из класса QuizActivity, представляет собой основной игровой экран.

Определение игровых возможностей для каждого экрана

Теперь пришло время определить основные возможности каждой деятельности в игровом приложении «Been There, Done That!».

Рис. 6.2. Экран-заставка в игре «Been There, Done That!».

ОПРЕДЕЛЕНИЕ ВОЗМОЖНОСТЕЙ ЭКРАНА-ЗАСТАВКИ

Экран-заставка в игре «Been There, Done That!» служит отправной точкой для последующих действий игрока. Вся его функциональность должна заключаться в классе QuizSplashActivity. На этом экране будут выполняться следующие действия:

- отображение имени и версии приложения;
- отображение графического изображения или логотипа игры;
- автоматический переход на экран главного меню через определенный промежуток времени.

На рис. 6.2 изображен примерный вид экрана-заставки.

ОПРЕДЕЛЕНИЕ ВОЗМОЖНОСТЕЙ ЭКРАНА ГЛАВНОГО МЕНЮ

Экран главного меню служит для навигации по возможностям приложения. Этот экран появляется сразу после заставки и предоставляет пользователю выбор дальнейших действий. Его функциональность заключается в классе QuizMenuActivity. Этот экран должен:

- автоматически отображаться после заставки;
- предоставлять пользователю возможность выбора пунктов меню **Play Game**, **Settings**, **Scores** или **Help**.

На рис. 6.3 изображен примерный вид экрана главного меню.

ОПРЕДЕЛЕНИЕ ВОЗМОЖНОСТЕЙ ЭКРАНА СПРАВКИ

На экране справки пользователь может ознакомиться с **правилами** игры, Эта функциональность заключается в классе QuizHelpActivity. На этом экране должны выполняться следующие операции:

Рис. 6.3. Экран главного меню в игре «Been There, Done That!»

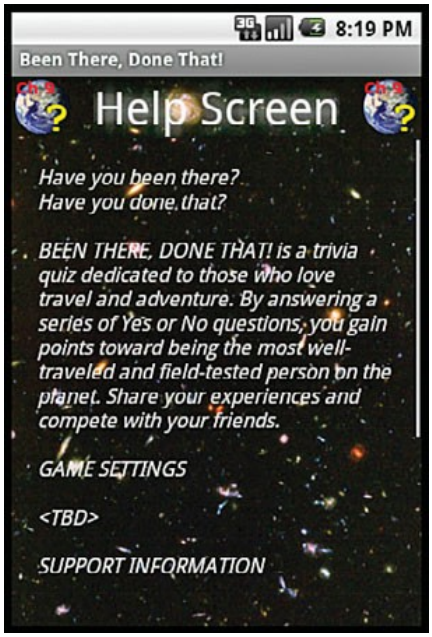

Рис. 6.4. Экран справки в игре «Been There, Done That!»

- отображение текста справки и предоставление возможности выполнять прокрутку текста;
- предоставление навигации по содержанию справочной информации.

На рис. 6.4 изображен примерный вид экрана-справки.

ОПРЕДЕЛЕНИЕ ВОЗМОЖНОСТЕЙ ЭКРАНА ТАБЛИЦЫ РЕКОРДОВ

В таблице рекордов пользователи могут видеть текущий игровой счет с перечнем игроков, заработавших наибольшее количество очков. Эта функциональность должна быть заключена в классе QuizScoresActivity. На этом экране должны выполняться следующие операции:

- отображение статистики лучших игровых достижений;
- отображение только текущего счета, если пользователь перешел на этот экран в процессе игры.

На рис. 6.5 представлен примерный вид экрана таблицы рекордов.

ОПРЕДЕЛЕНИЕ ВОЗМОЖНОСТЕЙ ЭКРАНА НАСТРОЕК

На экране настроек пользователи могут редактировать и сохранять параметры игры, включая имя пользователя и другие. Эта функциональность заключается в классе QuizSettingsActivity. Здесь должны выполняться следующие операции:

Рис. 6.5. Экран таблицы рекордов игры «Been There, Done That!»

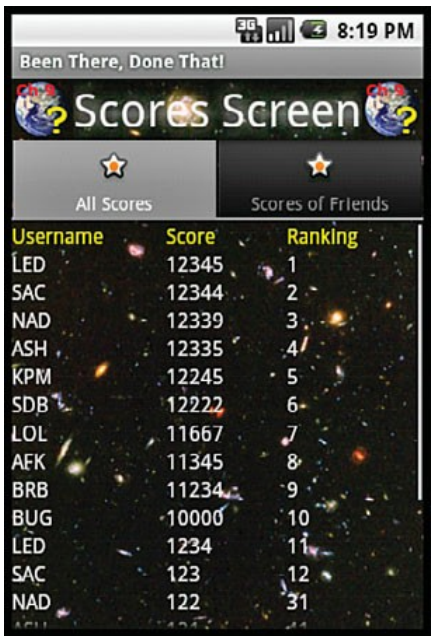

Рис. 6.6. Экран настроек в игре «Been There, Done That!»

- изменение игровых настроек пользователем;
- приглашение других игроков для совместной игры.

На рис. 6.6 изображен примерный вид экрана настроек.

ОПРЕДЕЛЕНИЕ ВОЗМОЖНОСТЕЙ ОСНОВНОГО ИГРОВОГО ЭКРАНА

На игровом экране будут отображаться вопросы. Эта функциональность должна быть заключена в классе QuizGameActivity. На этом экране должны выполнятся следующие действия:

- вывод серии вопросов да/нет;
- обработка вводимых данных, сохранение текущего счета и состояния игры;
- переход к экрану счета после завершения игры.

На рис. 6.7 изображен примерный экран основного игрового экрана.

Рис. 6.7. Основной игровой экран в игре «Been There, Done That!»

РЕАЛИЗАЦИЯ ПРОТОТИПА ПРИЛОЖЕНИЯ

Теперь, когда у вас есть общее представление о том, как будет выглядеть программа «Been There, Done That!» и как она будет работать, настало время для написания кода. Этот процесс состоит из нескольких этапов:

1. Создание ноною проекта Android в Eclipse/
2. Добавление новых ресурсов проекта, включая текст и графику.
3. Создание разметки для каждого экрана.
4. Реализация деятельности для каждого экрана
5. Настройка параметров для всего приложения.

КСТАТИ

Код для данного проекта доступен на прилагаемом *г* книге диске в папке /Книга/ Час 06.

Создание нового проекта Android

Создать новый проект приложения вы можете с помощью мастера проектов Android в Eclipse.

При создании проекта вам необходимо ввести следующие данные:

- **Project name** (Название проекта): **TriviaQuiz**.
- **Build target** (Целевая платформа): **Android 2.1**.
- **Application name** (Название приложения): Been There. Done That!.
- **Package name** (Название пакета): com.androidbook.triviaquiz.
- **Create activity** (Создать деятельность): QuizSplashActivity.

Это лишь основные параметры нового проекта Android. Кроме того, вам необходимо сделать еще несколько корректировок.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

В коде листингов этой книги (которые вы можете найти на прилагаемом диске), имена пакетов содержат номер часа, которому они принадлежат. Например, имя пакета для этого часа — **com.androidbook.triviaquiz6**. Таким образом, вы можете совмещать несколько проектов на одном мобильном телефоне и загружать их одновременно.

Добавление ресурсов в проект

В проекте «Been There, Done That!» необходимо добавить несколько дополнительных ресурсов. В частности, вам нужно добавить файл макета для каждой деятельности и текст для каждого имени деятельности. Еще неплохо бы изменить значок приложения на более подходящий.

ДОБАВЛЕНИЕ СТРОКОВЫХ РЕСУРСОВ

Сначала необходимо отредактировать файл ресурса **strings.xml**. Удалите стоку с именем `hello` и создайте шесть новых строковых ресурсов — по одному для каждого экрана. Например, вы можете создать строковый ресурс `help` со значением `Help Screen`**.** После выполнения этих действий файл strings.xml должен выглядеть так:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <string
name="app_name">Been There, Done That!</string>
      <string
name="help">Help Screen</string>
      <string
name="menu">Main Menu Screen</string>
      <string
name="splash">Splash Screen</string>
      <string
name="settings">Settings Screen</string>
      <string
name="game">Game Screen</string>
      <string
name="scores">Scores Screen</string>
</resources>
```

ДОБАВЛЕНИЕ РЕСУРСОВ МАКЕТА

Далее вам нужно создать файл макета для каждой деятельности. Переименуйте файл макета **main.xml** в **splash.xml**. Затем сделайте пять копий файла **splash.xml** и присвойте этим копиям следующие имена: **game.xml**, **help.xml**, **menu.xml**, **scores.xml** и **settings.xml**.

Вы можете встретить предупреждение ошибке в каждом файле макета. Ошибка заключается в том, что элемент управления `TextView` в макете ссылается на несуществующую строку `@string/hello`, которую вы удалили. Поэтому в каждом файле макета вы должны изменить ссылку на ресурсы, загружаемые элементом управления `TextView`. Например, в файле game.xml вам необходимо заменить ссылку на ресурс `@string/hello` новой строкой — `@string/game`. Теперь после загрузки файлов разметки они должны отображать соответствующий экран.

ДОБАВЛЕНИЕ ГРАФИЧЕСКИХ РЕСУРСОВ

Нам нужно заменить значок приложения на более подходящий. Например, вы можете создать значок **quizicon.png** размером 48x48 из исходного снимка Земли со спутника и добавить в папку ресурсов **/drawable**. После этого удалите файл **icon.png**, используемый по умолчанию.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

В коде проекта этого часа и будущих часов для графических ресурсов будет использоваться единственная папка — **/drawable**. Каждый раз, когда системе Android понадобится графический ресурс, она будет обращаться только к этой папке, вместо папок по умолчанию [**/drawabte-ldpi**, **/drawable-mdpi** и **/drawable-hdpi**). В этой книге к имени файла значка **quizicon.png** будет присоединяться порядковый номер часа, чтобы отличать, к какому часу принадлежит тот или иной файл.

Реализация деятельностей приложения

Чтобы реализовать основной класс `Activity`, вы можете просто скопировать исходный файл `QuizSplashActivity.java`. Затем создайте новый файл класса `QuizActivity`. Этот класс должен выглядеть очень просто:

```
package com.androidbook.triviaquiz6;
import android.app.Activity;
public class QuizActivity extends Activity {
      public static final String GAME_PREFERENCES = "GamePrefs";
}
```

Остальное вы добавите в этот класс позже, Обновите класс `QuizSplashActivity`, чтобы расширить его из класса `QuizActivity` вместо класса Activity напрямую.

СОЗДАНИЕ ОСТАЛЬНЫХ ДЕЯТЕЛЬНОСТЕЙ ПРИЛОЖЕНИЯ

Скопируйте деятельность `QuizSplashActivity` пять раз — по одному разу для каждой активности: `QuizMenuActivity`, `QuizHelpActivity`, `QuizScoresActivity`, `QuizSettingsActivity` и `QuizGameActivity`.

Обратите внимание, как удобно Eclipse обновляет имя класса при копировании файла класса. Создавать файлы классов вы можете также, щелкнув правой кнопкой по имени пакета `com.androidbook.triviaquiz` и выбрав команду **New Class** (Новый Класс). В открывшемся диалоговом окне вы можете выбрать параметры создаваемого класса.

КСТАТИ

Несколько полезных рекомендаций, касающихся работы в Eclipse, вы можете найти в приложении Б.

Обратите внимание, что сейчас в каждом Java-файле присутствует ошибка. Она возникает из-за того, что каждая деятельность пытается загрузить файл макета main.xml, которого больше нет. Вы должны изменить каждый класс, чтобы иметь возможность загружать

соответствующие макеты, связанные с деятельностью. Например, в классе `QuizHelpActivity` вы должны изменить метод `setContentView()`, чтобы загружать файл макета, созданного вами для экрана справки, следующим образом:

```
setContentView(R.layout.help);
```

Сделайте аналогичные изменения в других файлах деятельности, чтобы при каждом вызове метода `setContentView()` загружался соответствующий ему файл макета.

ОБНОВЛЕНИЕ ФАЙЛА МАНИФЕСТА ANDROID

Теперь вам необходимо сделать некоторые изменения в файле манифеста Android. Во-первых, вам надо изменить ресурс значка приложения, чтобы ссылка `@drawable/quizicon` указывала на созданный вами значок. Во-вторых, чтобы новые деятельности выполнялись правильно, их нужно зарегистрировать в файле манифеста. Наконец, вы должны установить значение `true` в атрибуте приложения `Debuggable` и убедиться, что деятельность `QuizSplashActivity` является деятельностью запуска по умолчанию.

КСТАТИ

В нашем случае мы создали только один значок. Тем не менее, если вы уже создали три значка разных размеров и поместили их в папки по умолчанию (`/drawable-Idpi`, `/drawable-mdpi` и `/drawable-hdpi`), использоваться будет только один значок. Обязательно убедитесь, что все значки называются одинаково. Таким образом, система Android может автоматически выбрать подходящий значок с учетом аппаратных особенностей устройства, на котором запущено приложение.

Установка параметров приложения

В игре «Been There, Done That!» вам нужен простой механизм сохранения информации о состоянии и пользовательских данных. Эту возможность вы можете реализовать с помощью общедоступных параметров системы Android **(`android.content.SharedPreferences`).**

Получить доступ к общедоступным параметрам вы можете по его имени из любой деятельности приложения. Для этого объявите имя набора параметров в базовом классе **`QuizActivity`** так, чтобы они стали доступны всех подклассов:

```
public static final String GAME_PREFERENCES = "GamePrefs";
```

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Количество наборов параметров, которые вы можете создать, не ограничено Разделить параметры по категориям вы можете строкой имени параметра, например игровые параметры и параметры пользователя То, как будут организованы общедоступные параметры, зависит от вас.

Чтобы добавить общедоступные параметры в приложение, выполните еле- дующие шаги:

1. Методом `getSharedPreferences()` получите экземпляр объекта `SharedPreferences`.
2. Создайте объект `SharedPreferences.Editor`, чтобы изменить параметры.
3. Сделайте изменения в параметрах с помощью редактора.
4. Примените изменения методом commit () в редакторе.

СОХРАНЕНИЕ ОБЩЕДОСТУПНЫХ ПАРАМЕТРОВ

Каждый параметр представляет собой пару `ключ/значение`. Значения параметров могут быть следующих типов:

- булев;
- число с плавающей точкой;
- целое число;
- длинное целое число;
- строка.

После того, как вы решили, какие параметры хотите сохранять, вы должны получить экземпляр объекта `SharedPreferences`, затем с помощью объекта `Editor` сделать изменения и сохранить их. В следующем примере сохраняются два параметра — имя пользователя и его возраст:

```
import android.content.SharedPreferences;
// ...
SharedPreferences settings =
getSharedPreferences(GAME_PREFERENCES, MODE_PRIVATE);
SharedPreferences.Editor prefEditor = settings.edit();
prefEditor.putString("UserName", "JaneDoe");
prefEditor.putInt("UserAge", 22);
prefEditor.commit();
```

Вы можете также использовать редактор общедоступных параметров для их сброса методом `clear()` и удаления определенных параметров по имени методом `remove ()`.

ПОЛУЧЕНИЕ ОБЩЕДОСТУПНЫХ ПАРАМЕТРОВ

Получение значений общедоступных параметров намного проще, чем их создание, потому что для этого вам не нужен редактор. В следующем примере показано, как можно получить доступ к значениям общедоступных параметров:

```
SharedPreferences settings =
getSharedPreferences(GAME_PREFERENCES, MODE_PRIVATE);
if (settings.contains("UserName") == true) {
// We have a user name
String user = Settings.getString("UserName", "Default");
}
```

Вы можете использовать объект `SharedPreferences` для проверки параметров по имени, получить строго введенные параметры или получить все параметры и сохранять их на карте.
Несмотря на то, что в игре «Been Three, Done That!» общедоступные параметры пока не используются, полученные знания о них помогут вам при разработке активностей

приложения. Они пригодятся, когда вы начнете применять каждую деятельность в этой книге позже.

ПОДГОТОВКА ПРОТОТИПА ИГРЫ

Уже сейчас вы почти готовы к выполнению и тестированию вашего приложения. Но сначала вы должны создать конфигурацию отладки для вашего нового проекта в Eclipse.

Создание конфигурации отладки

Вы должны создать в Eclipse конфигурацию отладки для каждого проекта. Убедитесь, что выбранное для проекта AVD совместимо с Google API (API Level 7). Если у вас нет такого AVD (или вы не уверены в этом), нажмите на кнопку вызова Android SDK and AVD Manager в Eclipse. Здесь вы можете определить, какие AVD лучше всего подходят приложению, и создать новые, если необходимо.

Запуск прототипа в эмуляторе

Теперь все готово для запуска приложения «Been There, Done That!» в эмуляторе Android. Вы можете сделать это, щелкнув по кнопке отладки в Eclipse или на кнопке **Run** (Выполнение) в созданной только что конфигурации отладки.

Рис. 6.8. Прототип игры «Been Three, Done That!» рядом с другими приложениями

Как вы можете видеть на рис. 6.8. приложение пока еще довольно простое. У него есть оригинальный значок, нажав на который, пользователь может запустить деятельность по умолчанию `QuizSplashActivity`. Эта деятельность выводит на экран элемент интерфейса `TextView`, сообщая вам о том, что вы находитесь на экране-заставке. В приложении пока еще нет никакого пользовательского интерфейса, поэтому вы должны

переладить между различными деятельностями вручную. Однако у вас уже есть готовый каркас приложения для последующей разработки. В следующих часах вы расширите возможности различных экранов и начнете реализовывать функции игры.

Установка прототипа приложения

Приложение «Been There, Done That!» в текущем виде выполняет довольно мало задач, но с помощью эмулятора Android и других инструментов вы можете оценить результат своей работы.

- **Application Manager** (Менеджер приложений) — полезен для поиска интересной информации о приложении. В эмуляторе переместитесь на экран по умолчанию, нажмите на кнопку **Menu** (Меню), выберите комманду **Settings=>Applications=> Manage applications** (Настройки=>Приложение=>Управление приложениями) и выберите приложение «Been There, Done That!» из списка. Здесь вы можете видеть основную информацию о приложении, включая занимаемый объем и используемые права, а также информацию о кэше и т.д.

- **Dev Tools** (Средства разработки) - с их помощью вы можете получить более подробную информацию о приложении. Зайдите в эмулятор, перечите в меню приложений, запустите приложение **Dev Tools** (Средства разработки) и выберите **Package Browser** (Обзор пакетов). Переместитесь к пакету com.androidbook.triviaquiz. С помощью этого средства вы можете увидеть информацию о манифесте и проверить настройки каждой зарегистрированной активности.

Разумеется, вы можете исследовать приложение в перспективе **DDMS.** Например. вы можете проверить папку приложения пакета com.androidbook.triviaquiz в файловой системе Android. Вы можете также рассмотреть код деятельности `QuizSplashActivity`**.**

ИТОГИ

В этом часе вы создавали прототип приложения, на основе которого в следующих часах вы можете собрать полноценное приложение. Вы спроектировали прототип и подробно определили его требования. Потом вы создали новый проект Android, сконфигурировали его и создали деятельность для каждого экрана. Вы также добавили собственные макеты и применили общедоступные параметры в приложении.

ВОПРОСЫ И ОТВЕТЫ

Вопрос. Какой класс может быть наследован для обеспечения приложению общедоступных компонентов?

Ответ. Создав собственный общедоступный класс `Activity`**,** вы можете реализовать одинаковое поведение для каждого экрана.

Вопрос. Могут ли у деятельности быть собственные параметры?

Ответ. Да, параметры могут совместно использоваться деятельностями, и у каждой деятельности могут быть собственные параметры. Для доступа к общедоступным параметрам используется метод `getSharedPreferences()`. Для доступа к параметрам деятельности используется метод `getPreferences()`.

Вопрос. Что необходимо сконфигурировать перед выполнением и отладкой приложения Android в Eclipse?

Ответ. Вы должны сконфигурировать и AVD, и конфигурацию отладки. После этого вы можете легко запустить свое приложение прямо в Eclipse для последующей отладки и тестирования.

ПРАКТИКУМ
Контрольные вопросы

1. В приложении «Been There, Done That!» всею три деятельности. Так ли это?
2. Какие типы данных поддерживаются в общедоступных параметрах приложения?
   A. Булев тип, вещественное число, целое число, длинное целое число и строковый тип.
   B. Булев тип, целое число и строковых данных.
   C. Все типы, доступные в Java.
3. Вы можете обойтись лишь основным классом деятельности (например, QuizActivity) в файле манифеста Android. Правда ли это?

Ответы

1. Нет, не так. В приложении «Been There, Done That!» для каждого из шести экранов создана отдельная деятельность. В приложении есть также основной класс деятельности, из которого происходят все остальные. Всего в проекте семь классов деятельности.

2. Правильный ответ: A. Допускается использование данных булева типа, вещественных чисел, целых чисел, длинных целых чисел и строковых данных.

3. Нет. У каждой деятельности должна быть собственная запись в файле манифеста Android.

Упражнения

1. Добавьте к параметрам приложения еще один параметр — `lastLaunch`. Произведите считывание деятельности `QuizSplashActivity` и сохраняйте текущее состояние после каждого вызова метода `onCreate()`. После считывания этого параметра занесите его в журнал методом `Log.i()`.

2. Создайте ряд параметров для деятельности `QuizSettingsActivity` методом `getPreferences()` вместо `getSharedPreferences()`.

ЧАСТЬ II. ПОСТРОЕНИЕ КАРКАСА ПРИЛОЖЕНИЯ
Час 7. РЕАЛИЗАЦИЯ АНИМИРОВАННОГО ЭКРАНА-ЗАСТАВКИ

Вопросы, рассматриваемые в этом часе:

- **разработка дизайна экрана-заставки;**
- **обновление макета экрана-заставки;**
- **работа с анимацией.**

В этом часе мы сосредоточимся на реализации экрана-заставки для приложения «Been There, Done That!*. Создав эскиз дизайна экрана-заставки, вы сможете точно определить, какие элементы пользовательского интерфейса View платформы Android вам понадобится включить в файл макета **splash.xml.** Когда вы будете удовлетворены внешним видом макета экрана- заставки. вы сможете добавить несколько анимаций движения, чтобы придать экрану-заставке некую пикантность. Наконец, вам нужно создать плавный переход между экраном-заставкой и экраном с основным меню, когда воспроизведение ваших анимаций будет завершено.

РАЗРАБОТКА ДИЗАЙНА ЭКРАНА-ЗАСТАВКИ

Вы будете реализовывать приложение «Been There, Done That!», начав с экрана-заставки, который будет виден пользователям сразу после запуска приложения. Как было отмечено в часе 6, к этому экрану есть несколько требований. В частности, на нем должна отображаться определенная информация о приложении (его название и версия), и затем, по истечении небольшого промежутка времени, должен происходить автоматический переход к экрану основного меню. На рис. 7.1 изображен эскиз дизайна экрана-заставки.

Для начала разработайте экран-заставку в портретной ориентации, но сразу подумайте о том, как максимально упростить последующий переход на альбомную или квадратную ориентацию. Разработав простой дизайн экрана-заставки, вы можете быть уверены, что в других ориентациях элементы экрана будут довольно хорошо масштабироваться при условии, что вы настроите размеры шрифтов и изображений. Проблемы, связанные с поддержкой различных устройств, рассматриваются в последующих часах.

Рис. 7.1. Эскиз дизайна экрана-заставки
для приложения Been There, Done That!

РЕАЛИЗАЦИЯ МАКЕТА ЗАСТАВКИ

Теперь, когда вы знаете, как должен выглядеть экран-заставка вашего приложения, нужно преобразовать эскиз в соответствующий дизайн-макет. Напомним, что файл макета **/res/layout/sptash.xml** используется деятельностью `QuizSplashActivity`. Вы должны обновить стандартный макет, на котором отображается один элемент `TextView` (сообщая нам о том, что это экран-заставка), чтобы добавить элементы пользовательского интерфейса для каждого элемента, представленного на эскизе дизайна.

Макеты экранов формируются при помощи разнообразных элементов пользовательского интерфейса. Каждый элемент — прямоугольник, контролирующий определенную часть экрана. На экране-заставке используются два распространенных элемента пользовательского интерфейса:

- Элемент TextView используется для отображения на экране текстовой строки.
- Элемент ImageView используется для отображения на экране изображения.

Вам также нужен способ для организации различных элементов пользовательского интерфейса View в определенном порядке. Для этого применяются элементы-контейнеры `Layout`. Например, элемент-контейнер `LinearLayout` позволяет расположить дочерние элементы-представления друг за другом в вертикальном или горизонтальном направлении.

Помимо элемента-контейнера `LinearLayout` существует ряд других элементов-контейнеров `Layout`. Элементы-контейнеры могут быть вложенными и контролировать либо некоторую часть экрана, либо весь экран. Довольно часто элементы пользовательского интерфейса размещаются в одном большом родительском элементе-контейнере — например, `LinearLayout`. В табл. 7.1 перечислены доступные элементы-контейнеры `Layout`.

Таблица 7.1

Распространенные элементы-контейнеры Layout

Название элемента-контейнера Layout	Описание	Ключевые атрибуты/элементы
LinearLayout	Каждый дочерний элемент-представление размещается за предыдущим элементом-представлением в одной строке или столбце	Ориентация (вертикальная или горизонтальная)
RelativeLayout	Каждый дочерний элемент-представление размещается относительно других элементов-представлений в макете или относительно границ родительского элемента-контейнера	Множество атрибутов выравнивания, позволяющих управлять размещением дочернего элемента-представления относительно других дочерних элементов-представлений `View`
FrameLayout	Каждый дочерний элемент-представление размещается внутри фрейма, относительно его левого верхнего угла. Элементы-представления `View` могут перекрываться	Порядок размещения дочерних элементов-представлений `View` имеет значение, когда используются соответствующие значения атрибута `gravity`
TableLayout	Каждый дочерний элемент-представление размещается в одной из ячеек сетки, образуемой строками и столбцами	Каждая строка представляется элементом `TableRow`

КСТАТИ

В инструментарии Android 1.5 SDK элемент-контейнер `AbsoluteLayout`, позволяющий указывать определенные значения координат X/Y для элементов, был объявлен устаревшим, однако он до сих пор доступен для использования. И хотя использовать этот элемент-контейнер не рекомендуется, иногда полезно иметь элемент-контейнер с возможностью указания точных координат X/Y для определенных типов макетов экранов, которые требуют точного размещения элементов по пикселам. Этот элемент-контейнер используется в элементе пользовательского интерфейса `WebView` инструментария SDK.

Элементы-контейнеры и их дочерние элементы-представления `View` имеют определенные атрибуты, которые помогают управлять их повелением. Например, все элементы-контейнеры имеют атрибуты `android:layout_width` и `android:layout_height`, позволяющие управлять шириной и высотой элемента-контейнера. В качестве значений лих атрибутов могут использоваться конкретные измерения, например число пикселов, или может применяться более гибкий подход с использованием значений `fill_parent` или `wrap_content`. Когда используется значение `fill_parent`, элемент-контейнер масштабируется до размеров родительского элемента-контейнера, а использование значения `wrap_content` позволяет «растянуть» родительский элемент-контейнер вокруг дочернего элемента-представления `View`, подогнав размеры родительского элемента-контейнера до размеров дочернего элемента-представления `View`. Существует ряд других интересных свойств, которые могут быть использованы для управления поведением отдельных элементов-контейнеров, включая

настройки отступов и атрибуты, характерные для конкретных типов элементов-контейнеров.

Давайте воспользуемся элементом-контейнером `TableLayout`, чтобы отобразить несколько элементов-представлений `ImageView`, являющихся частью нашего экрана-заставки.

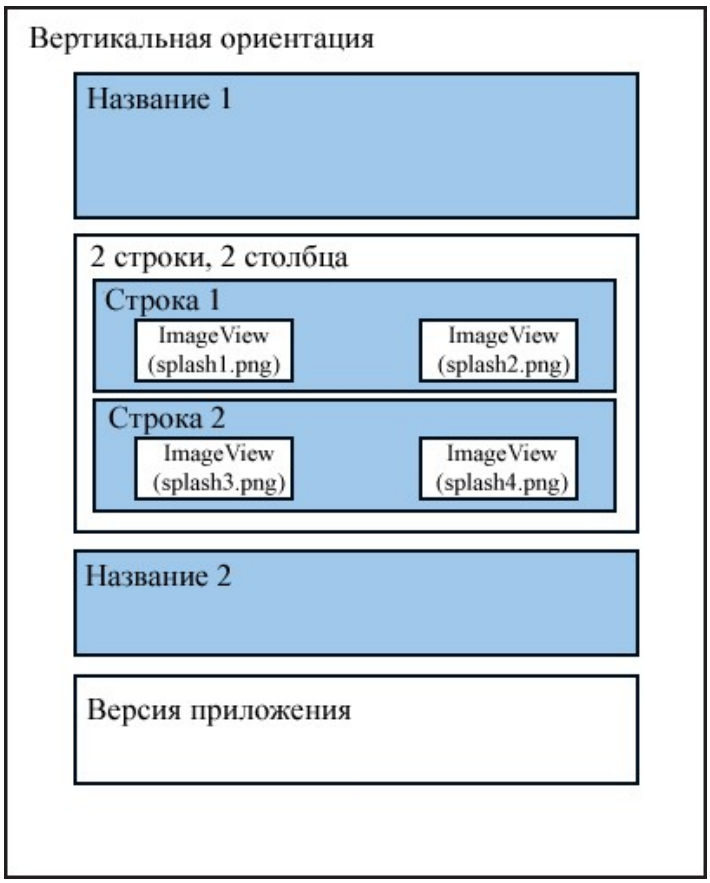

Рис. 7.2. Дизайн-макет экрана-заставки для приложения «Been There, Done That!»

В дизайне экрана-заставки вы можете использовать элемент-контейнер `LinearLayout` с вертикальной ориентацией, чтобы разместить элементы пользовательского интерфейса данного экрана в следующем порядке: элемент `TextView`, элемент-контейнер `TableLayout` с несколькими элементами `TableRow`, содержащими элементы `ImageView`, и два дополнительных элемента `TextView`. На рис. 7.2 изображен дизайн-проект экрана-заставки.

Добавление новых ресурсов в проект

Теперь, когда у вас есть дизайн-макет экрана-заставки для приложения, настаю время создать строковые ресурсы, ресурсы цветов и размеров, которые будут использованы в макете.

Сначала вы добавите четыре новых графических ресурса в каталог **/res/drawable** (создав при необходимости этот каталог): **splashl.png**, **splash2.png**, **splash3.png** и **splashl.png**. Эти изображения будут отображаться внутри элемента-контейнера Table Layout в центре экрана-заставки.

КСТАТИ

Код для данного проекта доступен на прилагаемом к книге диске в папке **/Книга/Час 07**

Затем вы можете добавить три новых строковых ресурса в файл ресурсов **/res/values/strings.xml**: один для верхнего заголовка (Been There), один для нижнего заголовка (Done That!) и один для информации о версии приложения (эта информация будет многострочной). Строковый ресурс `splash` теперь можно удалить, поскольку он больше не будет использоваться.

После этого вы создадите новый файл ресурсов **/res/values/colors.xml**, который будет содержать три необходимых ресурса цвета: один для текста заголовка (золотисто-желтый), один для текста версии приложения (серовато-белый) и один для цвета фона, поверх которого будет отображаться текст версии (насыщенно-синий).

КСТАТИ

Поскольку общий цвет фона заставки будет черным, вам не нужно создавать отдельный ресурс цвета. Вместо этого вы просто воспользуетесь встроенным ресурсом платформы Android `@android:color/black`.

Наконец, вам нужно создать несколько ресурсов размеров в новом файле ресурсов **/res/values/dimens.xml**. Вам потребуется три новых значения размеров: один для управления размером шрифта заголовка (24pt), один для управления размером шрифта версии приложения (5pt) и один для задания промежутка между строками текста с информацией о версии приложения (3pt).

После того, как вы сохраните файлы ресурсов, можно приступать к использованию новых ресурсов в файле макета **splash.xml**.

Обновление макета экрана-заставки

Вы могли бы изменить существующий макет **splash.xml,** но иногда бывает проще начать с нуля, используя редактор ресурсов в среде разработки Eclipse**,** чтобы удалить все существующие элементы пользовательского интерфейса. Затем нужно выполнить следующие шаги, чтобы создать желаемый макет экрана в соответствии с разработанным дизайн-макетом.

1. Сначала добавьте новый элемент-контейнер LinearLayout и присвойте его атрибуту `background` значение `@android:color/black`, а атрибуту `orientation` — значение `vertical`. Все остальные элементы пользовательского интерфейса будут добавляться в элемент-контейнер `LinearLayout`. Для добавления элементов-представлений `View` внутрь других элементов или для их перемещения удобно использовать панель **Outline** (Обзор).

2. Добавьте элемент `TextView` под названием `TextViewTopTitie`. Присвойте его атрибуту `layout_width` значение `fill_parent`, а атрибуту `layout_height` — значение `wrap_content`. Присвойте атрибуту `text` соответствующий строковый ресурс, атрибуту `textColor` — ресурс желтого цвета, а атрибуту `textSize` — соответствующий ресурс размера.

3. Добавьте элемент-контейнер `TableLayout` под названием `TabeLayout01`. Присвойте атрибуту `layout_width` этого элемента значение `fill_parent`, а атрибуту `layout_height` — значение `wrap_content`. Также присвойте атрибуту `stretchColumns` значение *, которое позволяет при необходимости растягивать любой столбец, чтобы заполнить свободное пространство экрана.

4. Добавьте элемент `TableRow` внутрь добавленного элемента-контейнера `TableLayout`. Добавьте два элемента `ImageView` внутрь элемента `TableRow`. Для первого элемента `ImageView` присвойте атрибуту src значение `@drawable/splashl`, которое соответствует графическому ресурсу **splas1.png.** Добавьте второй элемент `ImageView` и присвойте его атрибуту src значение, соответствующее графическому ресурсу **splash2.png.**

5. Повторите шаг 4, создав второй элемент `TableRow`. Опять же добавьте два элемента `ImageView` для графических ресурсов **splash3.png** и **splash4. png.**

6. По аналогии с шагом 2 добавьте еще один элемент `TextView` под названием `TextViewBottomTitle` внутрь родительского элемента-контейнера `LinearLayout`. Присвойте его атрибуту `layout_width` значение `fill__parent`, а атрибуту `layout_height` — значение `wrap_content`. Присвойте атрибуту text соответствующий строковый ресурс, атрибуту `textColor` — ресурс желтого цвета, а атрибуту `textSize` — соответствующий ресурс размера.

7. Для отображения информации о версии последний элемент `TextView` под названием `TextViewBottomVersion`. Присвойте его атрибутам `layout_width` и `layout_height` значение `fill_parent`. Присвойте атрибуту `text` соответствующий строковый ресурс, атрибуту `textColor` – ресурс сероватого цвета, а атрибуту `textSize` – соответствующий ресурс размера. Также присвойте атрибуту background соответствующий ресурс цвета (темно-синий) и атрибуту `lineSpacingExtra` – соответствующий ресурс размера, определяющий расстояние между строками.

8. Настройте значения атрибутов `layout_gravity` и `gravity` добавленных элементов пользовательского интерфейса, чтобы макет выглядел приемлемо в редакторе ресурсов среды разработки Eclipse.

Теперь сохраните файл макета **splash.xml** и запустите приложение «Been There, Done That!» на эмуляторе Android. Экран-заставка должен выглядеть так, как показано на рис. 7.3.

На этом можно было бы остановиться, но в вашей заставке явно не хватает какой-то живинки. Кроме того, нужно добавить переход между экраном-заставкой и основным навигационным экраном.

Рис. 7.3. Экран-заставка приложения «Been There, Done That!»

ИСПОЛЬЗОВАНИЕ АНИМАЦИИ

Отличный способ придать экспрессии вашему экрану-заставке — добавить движение. Платформа Android поддерживает четыре типа анимации:

- **Анимированные GIF-изображения.** Это графические файлы, состоящие из нескольких кадров.

- **Покадровая анимация.** Инструментарий Android SDK предоставляет механизм покадровой анимации, схожий с механизмом создания анимированных GIF-изображений, когда разработчик добавляет отдельные графические кадры и настраивает переходы между ними (дополнительную информацию можно найти в описании класса AnimationDrawable).

- **Анимация движения.** Анимация движения — это простой и универсальный способ определения анимационных действий, которые в дальнейшем могут быть применены к любому элементу-представлению или элементу-контейнеру.

- **Интерфейс OpenGL ES.** Интерфейс OpenGL ES платформы Android предоставляет широкие возможности для трехмерного рисования, создания анимаций, настройки освещения и наложения текстур.

Для вашего приложения лучше всего подходит анимация движения. Платформа Android поддерживает анимацию движения с применением альфа- канала, вращения, масштабирования и перемещения. Вы можете создавать наборы анимационных действий, которые могут выполняться одновременно, последовательно или в заданное время. Таким образом, анимация движения — отличный выбор для экрана-заставки.

Создать анимацию движения можно либо программным путем, либо добавив анимационный ресурс в каталог **/res/anim.** Для каждой анимационной последовательности необходимо создавать отдельный XML-файл, но при этом созданная анимация может применяться к любому числу элементов- представлений View.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...
Вы можете использовать анимационные клипы, доступные в классе `android.R.anim`.

Добавление анимационных ресурсов

Для заставки вам потребуется создать три пользовательских анимационных последовательности в виде XML-файлов и сохранить их в каталоге ресурсов **/res/anim: fade_in.xml, fade_in2.xml** и **custom_anim.xml.**

Первая анимация, представленная файлом **fade_in.xml,** плавно увеличивает непрозрачность (альфа-канал) целевого элемента от значения 0 (прозрачный) до значения 1 (непрозрачный) в течение 2500 миллисекунд, или 2,5 секунд. В среде Eclipse нет редактора анимации. Содержимое XML-файла **fade_in.xml** выглядит следующим образом:

```
<?xml version="1.0" encoding="utf-8" ?>
<set
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <alpha
        android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="2500">
    </alpha>
</set>
```

Вы можете применить эту анимацию к верхнему элементу `TextView` с названием вашего приложения.

Теперь создайте анимацию **fade_in2.xml.** Она делает то же самое, что и **fade_in.xml,** за исключением того, что атрибуту startOffset будет присвоено значение 2500 миллисекунд. Это значит, что ее продолжительность будет составлять 5 секунд: после задержки, равной 2,5 секунды, начинается плавное отображение элемента-представления в течение 2,5 секунды. Поскольку для заставки пяти секунд вполне достаточно, но завершении анимации **fade_in2.xml** начинается переход к экрану с основным меню.

Наконец, вам потребуется некая забавная анимация для изображений внутри элемента-контейнера `TableLayout`**.** В данном случае ваш набор анимационных действий включает несколько операций, выполняющихся одновременно: вращение, масштабирование и изменение альфа-прозрачности, В результате целевой элемент-представление `View` будет совершать полный оборот вокруг своею центра. Содержимое файла **custom_anim.xml** выглядит следующим образом:

```
<?xml version="1.0" encoding="utf-8" ?>
<set
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:shareInterpolator="false">
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="2000" />
    <alpha
        android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="2000">
    </alpha>
    <scale
        android:pivotX="50%"
        android:pivotY="50%"
        android:fromXScale=".1"
        android:fromYScale=".1"
        android:toXScale="1.0"
        android:toYScale="1.0"
        android:duration="2000" />
</set>
```

Как видим, поворот на 360 градусов занимает 2 секунды, при этом вращение осуществляется вокруг центра элемента-представления. Операция по изменению альфа-прозрачности должна быть вам знакома; плавное изменение прозрачности происходит в течение тех же 2 секунд. Наконец, операция по изменению масштаба от 10% до 100% также происходит в течение двухсекундного интервала. Воспроизведение всей анимации занимает 2 секунды.

Сохранив все три файла анимации, вы можете приступать к применению анимаций к конкретным элементам-представлениям.

Анимация конкретных элементов-представлений

Применение и управление анимациями должно производиться программным путем. Не забывайте, что ресурсоемкие операции должны прекращаться, если выполнение приложения приостанавливается по какой-либо причине. Воспроизведение анимации может быть продолжено после того, как приложение снопа перейдет в активное состояние.

Давайте начнем с простейшего случая: применим анимацию **fade_in.xml** к элементу-представлению `TextView` под названием `TextviewTopTitle`, используемого для отображения названия приложения. Все, что вам нужно, - это получить экземпляр вашего элемента `TextView` в методе `onCreate()` класса `QuizSplashActivity`, загрузить анимационный ресурс в объект Animation и вызвать метод `startAnimation()` элемента-представления `TextView`:

```
TextView logo1 = (TextView) findViewById(R.id.TextViewTopTitle);
Animation fade1 = AnimationUtils.loadAnimation(this, R.anim.fade_in);
logo1.startAnimation(fade1);
```

Если воспроизведение анимации должно быть остановлено — например, в методе `onPause()` соответствующей деятельности, — вы просто вызываете метод

clearAnimation(). Например, следующий метод onPause() демонстрирует этот подход для угловых логотипов:

```
@Override
protected void onPause() {
    super.onPause();
    // Stop the animation
    TextView logo1 = (TextView) findViewById(R.id.TextViewTopTitle);
    logo1.clearAnimation();
    TextView logo2 = (TextView)
    findViewById(R.id.TextViewBottomTitle);
    logo2.clearAnimation();
    // ... stop other animations
}
```

Анимация всех элементов в элементе-контейнере

Помимо возможности применения анимаций к отдельным элементам View, вы также можете применить анимации к каждому элементу View, содержащемуся в элементе-контейнере (например, в элементе-контейнере TableLayout и к каждому элементу TableRow), используя класс LayoutAnimationController.

Чтобы анимировать элементы-представления View с использованием данного подхода, вы должны загрузить соответствующую анимацию, создать экземпляр класса LayoutAnimationController, произвести необходимые настройки этого экземпляра и затем вызвать метод setLayoutAnimation() элемента-контейнера. Например, следующий код загружает анимацию custum_anim, создает экземпляр класса LayoutAnimationController и затем применяет эту анимацию к каждому элементу TableRow в элемент-контейнере TableLayout:

```
Animation spinin = AnimationUtils.loadAnimation(this,
R.anim.custom_anim);
LayoutAnimationController controller =
    new LayoutAnimationController(spinin);
TableLayout table = (TableLayout) findViewById(R.id.TableLayout01);
for (int i = 0; i < table.getChildCount(); i++) {
    TableRow row = (TableRow) table.getChildAt(i);
    row.setLayoutAnimation(controller);
}
```

В данном случае нет необходимости вызывать метод startAnimation(), поскольку для вас это сделает экземпляр класса LayoutAnimationController. При использовании данного подхода анимация применяется к каждому дочернему элементу-представлению, однако воспроизведение каждой анимации начинается в разное время. (По умолчанию задержка составляет 50% от длительности анимации — в данном случае эта задержка равна 1 секунде.) Это создает красивый эффект, когда каждый из элементов-предстаатений ImageView поочередно делает полный оборот вокруг своего центра.

Остановка анимаций, реализуемых при помощи экземпляра класса LayoutAnimationController, ничем не отличается от остановки отдельных анимаций: для этого применяется метод clearAnimation(). Дополнительные строки

кода, которые потребуется добавить в существующий метод `onPause()`, представлены ниже:

```
TableLayout table = (TableLayout) findViewById(R.id.TableLayout01);
for (int i = 0; i < table.getChildCount(); i++) {
    TableRow row = (TableRow) table.getChildAt(i);
    row.clearAnimation();
}
```

Обработка событий жизненного цикла анимаций

Теперь, когда вы не можете нарадоваться на ваши анимации, осталось создать переход между деятельностями `QuizSplashActivity` и `QuizMenuActivity` по завершении воспроизведения анимаций. Для этого вы создадите новый объект типа Intent, который будет создавать экземпляр класса QuizMenuActivity и вызывать метод startActivity(). Вы также должны вызывать метод finish() класса QuizMenuActivity, поскольку хранить эту деятельность в стеке нет необходимости (если вы не хотите, чтобы при нажатии на кнопку Back пользователь возвращался к экрану-заставке).

Из всех ваших анимаций самая длинная — fade_in2, продолжительностью 5 секунд. Таким образом, момент завершения воспроизведения этой анимации можно использовать in я запуска процесса перехода к экрану с основным меню. Для этого нужно создать экземпляр класса `AnimationListener`, который имеет методы обработки событий для жизненного цикла анимации: начало, конец и повторение. В данном случае нас интересует только метод `onAnimationEnd()`. Ниже представлен код, который создает экземпляр класса `AnimationListener` и реализует метод обработки событий `onAnimationEnd()`:

```
Animation fade2 = AnimationUtils.loadAnimation(this, R.anim.fade_in2);
fade2.setAnimationListener(new AnimationListener() {
    public void onAnimationEnd(Animation animation) {
        startActivity(new Intent(QuizSplashActivity.this,
            QuizMenuActivity.class));
        QuizSplashActivity.this.finish();
    }
});
```

Теперь, если снова запустить приложение «Been There, Done That!» на эмуляторе или на мобильном телефоне, вы увидите привлекательную анимацию на экране-заставке. После завершения воспроизведения анимации происходит плавный переход к экрану с основным меню, реализацией которого вы займетесь в следующем часе.

ИТОГИ

Примите наши поздравления! Вы завершили работу над вашим первым экраном опроса-викторины «Been There, Done That!». В этом часе вы разработали дизайн экрана-заставки, а также определили подходящий дизайн- макет и элементы представления `View`, необходимые для реализации данного дизайна. После этого вы добавили необходимые ресурсы и внесли изменения в файл макета **splash.xml.** Наконец, вы добавили несколько анимаций движения на заставку и затем реализовали переход между деятельностями QuizSplashActivity и QuizManuActivity.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Насколько хорошо платформа Android справляется с анимациями?

Ответ: Android обладает приемлемой производительностью при работе с анимациями. Тем не менее вы можете очень легко переполнить экран анимациями и другими элементами пользовательского интерфейса. Например, если бы вы поместили в центр экрана элемент-представление VideoView вместе с остальными анимациями, проблемы с производительностью были бы весьма заметны. Всегда проверяйте выполнение различных операций, в том числе анимаций, на реальном устройстве, чтобы гарантировать, что ваша реализация приемлема.

Вопрос: Почему в цикле вы обращаетесь к каждому дочернему элементу в элементе-контейнере `TableLayout` вместо того, чтобы обращаться к каждому элементу `TableRow(R. id. TableRowOl и R. id.TableRow02)` по его имени?

Ответ: Обращаться к каждому элементу `TableRow` по его имени совершенно допустимо при условии, что это имя существует. Вы сможете воспользоваться преимуществом данного итеративного подхода в дальнейшем, когда будете адаптировать ваш проект для различных ориентаций экрана. На данный момент экран-заставка отображается хорошо только в портретной ориентации.

Вопрос: Что произойдет, если применить экземпляр класса `Layout AnimationController` к элементу-контейнеру `TableLayout`**,** а не к каждому элементу `TableRow`?

Ответ: Если применить экземпляр класса `LayoutAnimationController` к элементу-контейнеру `TableLayout`**,** полный оборот вокруг своего центра совершит каждый элемент `TableRow`**,** а не каждый элемент `ImageView`**.** Это будет совершенно другой, менее привлекательный, эффект.

ПРАКТИКУМ
Контрольные вопросы

1. Не существует способа остановить воспроизведение анимации. Верно ли этой?

2. Какие типы операций поддерживаются анимациями движения?

   A. Альфа-прозрачность, движение и трехмерное вращение.
   B. Альфа-прозрачность, масштабирование, вращение и переходы.
   C. Танцы, пение и веселье.

3. Элемент-контейнер `LinearLayout` может быть использован для размещения всех дочерних элементов-представлений View друг за другом (по вертикали). Верно ли это?

Час 8. РЕАЛИЗАЦИЯ ЭКРАНА С ОСНОВНЫМ МЕНЮ

Вопросы, рассматриваемые в этом часе:

- **разработка дизайна экрана с основным меню;**
- **реализация макета экрана с основным меню;**
- **работа с элементами** `ListView`**;**
- **работа с другими типами меню.**

В этом часе вы узнаете о различных типах меню, доступных в операционной системе Android. Сначала вы создадите экран с основным меню для приложения «Been There, Done That!», используя новые элементы `ListView` и `RelativeLayout`**.** Затем вы познакомитесь с другими экранами, для которых могут подойти особые типы меню, например меню опций.

РАЗРАБОТКА ДИЗАЙНА ЭКРАНА С ОСНОВНЫМ МЕНЮ

Для создания экрана с основным меню вам понадобится сделать набросок того, как должен выглядеть этот экран. Если взглянуть на требования к этому экрану, вы увидите, что он должен обеспечивать основную навигацию по остальной части приложения. Пользователям доступны четыре различных варианта: они могут сыграть в игру, ознакомиться с се правилами, настроить параметры или просмотреть таблицу результатов. На рис. 8.1 представлен эскиз дизайна экрана с основным меню.

Рис. 8.1. Эскиз дизайна экрана с основным меню
для приложения «Been There, Done That!»

Существует ряд различных подходов для реализации экрана с основным меню. Например, можно создать кнопку для каждого пункта меню, прослушивать события нажатий и переправлять пользователя на соответствующий экран. Тем не менее по мере увеличения количества пунктов меню масштабируемость данного подхода будет ухудшатся. Таким образом, более приемлемо использование меню в виде списка, реализованного при помощи элемента `ListView`. В этом случае, если количество элементов в списке выйдет за границы экрана, в нашем распоряжении окажется встроенная возможность прокрутки содержимого списка.

Навигационный экран нужно снабдить какими-то «бантиками», делающими его привлекательным. Сначала вы будете использовать стандартное поведение для каждого элемента

макета, а затем добавите специальный стиль к этим элементам. Например, можно добавить красивое фоновое изображение для всего экрана и отдельное для активного элемента списка.

Наконец, вы добавите бизнес-логику для элемента `ListView`, чтобы гарантировать, что при нажатии на элемент списка будет открыт соответствующий экран. Это позволит пользователям легко переходить на другие экраны, которые вам потребуется реализовать для приложения «Been There, Done That!».

Определение требований к экрану с основным меню

Теперь, когда вы знаете, как будет выглядеть ваш экран с основным меню, мы необходимо преобразовать эскиз дизайна в соответствующий дизайн- макет Вам потребуется обновить файл макета **/res/layout/menu.xml.** который используется в деятельности `QuizMenuActivity`. Для макета экрана с основным меню вам потребуется заголовок, за которым расположится элемент `LisiView` и затем элемент `ImageView`.

Создание заголовка экрана при помощи элемента управления RelativeLayout

Очевидно, что для отображения заголовка экрана можно использовать элемент `TextView`. Но для красоты можно добавить изображения по обе стороны от элемента `TextView`, не так ли? Самое время воспользоваться элементом-контейнером `RelativeLayout`, который позволяет расположить каждый дочерний элемент-представление в определенной позиции относительно родительского элемента-контейнера или других дочерних элементов-представлений. Таким образом, вы можете легко создать заголовок экрана при помощи элемента-контейнера `RelativeLayout` и трех дочерних элементов-представлений:

- элемента `ImageView`, выровненного по левому верхнему углу родительского элемента-контейнера;
- элемента `TextView`, выровненного по центру верхнего края родительского элемента-контейнера;
- элемента `ImageView`, выровненного по правому верхнему углу родительского элемента-контейнера.

Добавление элемента ListView

Далее вы должны добавить в макет элемент `ListView`. Элемент `ListView` просто исполняет роль контейнера, который будет содержать список элементов `View`. По умолчанию `ListView` используется для размещения элементов `TextView`, однако элементы `ListView` могут содержать множество разнообразных элементов `View`.

Элемент `ListView`, содержащий элементы `TextView`, отлично подойдет для данного примера. Чтобы переопределить стандартное поведение для каждого элемента `TextView`, содержащегося в элементе `ListView`, вас потребуется определить ресурс макета, который будет служить шаблоном для каждого элемента `TextView` внутри элемента

ListView. Также вы можете сделать ваше меню более интересным, определив собственный разделитель и селектор для элемента ListView.

Завершающие штрихи для макета экрана с основным меню

Чтобы полностью завершить разработку макета экрана с основным меню, вы добавите элемент управления ImageView, расположив его за элементом управления ListView. Как и раньше, все элементы экрана необходимо поместить в элемент-контейнер LinearLayout с вертикальной ориентацией при этом элементы RelativeLayout, ListView и ImageView должны следовать друг за другом по вертикали. На рис. 8.2 проставлен дизайн экрана с основным меню.

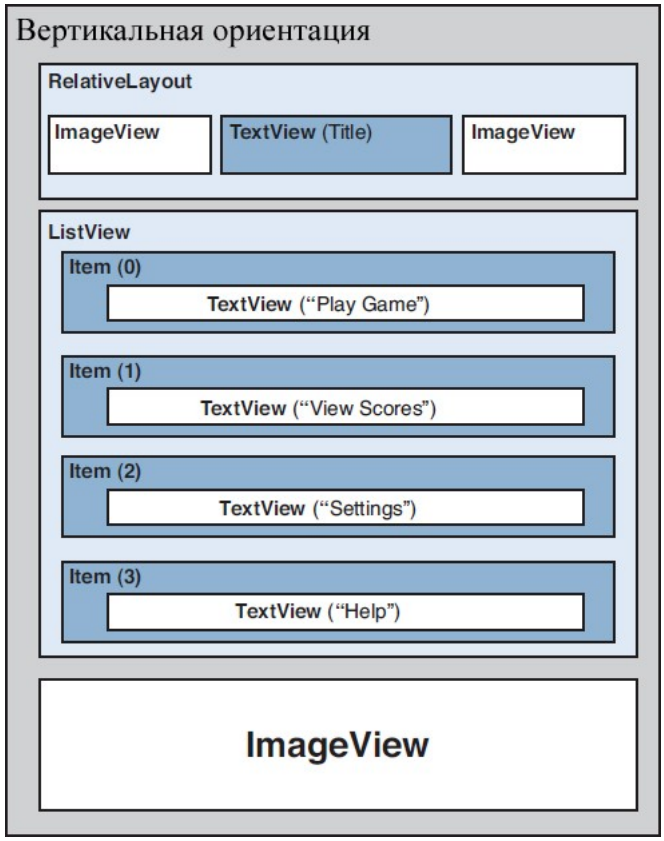

Рис. 8.2. Дизайн экрана с основным меню для приложения «Been There, Done That!»

РЕАЛИЗАЦИЯ МАКЕТА ЭКРАНА С ОСНОВНЫМ МЕНЮ

Реализацию экрана с основным меню вы начнете с добавления новых ресурсов в проект. После этот вам потребуется обновить файл макета **menu**, **xml**, чтобы реализовать разработанный дизайн-макет экрана с основным меню.

Добавление новых ресурсов в проект

Теперь, когда у вас есть дизайн-макет экрана, вам нужно создать графические ресурсы, строковые ресурсы, ресурсы цветов и размеров, которые будут использоваться в макете экрана с основным меню.

Сначала вы добавите четыре новых графических ресурса в каталог **/res/drawable: bkgrnd.jpg, divider.png, half.png** и **textured.png.** В качестве фонового изображения для элемента-контейнера `LinearLayout` будет использован файл **bkgrnd.jpg.** В качестве разделителя и селектора для элемента `ListView` будут использованы файлы **divider.png** и **textured.png** соответственно. Для элемента `ImageView`**,** расположенного в нижней части экрана, будет использован файл **half.png.**

После этого вы добавите несколько новых и измените несколько существующих строковых ресурсов в файле **/res/values/strings.xml,** которые будут использованы для каждого пункта меню и для элемента `TextView`, представляющего заголовок экрана.
Наконец, вы обновите ресурсы цветов в файле **/res/menu/colors.xml,** чтобы определить цвета, которые будут использованы для заголовка экрана, отображаемого при помощи элемента `TextView`, а также для элементов `TextView`, расположенных внутри элемента `ListView`. Кроме того, вам также понадобится обновить ресурсы в файле **/res/values/dimens.xml,** чтобы определить размеры для текста заголовка и для текста, отображаемого при помощи элемента `ListView`**.**

В качестве значений конкретных ресурсов вы можете либо использовать значения, представленные в этой книге, либо указать свои собственные.
Сохранив файлы ресурсов, вы можете приступать к их использованию в макетах экрана с основным меню.

Обновление макетов экрана с основным меню

Возможно, вы уже заметили, что для экрана с основным меню используется множество файлов с ресурсами макета. Основной файл макета — **menu, xml** — определяет макет для всего экрана. Кроме того, вы также должны создать новый файл макета шаблона для каждого пункта в вашем элементе `ListView`**.**

ОБНОВЛЕНИЕ ОСНОВНОГО МАКЕТА

Как и раньше, откройте редактор ресурсов в среде разработки Eclipse и удалите все существующие элементы из файла макета **menu.xml**. Затем выполните следующие шаги, чтобы создать желаемый макет экрана на основе
заранее подготовленного дизайн-макета.

1. Добавьте новый элемент-контейнер `LinearLayout` и присвойте его атрибуту `background` значение `@drawable/bkgrnd`**.** Все остальные элементы будут размещаться внутри данного элемента-контейнера `LinearLayout`**.**

2. Добавьте элемент-контейнер `RelativeLayout`**.** Присвойте его атрибуту `layout_width` значение `wrap_content`**,** а атрибуту `layout_height` — значение `wrap_content`.

3. Добавьте элемент `ImageView` внутрь элемента-контейнера `RelativeLayout`**.** Атрибутам `layout_alignParentLeft` и `layout_alignParentTop` этого элемента `ImageView` присвойте значение `true`**.** Атрибуту `src` присвойте значение `@drawable/quizicon`**.**

4. Добавьте элемент `TextView`, который будет использоваться для отображения заголовка экрана, внутрь элемента-контейнера `RelativeLayout`. Атрибутам `text`, `textSize` и `textColor` этого элемента `TextView` присвойте названия соответствующих ресурсов, которые вы создали чуть ранее. Затем присвойте атрибутам `layout_centerHorizontal` и `layout_ alignParentTop` значение `true`.

ЗНАЕТЕ ЛИ ВЫ. ЧТО...

Чтобы «оживить» текст, отображаемый при помощи элемента `TextView`, вы можете использовать атрибуты для создания эффекта тени, включая атрибуты `shadowColor`, `shadowDx`, `shadowDy` и `shadowRadius`.

5. Завершите работу с элементом-контейнером `RelativeLayout`, добавив внутрь него еще один элемент `ImageView`. Атрибутам `layout_align ParentRight` и `layout_alignParentTop` этого элемента `ImageView` присвойте значение `true`. Атрибуту `src` присвойте значение `@drawable/quizicon`.

6. Теперь мы должны добавить второй элемент-контейнер `RelativeLayout`, который будет использоваться для размещения элементов `ListView` и `ImageView`. Начните с добавления нового элемента-контейнера `RelativeLayout` за пределами элемента-контейнера `RelativeLayout`, с которым мы работали до недавнего времени, но при этом новый элемент- контейнер `RelativeLayout` должен размещаться внутри существующего элемента-контейнера `LinearLayout`.

7. Теперь добавьте последний элемент `ImageView` внутрь только что добавленного элемента-контейнера `RelativeLayout`. Присвойте его атрибуту `src` значение `@drawable/half`, а атрибутам `layout_width` и `layout_height` — значение `fill_parent`, чтобы гарантировать, что этот элемент займет все свободное пространство в нижней части экрана. Также присвойте его атрибуту `layout_alignParentBottom` значение true, а атрибуту `scaleType` – значение `fitEnd`, чтобы после масштабирования этот элемент управления по-прежнему оставался внизу экрана.

8. Теперь добавьте элемент `ListView` с именем `ListView_Menu` непосредственно под элементом `ImageView`. Присвойте его атрибуту `layout_width` значение `fill_parent`, а атрибуту `layout_height` – значение `wrap_content`. Также присвойте атрибуту `layout_alignTop` значение `true`. Этот атрибут позволяет сделать так, чтобы элемент отображался над изображением.

После этого сохраните файл макета **menu.xml**.

ВНИМАНИЕ!

Редактор ресурсов среды разработки Eclipse не отображает элементы `ListView` в режиме дизайна. Чтобы увидеть элемент `ListView`, используйте эмулятор Android. В данном случае редактор ресурсов не отражает реальный вид приложения.

Добавление макета шаблона элемента ListView

Теперь вам нужно создать новый файл макета **/res/layout/menu_item.xml,** который будет служить шаблоном для пунктов вашего элемента `ListView`.
В данном случае файл макета **menu_item.xml** будет содержать элемент `TextView`**.**

Для элемента `TextView` настраиваются все типовые атрибуты, за исключением одного атрибута — атрибута, который представляет текстовое значение этого элемента. Атрибут text будет автоматически заполняться элементом `ListView`. Пока вы можете настроить значения атрибутов `textColor` и `textsize` элемента `TextView`, используя ресурсы цветов и размеров, созданных ранее.

Содержимое файла **menu_item.xml** выглядит следующим образом:

```
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:textSize="@dimen/menu_item_size"
    android:text="test string"
    android:layout_gravity="center_horizontal"
    android:layout_height="wrap_content"
    android:shadowRadius="5"
    android:gravity="center"
    android:textColor="@color/menu_color"
    android:shadowColor="@color/menu_glow"
    android:shadowDy="3"
    android:shadowDx="3" />
```

ЗНАЕТЕ ЛИ ВЫ. ЧТО...
Чтобы увидеть, как выглядят атрибуты элемента `TextView` в редакторе ресурсов среды разработки Eclipse, можно присвоить атрибуту text какую-либо тестовую строку. В дальнейшем эта строка будет перезаписываться элементом `ListView` программным путем.

Теперь сохраните файл макета **menu_item.xml**.

РАБОТА С ЭЛЕМЕНТОМ LISTVIEW

Теперь вы должны открыть файл **QuizMenuActivity.java**. Вам нужно доработать элемент `ListView`. Во-первых, вам потребуется заполнить элемент `ListView` содержимым, а затем вам потребуется прослушивать события нажатий для отдельных пунктов элемента `ListView` и переправлять пользователя на соответствующий экран.

Заполнение элемента ListView

Элементу `ListView` требуется контент. Элементы `ListView` могут быть заполнены содержимым из множества источников данных, включая массивы и базы данных, путем

использования адаптеров данных. В данном случае у вас есть список, состоящий из четырех элементов, поэтому простой массив значений типа String подходит в качестве источника данных для вашего элемента `ListView`.

Сначала вы должны получить экземпляр элемента `ListView` сразу после вызова метода `setContentview()` в методе `onCreate()` вашей деятельности. Чтобы заполнить элемент `ListView`, вы должны получить экземпляр этого элемента, используя метод `findViewById()`, как показано в следующем коде:

```
ListView menuList = (ListView) findViewById(R.id.ListView_Menu);
```

Далее вам нужно определить значения типа string — они будут использованы для создания элементов `TextView`, которые будут добавлены в элемент `ListView`. В данном случае вы загрузите четыре строковых ресурса, представляющих пункты меню:

```
String[] items = { getResources().getString(R.string.menu_item_play),
      getResources().getString(R.string.menu_item_scores),
      getResources().getString(R.string.menu_item_settings),
      getResources().getString(R.string.menu_item_help) };
```

Теперь, когда вы получили экземпляр элемента `ListView` и определили данные, которые должны быть помещены в этот элемент, вам потребуется адаптер данных, чтобы сопоставить данные с созданным макетом шаблона (**menu_item.xml**). Выбор адаптера зависит от типа используемых данных. В данном случае вы будете использовать класс `ArrayAdapter`:

```
ArrayAdapter<String> adapt = new ArrayAdapter<String>(this,
    R.layout.menu_item, items);
```

Теперь вы должны сделать так, чтобы элемент `ListView` использовал этот адаптер:

```
menuList.setAdapter(adapt);
```

После этого вы можете сохранить файл **QuizMenuActivity.java** и запустить приложение «Been There, Done That!» на эмуляторе Android. После экрана- заставки появится экран с основным меню, который должен выглядеть примерно как на рис. 8.3.

Рис. 8.3. Экран с основным меню приложения «Been There, Done That!»

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Если вы устали просматривать заставку при каждом запуске приложения, просто измените файл **AndroidManifest.xml**, чтобы по умолчанию пускалась деятельность `QuizMenuActivity`, пока вы не завершите отладку.

Как видите, экран с основным меню начинает приобретать форму. Тем не менее, нажатие по пунктам меню пока не приводят к желаемым результатам.

Прослушивание событий элемента ListView

Вы должны прослушивать и реагировать на определенные события, генерируемые элементом `ListView`. И хотя этот элемент может генерировать несколько различных событий, нас больше всего интересует событие, которое возникает в тот момент, когда пользователь нажимает на один из пунктов элемента `ListView`.

Чтобы прослушивать события, возникающие при нажатии по одному из пунктов элемента `ListView`, вам нужно использовать метод `setOnltemClickListener()` элемента `ListView`. В частности, необходимо реализовать метод `onltemClick()`, метод класса `AdapterView.OnltemClickListener`, как показано ниже:

```
menuList.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
    public void onItemClick(AdapterView<?> parent, View itemClicked,
        int position, long id) {
        TextView textView = (TextView) itemClicked;
        String strText = textView.getText().toString();
        if (strText.equalsIgnoreCase(getResources().getString(
               R.string.menu_item_play))) {
            // Launch the Game Activity
            startActivity(new Intent(QuizMenuActivity.this,
```

```
                QuizGameActivity.class));
        } else if (strText.equalsIgnoreCase(getResources().getString(
                R.string.menu_item_help))) {
            // Launch the Help Activity
            startActivity(new Intent(QuizMenuActivity.this,
                QuizHelpActivity.class));
        } else if (strText.equalsIgnoreCase(getResources().getString(
                R.string.menu_item_settings))) {
            // Launch the Settings Activity
            startActivity(new Intent(QuizMenuActivity.this,
                QuizSettingsActivity.class));
        } else if (strText.equalsIgnoreCase(getResources().getString(
                R.string.menu_item_scores))) {
            // Launch the Scores Activity
            startActivity(new Intent(QuizMenuActivity.this,
                QuizScoresActivity.class));
        }
    }
});
```

В метод `onItemClick()` передается вся информацию, которая необходима для определения того, на какой пункт нажал пользователь. В данном случае один из наиболее простых способов — привести выбранный пользователем объект представления к элементу `TextView` (поскольку известно, что все пункты являются элементами `TextView`, хотя вы можете дополнительно проверить это, используя оператор `instanceof`) и просто извлечь соответствующий элемент `Text` и сопоставить его с соответствующим экраном. Другой способ определить, на каком пункте нажал пользователь, — проверить атрибут `id` элемента `View`.

Теперь, реализовав метод `OnitemClickListener()` и перезапустив приложение на эмуляторе, вы можете использовать основное меню для перехода между экранами в приложении «Been There, Done That!».

Настройка внешнего вида элемента ListView

Теперь вы готовы приступить к настройке достаточно безвкусного (учитывая тот факт, что мы создаем меню для игры) элемента `ListView`, указав собственное изображение-разделитель и изображение-селектор. `ListView` состоит из нескольких частей — верхней части, списка элементов и нижней части. По умолчанию элемент `ListView` отображается без верхней и нижней частей.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Если у вас есть экран с единственным элементом ListView, подумайте над возможностью использования класса ListActivity, который упрощает управление элементом ListView.

ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЬСКОГО РАЗДЕЛИТЕЛЯ

Разделитель элемента `ListView` отображается между отдельными пунктами данного элемента. В качестве значения атрибута `divider` можно указать либо ресурс цвета, либо графический ресурс. Если указан ресурс цвета, между элементами в списке будет отображаться горизонтальная линия (толщину этой линии можно настраивать). Если указан графический ресурс, между элементами и списке будет отображаться

соответствующее изображение. По умолчанию перед первым элементом списка и после последнего разделитель не отображается.

ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЬСКОГО СЕЛЕКТОРА

Селектор элемента `ListView` обозначает, какой элемент списка выбран в настоящий момент. Селектор элемента `ListView` устанавливается при помощи атрибута `listselector`. По умолчанию в качестве селектора элемента `ListView` используется полоса светло-оранжевого цвета.

Выполните самостоятельно

Чтобы установить разделитель для элемента `ListView`, просто откройте файл макета menu.xml и присвойте атрибуту `divider` элемента `ListView` значение `@drawable/divider`, представляющее графический ресурс (волнообразная линия желтого цвета), который вы добавили ранее.

Теперь установите пользовательский селектор для элемента `ListView`. Для этого просто откройте файл макета menu.xml и присвойте атрибуту `listSelector` элемента `ListView` значение `@drawable/textured`, представляющее графический ресурс (текстурированный ореол оранжевого цвета), который вы добавили ранее.

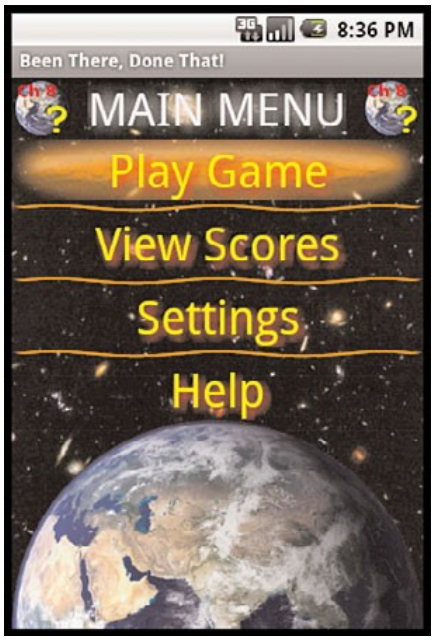

Рис. 8.4. Экран с основным меню приложения «Been There, Done That!», включая элемент ListView с измененным внешним видом

Если вы установите новый разделитель и селектор для элемента `ListView`, и перезапустите приложение «Been There, Попе That!» на эмуляторе, экран с основным меню будет выглядеть так, как показано на рис. 8.4.

РАБОТА С ДРУГИМИ ТИПАМИ МЕНЮ

Платформа Android предоставляет два других полезных типа меню:

- **Контекстное меню.** Отображается на экране, когда пользователь совершает продолжительное нажатие любого объекта типа `View`. Этот тип меню зачастую используется вместе с элементами `ListView`, содержащими похожие элементы, например названия музыкальных композиций в списке воспроизведения. Пользователь может сделать продолжительное нажатие по названию композиции, чтобы получить доступ к контекстному меню с такими командами, как, например, **Play** (Воспроизвести). **Delete** (Удалить) и **Add to Playlist** (Добавить в список воспроизведения), для этой конкретной композиции.

- **Меню опций.** Отображается на экране всякий раз, когда пользователь нажимает кнопку **Menu** на мобильном телефоне. Этот тип меню зачастую используется для предоставления пользователю возможности настройки параметров приложения и выполнения похожих операций.

Поскольку в этом часе мы работали над экраном навигации по приложению, давайте посмотрим, на каких экранах можно было бы использовать перечисленные типы меню в приложении «Been There, Done That!». Текущая архитектура приложения позволяет легко интегрировать меню опций в игровой экран, позволяя пользователю приостановить процесс игры, быстро получить доступ к экранам с настройками приложения или помощи и затем вернуться обратно к игровому экрану.

Добавление меню опций на игровой экран

Чтобы добавить меню опций на игровой экран, вам потребуется добавить специальный тип ресурса, называемый ресурсом меню. После этого вы можете изменить класс `QuizGameActivity`, чтобы активизировать меню опций и добавить обработку выбранных пунктов меню.

ДОБАВЛЕНИЕ РЕСУРСОВ МЕНЮ

Для вашего меню опций необходимо создать ресурс с определением меню в формате XML и сохранить его в каталоге ресурсов **/res/menu** в файле с именем **gameoptions.xml.**

Ресурс меню содержит тег `<menu>`, за которым следует несколько дочерних элементов `<item>`. Каждый элемент `<item>` представляет отдельный пункт меню и имеет ряд атрибутов. Ниже перечислены некоторые наиболее часто используемые атрибуты.

- id — этот атрибут позволяем легко идентифицирован, отдельные пункты меню.

- title — этот атрибут представляет строку, которая отображаем а для соответствующего пункта в меню опций.

- icon — этот атрибут используется для указания графического ресурса, представляющего значок для соответствующего пункта меню.

Ваше меню опций будет содержать всего два пункта; **Settings** (Настройки) и **Help** (Помощь). Таким образом, в содержимом файла вашего ресурса меню **gameoptions.xml** нет ничего сложного:

```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
       android:id="@+id/settings_menu_item"
       android:title="@string/menu_item_settings"
       android:icon="@android:drawable/ic_menu_preferences"></item>
    <item
       android:id="@+id/help_menu_item"
       android:title="@string/menu_item_help"
       android:icon="@android:drawable/ic_menu_help"></item>
</menu>
```

Для атрибута title каждого пункта меню опций вы указали те же строковые ресурсы, которые используются на экране с основным меню. Обратите внимание, что вместо добавления новых графических ресурсов для значков пунктов меню опций используются встроенные графические ресурсы из инструментария Android SDK, чтобы обеспечить согласованный внешний вид между различными приложениями.

ЗНАЕТЕ ЛИ ВЫ. ЧТО...

Вы можете использовать встроенные графические ресурсы, доступные в классе `android.R.drawable`, точно так же, как вы используете ресурсы, включенные в пакет вашего приложения. Если вы хотите увидеть, как выглядит каждый из этих встроенных ресурсов, обратитесь к содержимому инструментария Android SDK, установленного на вашем компьютере. В частности, найдите каталог **/platforms**, выберите соответствующую целевую платформу и откройте каталог **/data/res/drawable**.

ДОБАВЛЕНИЕ МЕНЮ ОПЦИЙ К ДЕЯТЕЛЬНОСТИ

Чтобы меню опций отображалось в тот момент, когда пользователь нажмет кнопку **Menu,** находясь на игровом экране, вы должны реализовать метод `onCreateOptionsMenu()` в классе `QuizGameActivity`**.** В частности, вы должны создать (загрузить) меню опции из ресурса меню и установить соответствующий объект типа Intent для каждого меню. Вот пример реализации метода `onCreateOptionsMenu()` для каждого класса `QuizGameActivity`:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    getMenuInflater().inflate(R.menu.gameoptions, menu);
    menu.findItem(R.id.help_menu_item).setIntent(
        new Intent(this, QuizHelpActivity.class));
    menu.findItem(R.id.settings_menu_item).setIntent(
        new Intent(this, QuizSettingsActivity.class));
    return true;
}
```

ОБРАБОТКА СОБЫТИЯ ВЫБОРА ПУНКТА МЕНЮ

Чтобы обрабатывать ситуацию, когда пользователь открывает меню опций и выбирает некоторый пункт меню, необходимо реализовать метод `onOptionsltemSelected()` нужной деятельности. Например, вы можете запустить соответствующую деятельность, получив интент из пункта меню, как показано ниже:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
super.onOptionsItemSelected(item);
startActivity(item.getIntent());
return true;
}
```

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Представленная реализация метода `onOptionsltemSelected()` работает как положено. С технической точки зрения, если ваше меню будет использоваться исключительно для запуска определенного интента, указанного посредством вызова метода `setlntent()`, реализовывать данный метод нет никакой необходимости. Тем не менее, если нужно добавить любую другую функциональность для каждого элемента `Menultem`, вам придется реализовать данный метод.

Итак, все готово: вы создали меню опций на игровом экране. Если вы сохраните класс и запустите приложение еще раз, вы увидите, что можно перейти на игровой экран, нажать кнопку **Menu** и использовать полнофункциональное меню опций (рис. 8.5).

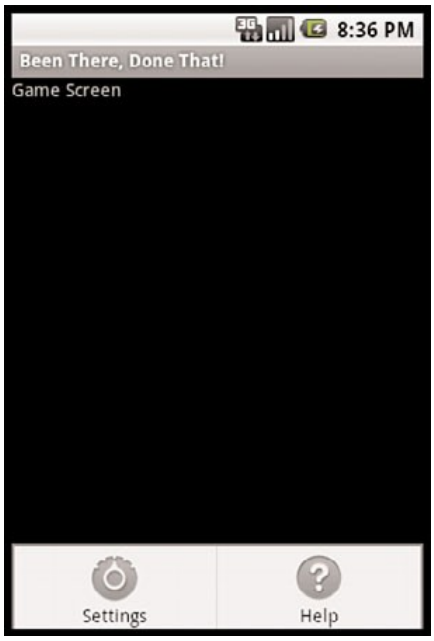

Рис. 8.5. Игровой экран приложения «Been There, Done That!» с меню опций

ИТОГИ

Вы проделали большую работу. Экран с основным меню теперь полностью функционален, и вы освоили ключевые навыки разработки Android- приложений, включая использование новых элементов-контейнеров, например `RelativeLayout`, а также использование элемента с широкими возможностями `ListView`. Вы также познакомились с другими типами навигации, доступными в операционной системе Android, и реализовали меню опций на игровом экране.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: В чем отличие между методом `setOnClickListener()` и методом `setOnltemClickListener()` элемента ListView?

Ответ: Метод `setOnClickListener()` позволяет прослушивать события нажатий по всей области элемента `ListView`**.** Метод `setOnltemClick Listener()` позволяет прослушивать события нажатий по конкретному элементу- представлению `View` внутри элемента `ListView`.

Вопрос: В созданном мной элементе `ListView` нет элемента, выбираемого по умолчанию. Как можно указать элемент, который будет использоваться в качестве элемента по умолчанию?

Ответ: Чтобы в элементе `ListView` по умолчанию подсвечивался определенный элемент списка, используйте метод `setSelection()`.

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Контекстные меню открываются при нажатии на кнопку **Menu**.

2. Какой механизм выступает в роли «связующего звена» между источником данных и элементом `ListView`?

   A. База данных.
   B. Интерполятор.
   C. Адаптер данных.

3. Какой элемент-контейнер лучше всего подходит для выравнивания дочерних элементов `View` относительно родительского элемента?

   A. `RelativeLayout`.
   B. `AbsoluteLayout`.
   C. `LinearLayout`.

4. Верно ли это? Использование класса `ListActivity` — это удобный способ построения экранов, которые представляют собой обычные элементы `ListView`.

Ответы

1. Неверно. При нажатии на кнопку Menu открываются меню опций. Контекстные меню открываются, совершает продолжительное нажатие по любому элементу `View`.

2. Для связывания источника данных с макетом шаблона, используемого элементом `ListView` для отображения каждого элемента списка, предназначен адаптер данных.

3. Элемент-контейнер `RelativeLayout` особенно удобно применять в тех случаях, когда его дочерние элементы view должны быть выровнены по верхнему, нижнему, левому или правому краю или по центру родительского элемента-контейнера. Элемент-контейнер `RelativeLayout` также может быть использован для позиционировании дочерних элементов `View` относительно друг друга внутри родительского элемента-контейнера.

4. Верно. Класс `ListActivity` упрощает работу с элементами `ListView`.

Упражнения

1. Познакомьтесь с другими приемниками данных, которые могут быть использованы при работе с элементом `ListView`. Измените элемент `ListView` таким образом, чтобы каждый элемент списка включал значок, отображаемый при помощи элемента-представления `ImageView`, и элемент-представление `TextView`. (Подсказка: реализуйте собственный интерфейс данных.)

2. Добавьте третий пункт в меню опций на игровом экране, позволив пользователю переходить на экран с результатами игры.

3. Измените элемент `LinearLayout` в файле макета **menu.xml**, добавив к нему анимацию, осуществляющую плавное отображение элементов, чтобы реализовать плавное появление всего экрана с основным меню.

Час 9. РАЗРАБОТКА ЭКРАНА С ИНСТРУКЦИЯМИ И ЭКРАНА С РЕЗУЛЬТАТАМИ

Вопросы, рассматриваемые в этом часе:

- **подготовка дизайна и разработка экрана с инструкциями;**
- **работа с файлами;**
- **подготовка дизайна и разработка экрана с результатами;**
- **подготовка дизайна экранов с вкладками;**
- **работа с XML-данными.**

В этом часе вы разработаете еще два экрана приложения «Been There, Done That!»: экран с инструкциями и экран с результатами. Сначала вы займетесь разработкой экрана с инструкциями, взяв за основу элемент `TextView` и текст, загружаемый из текстового файла, что позволит вам познакомиться с некоторыми классами для работы с файлами, доступными в инструментарии Android SDK. После этого вы подготовите дизайн и реализуете экран с результатами. Поскольку к экрану с результатами предъявляются более сложные требования, он идеальный кандидат для использования элемента `TabHost`, предоставляющего набор вкладок. Наконец, вы протестируете экран, осуществив разбор тестовых XML-данных с результатами и отобразив соответствующую информацию на каждой вкладке.

ПОДГОТОВКА ДИЗАЙНА ЭКРАНА С ИНСТРУКЦИЯМИ

Требования, предъявляемые к экрану с инструкциями, достаточно просты: этот экран должен отображать большой объем текста и иметь возможность прокрутки содержимого. На рис. 9.1 представлен эскиз дизайна экрана с инструкциями.

Вы хотите сделать так, чтобы экраны приложения обладали некоторым сходством. Таким образом, на экран с инструкциями можно добавить черты экрана меню навигации, например заголовок. Чтобы воплотить эскиз дизайна в соответствующем дизайн-макете, вам потребуется обновить файл макета **/res/layout/help.xml** и класс `QuizHelpActivity`.

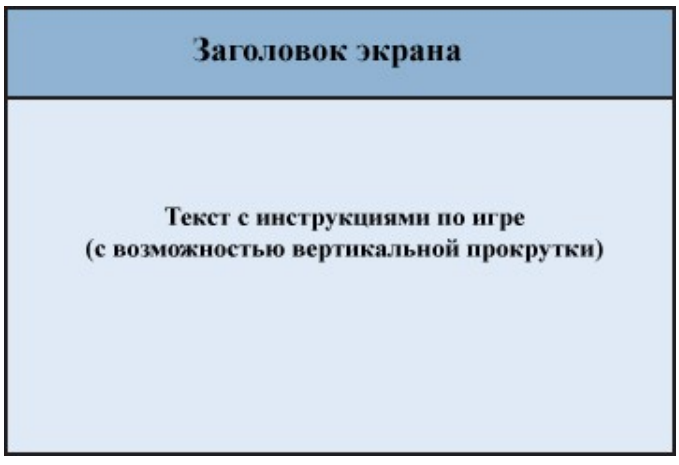

Рис. 9.1. Эскиз дизайна экрана с инструкциями

приложения «Been There, Done That!»

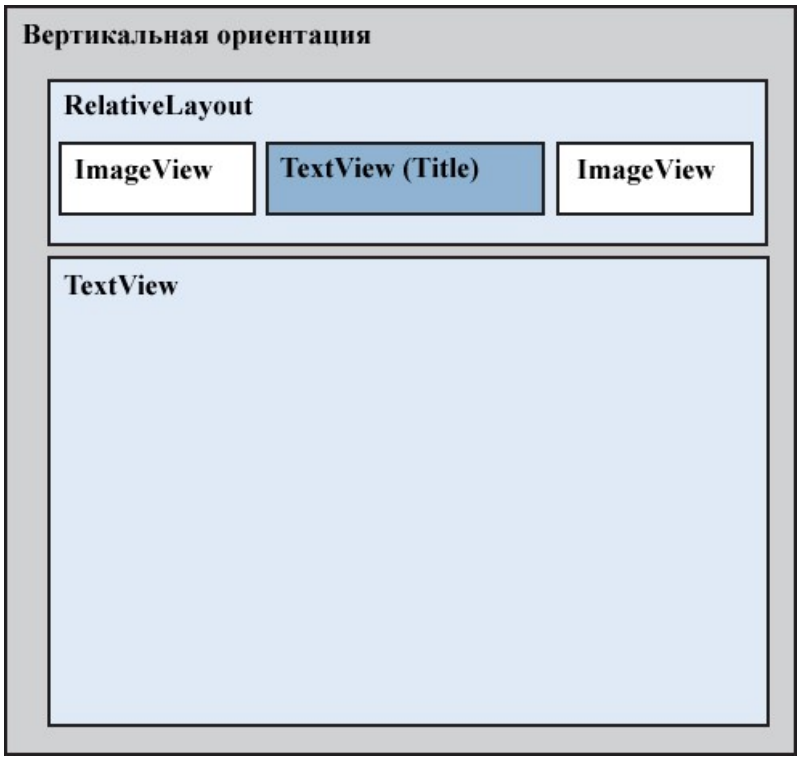

Рис 9.2. Дизайн-макет экрана с инструкциями приложения «Been There, Done That!»

Вы можете использовать такой же заголовок экрана, который применяется на экране с основным меню (реализованный при помощи элемента-контейнера `RelativeLayout`) и за которым будет следовать элемент `TextView` с возможностью прокрутки содержимого. На рис. 9.2 изображен дизайн макета экрана с инструкциями.

РЕАЛИЗАЦИЯ МАКЕТА ЭКРАНА С ИНСТРУКЦИЯМИ

Реализацию экрана с инструкциями вы начнете с добавлением новых ресурсов в проект. Затем вы обновите файл макета **help.xml**, чтобы отразить подготовленный дизайн-макет экрана с инструкциями.

Добавление новых ресурсов в проект

Помимо всех новых строковых ресурсов, ресурсов цветов и размеров, которые будут использованы в макете экрана с инструкциями, вам также потребуется добавить новый тип ресурса — ресурс файла с данными. В данном примере вы добавите текстовый файл **/res/raw/quizhelp.txt**, который содержит несколько абзацев текста с инструкциями для отображения в основном элементе `TextView` экрана с инструкциями.

ЗНАЕТЕ ЛИ ВЫ. ЧТО...

Вы также может добавлять большие блоки текста в качестве строковых ресурсов. Такой подход может оказаться полезным при локализации приложения. Использование строковых ресурсов также позволяет вам воспользоваться встроенной поддержкой тегов в стиле языка HTML. В данном примере мы остановили свой выбор на текстовом файле, чтобы продемонстрировать использование ресурсов файлов с данными.

Обновление макета экрана с инструкциями

Файл макета **help.xml** определяет пользовательский интерфейс экрана с инструкциями. Откройте редактор ресурсов среды разработки Eclipse и удалите все существующие элементы пользовательского интерфейса из макета. Далее выполните следующие шаги, чтобы создать желаемый макет на основе ранее подготовленного дизайн-макета экрана.

Добавьте новый элемент-контейнер `LinearLayout` и присвойте его атрибуту background значение `@drawable/bkgrnd`. Все остальные элементы будут добавляться внутрь этого элемента-контейнера `LinearLayout`.

Добавьте тот же заголовок, который вы использовали в файле макета **menu.xml**. Он должен включать элемент-контейнер `RelativeLayout`, содержащий два элемента `ImageView` и один элемент `TextView`. Атрибуту text элемента `TextView` присвойте строковый ресурс с именем `@string/help`, чтобы отобразить соответствующий заголовок экрана.

Добавьте новый элемент `TextView` с именем `TextView_HelpText` внутрь элемента-контейнера `LinearLayout`, но за пределами элемента-контейнера `RelativeLayout`. Присвойте его атрибутам layout_width и layout_height значение `fill_parent`.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Вы можете автоматически связать номера телефона, веб-сайты, адреса электронной почты и почтовые адреса, отображаемые в элементе `TextView`, с приложениями **Телефон**, **Браузер**, **Email** и **Карта операционной системы** Android, присвоив атрибуту `linksClickable` элемента `TextView` значение true, а атрибуту `autoLink` – значение `all`.

Внеся необходимые изменения, сохраните файл макета **help.xml**.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Чтобы установить полужирное или курсивное начертание для текста, отражаемого в элементе `TextView`, используйте атрибут `textStyle`.

РАБОТА С ФАЙЛАМИ

Теперь, когда файл макета **help.xml** готов, вам нужно обновить класс `QuizHelpActivity` так, чтобы он считывал содержимое файла **quizhelp.txt** и сохранял полученный текст в элементе `TextView` с именем `TextView_HelpText`.

Добавление ресурсов файлов с данными

Ресурсы файлов с данными, как, например, текстовый файл **quizhelp.txt.** могут быть добавлены в проект путем помещения в каталог **/raw.** находящийся внутри каталога с ресурсами проекта. Вы можете либо создать новые файлы внутри этой папки, либо перетащить существующие файлы в файловом менеджере, либо использовать любой другой способ, который вы привыкли использовать для добавления файлов в проекты Android в среде разработки Eclipse.

ЗНАЕТЕ ЛИ ВЫ. ЧТО...

Каждое Android-приложение имеет свой собственный закрытый каталог в файловой системе Android для хранения файлов приложения. Помимо широко известных классов `File` и `Stream`, вы можете обращаться к закрытым файлам и каталогам приложения при помощи следующих методов класса `Context`: `getFilesDir()`, `getDir()`, `openFileInput()`, `openFileOutput()`, `deleteFile()` и `getFileStreamPath()`.

Обращение к ресурсам файлов с данными

Платформа Android предоставляет множество классов, присущих языку Java, для выполнения операции ввода/вывода, включая классы для выполнения потоковых операций. Чтобы прочитать строковые данные из файла, используйте метод openRawResource(), как показано в следующем коде:

```
InputStream iFile = getResources().openRawResource(R.raw.quizhelp);
```

Обратите внимание, что, имея экземпляр класса `InputStream`, вы можете построчно или посимвольно прочесть данные из файла и получить результирующую строку. Сделать это можно несколькими способами. Создан соответствующую строку с текстом инструкций по игре, вы просто получаете ссылку на экземпляр элемента `TextView`, используя метод `findViewById()`, и присваиваете строку этому элементу, используя его метод `setText()`, как показано в следующем коде:

```
TextView helpText = (TextView) findViewById(R.id.TextView_HelpText);
String strFile = inputStreamToString(iFile);
helpText.setText(strFile);
```

КСТАТИ

В предыдущем коде используется вызов вспомогательного метода `inputStreamToString()`, реализованного в классе `QuizHelpActivity`, который доступен на диске. прилагаемом к данной книге. Этот метод просто считывает данные из экземпляра класса `InputStream` и возвращает объект типа string, представляющий содержимое экземпляра класса `InputStream`.

Теперь сохраните файл **QuizHelpActivity.java** и запустите приложение «Been There, Done That!» на эмуляторе Android. Когда заставка приложения исчезнет с экрана, выберите в основном меню пункт, соответствующий экрану с инструкциями. Экран с инструкциями изображен на рис. 9.3.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО

Теперь, когда на экране с инструкциями отображается нужная информация, вы можете захотеть улучшить его внешний вид, чтобы сделать этот экран более похожим на экран, изображенный на рис. 9.3. Начните с изменения размера шрифта текста с инструкциями в элементе `TextView`. Затем попробуйте изменить другие атрибуты, например, отступы, гарнитуру, цвета текста и т.д. В качестве подсказки вы можете взглянуть на значения атрибутов, указанных для элемента `TextView` в реализации, доступной на диске, прилагаемом к данной книге.

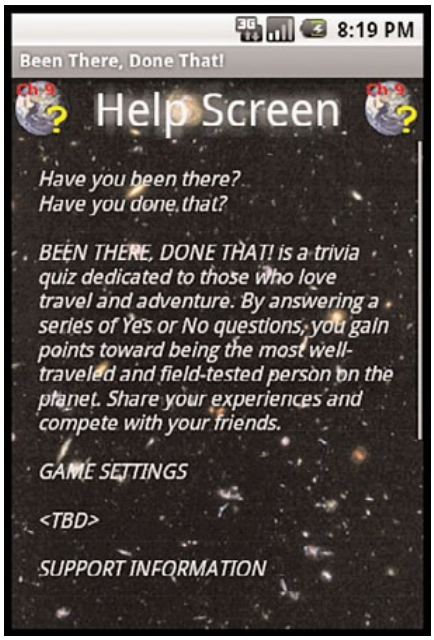

Рис. 9.3. Экран с инструкциями приложения «Been There, Done That!»

ПОДГОТОВКА ДИЗАЙНА ЭКРАНА С РЕЗУЛЬТАТАМИ

Теперь, когда вы создали экран с инструкциями, можно приступать к работе над другим экраном с похожей функциональностью — над экраном с результатами. В соответствии с требованиями, предъявляемыми к данному экрану, пользователь должен видеть несколько различных результатов. Существует два типа результатов: лучшие результаты игры и результаты друзей пользователя. Результаты обоих типов должны отображаться на одном экране. Каждый экран будет содержать имя пользователя, его результат и общий рейтинг.

Существует несколько вариантов реализации экрана с результатами. Например, вы могли бы использовать элементы `TextView` или `ListView` для отображения информации о результатах. Тем не менее вы работаете с экраном небольшого размера, и вы не хотите перегружать пользователя слишком большим количеством информации. Поскольку у вас есть два различных набора данных для отображения, идеальным решением для данного экрана было бы использование двух вкладок. На рис. 9.4 представлен эскиз экрана с результатами.

Рис. 9.4. Эскиз дизайна экрана с результатами приложения «Been There, Done That!»

Определение требований к макету экрана с результатами

Теперь, когда вы знаете, как должен выглядеть ваш экран с результатами игры, необходимо преобразовать эскиз дизайна в соответствующий дизайн-макет. В данном случае вы должны обновить файл макета **/res/layout/scores.xml**, который используется классом `QuizScoresActivity`. Как и раньше, вы можете воспользоваться преимуществами элемента-контейнера `RelativeLayout`, чтобы добавить уже знакомый вам заголовок в верхнюю часть экрана. В данном случае сразу после заголовка экрана будет расположен элемент `TabHost` с двумя вкладками, каждая из которых будет содержать элемент-контейнер `TableLayout` с результатами — одна вкладка будет представлять общие результаты игры, а другая — результаты друзей пользователя.

Добавление элемента TabHost

Чтобы добавить поддержку вкладок на экран с результатами игры, необходимо включить в макет элемент `TabHost`, содержащий дочерние вкладки, каждая из которых может использоваться для отображения некоего макета. Элемент `TabHost` довольно сложен в

настройке. Чтобы правильно сконфигурировать его в XML-файле макета, нужно выполнить следующие действия:

- добавить элемент `TabHost`;
- внутрь элемента `TabHost` добавить элсмет контейнер
- внутрь элемента-контейнера `LinearLayout`, добавить элемент `TabWidget` со специальным именем и элемент-контейнер `FrameLayout`;
- определить содержимое каждой вкладки внутри элемента-контейнера `FrameLayout`.

На рис. 9.5 представлен дизайн-макет экрана с результатами.

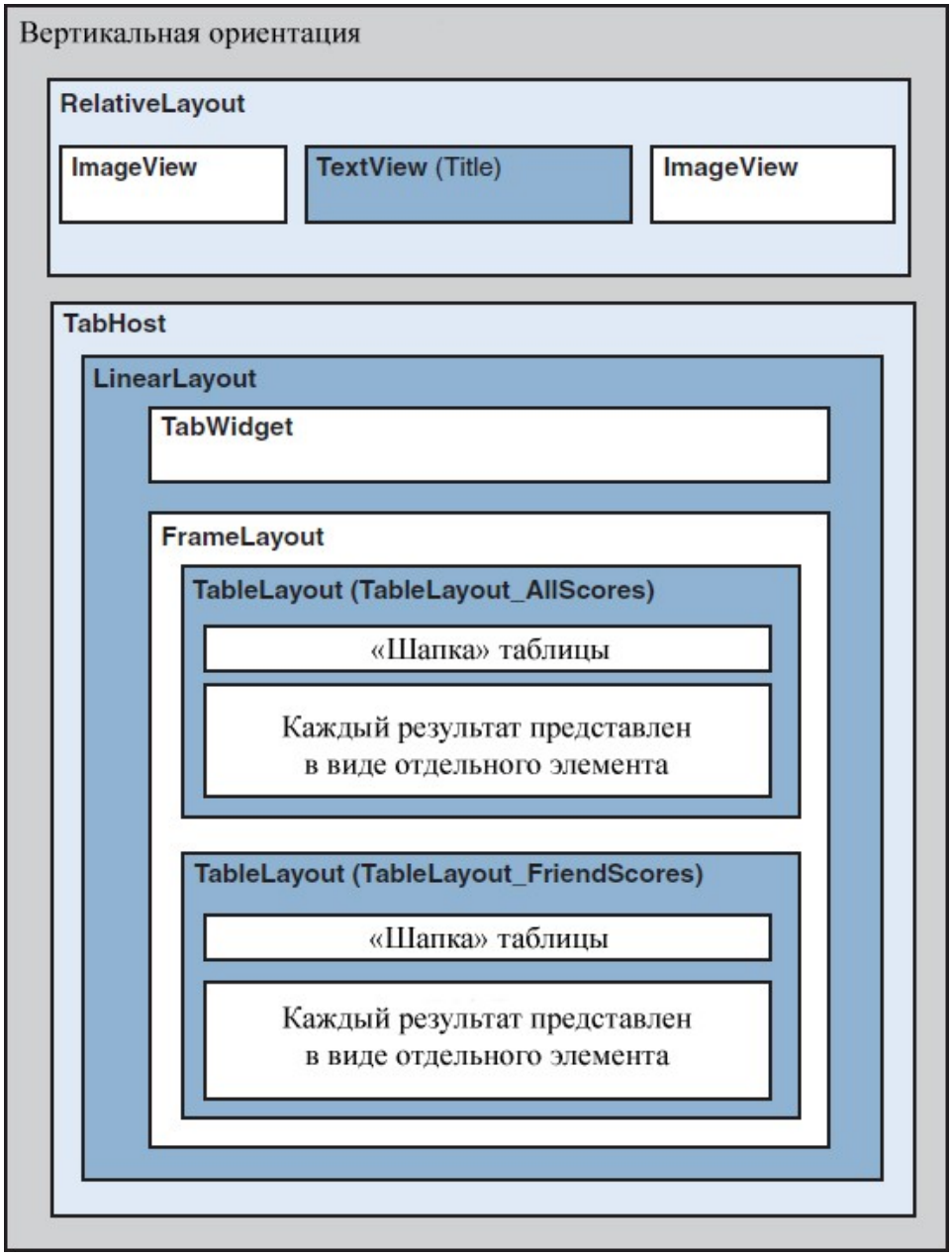

Рис. 9.5. Дизайн-макет экрана с результатами приложения «Been There, Done That!»

РЕАЛИЗАЦИЯ МАКЕТА ЭКРАНА С РЕЗУЛЬТАТАМИ ИГРЫ

Реализацию экрана с результатами игры вы начнете с добавления новых ресурсов в проект. Затем вы обновите файл макета **scores.xml**, чтобы отразить ранее подготовленный дизайн-макет экрана с результатами.

Добавление новых ресурсов в проект

Помимо новых строковых ресурсов, ресурсов пистон м размеров, которые 6\д\т использованы в макете экрана с результатами, вам также потребуется добавить новым тип ресурса — ресурс файла с XML.-данными. Со временем приложение «Been There, Done That!» начнет получать данные о результатах игры с удаленного сервера, однако пока вы можете разработать экран и использовать тестовые данные, представляющие результаты игры. Эти тестовые данные будут представлены в XML-формате, поэтому вы можете использовать структуру, которая в дальнейшем будет применяться для представления реальных результатов.

В этом примере вы добавите в каталог с ресурсами **/res/xml** два XML- файла — **allscores.xml** и **friendscores.xml**, которые представляют тестовые данные с результатами игры.

```
<?xml version="1.0" encoding="utf-8"?>
<!— Это фрагмент текстовых XML-данных с результатами игры —>
<scores>
    <score
        username="LED"
        score="12345"
        rank="1" />
    <score
        username="SAC"
        score="12344"
        rank="2" />
    <score
        username="NAD"
        score="12339"
        rank="3" />
</scores>
```

Для представления результатов игры используется очень простая схема. Единственный элемент `<scores>` имеет несколько дочерних элементов `<score>`. Каждый элемент `<score>` имеет три атрибута: `username`, `score` и `rank`. В нашем случае вы можете предположить, что данные результатов будут заранее отсортированы и будут содержать максимум 20 (или любое другое число) записей.

Обновление макета экрана с результатами

Пользовательский интерфейс экрана с результатами определяется в файле макета **scores.xml**. Чтобы обновить этот макет в соответствии с разработанным дизайн-макетом, выполните следующие шаги.

ВНИМАНИЕ!

В режиме дизайна редактор ресурсов среды разработки Eclipse неправильно отображает элементы TabHost. Для создания макетов, использующих данные элементы, вы должны переключиться в режим XML-разметки. Для просмотра вкладок нужно использовать эмулятор Android или реальное устройство, работающее над управлением операционной среды Android.

1. Удалите все существующие элементы пользовательского интерфейса из макета, как вы делали это для других макетов в этой книге.

2. Добавьте новый элемент-контейнер `LinearLayout`, присвоив его атрибуту `android:background` значение `@drawable/bkgrnd`. Все остальные элементы будут добавляться внутрь данного элемента-контейнера `LinearLayout`.

3. Добавьте тот же заголовок, который вы использовали в других макетах. Он должен включать элемент-контейнер `RelativeLayout` с двумя элементами `ImageView` и одним элементом `TextView`. Атрибуту text элемента `TextView` присвойте строковый ресурс `@string/scores`, чтобы отобразить соответствующий заголовок экрана.

4. Внутрь элемента-контейнера `LinearLayout`, но вне элемента-контейнера `RelativeLayout`, добавьте элемент `TabHost` с именем `TabHostl`. Присвойте его атрибутам layout `width` и `layout_height` значение `fill_parent`.

5. Внутрь элемента `TabHost` добавьте другой элемент-контейнер `LinearLayout`, присвоив его атрибуту `orientation` значение `vertical`. Атрибутам `layout_width` и `layout_height` этого элемента-контейнера присвойте значение `fill_parent`.

6. Внутрь элемента-контейнера `LinearLayout`, добавленного на предыдущем шаге, добавьте элемент `TabWidget`. Присвойте его атрибуту `id` значение `@android:id/tabs`.

7. Внутрь элемента-контейнера `LinearLayout`, добавленного на шаге 5, на том же уровне вложенности, на котором находится элемент `TabWidget`, добавьте элемент-контейнер `FrameLayout`. Присвойте атрибуту `id` этого элемента-контейнера `FrameLayout` значение `@android:id/tabcontent`, а атрибутам `layout_width` и `layout_height` — значение `fill_parent`.

8. Определите содержимое ваших вкладок. Внутрь элемента-контейнера `FrameLayout` добавьте два элемента-контейнера `TableLayout` — по одному для каждой вкладки. Вы будете использовать эти элементы-контейнеры `TableLayout` для отображения результатов игры. Назовите первый элемент-контейнер `TableLayout_FriendScores`. Присвойте атрибутам `layout_width` и `layout_height` этих элементов-контейнеров значение `fill_parent`. Присвойте

атрибуту `stretchColums` значение *, чтобы размеры столбцов изменялось в зависимости от содержимого.

ВНИМАНИЕ!
При создании подобного представления с вкладками вы должны использовать идентификатор, который был указан выше, — `@android:id/tabcontent`; в противном случае, в процессе выполнения приложения возникнут исключения. Это значение ссылается на специальный ресурс в пакете платформы Android. Это не то же самое, что значение `@+id/tabcontent`. В последнем случае будет создан новый идентификатор для объекта макета в пакете вашего приложения.

Вы можете обеспечить прокрутку содержимого для многих элементов, поместив их внутрь элемента `ScrollView`. Например, чтобы обеспечить вертикальную прокрутку содержимого для элемента-контейнера `TableLayout`, добавьте этот элемент-контейнер внутрь элемента `ScrollView` и присвойте атрибуту scrollbars элемента `ScrollView` значение vertical. Также необходимо присвоить значения атрибутам `layout_width` и `layout_height` элемента `ScrollView`.

Часть файла макета экрана с результатами, относящаяся к элементу `TabHost` (с возможностью прокрутки содержимого вкладок, реализованных при помощи элементов-контейнеров `TableLayout`), должна выглядеть следующим образом:

```
<TabHost
    android:id="@+id/TabHost1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <LinearLayout
       android:orientation="vertical"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent">
       <TabWidget
          android:id="@android:id/tabs"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content" />
       <FrameLayout
           android:id="@android:id/tabcontent"
           android:layout_width="fill_parent"
           android:layout_height="fill_parent">
           <ScrollView
               android:id="@+id/ScrollViewAllScores"
               android:layout_width="fill_parent"
               android:layout_height="fill_parent"
               android:scrollbars="vertical">
               <TableLayout
                  android:id="@+id/TableLayout_AllScores"
                  android:layout_width="fill_parent"
                  android:layout_height="fill_parent"
                  android:stretchColumns="*">
               </TableLayout>
           </ScrollView>
```

```
            <ScrollView
               android:id="@+id/ScrollViewFriendScores"
               android:layout_width="fill_parent"
               android:layout_height="fill_parent"
               android:scrollbars="vertical">
               <TableLayout
                    android:id="@+id/TableLayout_FriendScores"
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:stretchColumns="*"></TableLayout>
            </ScrollView>
         </FrameLayout>
   </LinearLayout>
</TabHost>
```

Сохраните файл макета **scores.xml**.

ВНИМАНИЕ!
Если сейчас вы переключитесь в режим визуального представления макета, среда разработки Eclipse может сгенерировать ошибку «`NullPointerException:null`». Редактор ресурсов не поддерживает элементы `TabHost` платформы Android 2.1. При этом макет будет без проблем отображаться на эмуляторе и на реальных устройствах.

РАЗРАБОТКА ДИЗАЙНА ЭКРАНА С ВКЛАДКАМИ

Теперь вы должны полностью сосредоточится на файле **QuizScoresActivity.java** и добавить код для элементе, используемых элементом `TabHost`. Сначала вы должны проинициализировать элемент `TabHost` и затем добавить в него две вкладки, одна из которых — вкладка **All Scores** (Общие результаты) — будет использоваться в качестве вкладки по умолчанию. Наконец, вы воспользуетесь тестовыми данными с результатами игры для заполнения элемента-контеннера `TableLayout` в каждой вкладке.

Настройка элемента TabHost

Для правильного функционирования элемент `TabHost` сначала должен быть проинициализирован. Таким образом, вы должны получить доступ к экземпляру этого элемента, используя метод `findViewById()`. Затем вы должны вызвать метод `setup()` элемента `TabHost`, который выполняет инициализацию элемента `TabHost` и «склеивает» вместе элементы `TabWidget` и `FrameLayout` со специальными именами, чтобы образовать набор вкладок, как показано в следующем коде:

```
TabHost host = (TabHost) findViewById(R.id.TabHost1);
host.setup();
```

Добавление вкладок в элемент TabHost

Теперь, когда вы получили доступ к вашему элементу TabHost и проинициализировали его, вам нужно настроить каждую вкладку, используя метод addTab(). Этот метод принимает параметр типа TabSpec, который определяет содержимое вкладки. Например, следующий код создает вкладку **All Scores** (Общие результаты):

```
TabSpec allScoresTab = host.newTabSpec("allTab");
allScoresTab.setIndicator(getResources().getString(R.string.all_scores
),
   getResources().getDrawable(android.R.drawable.star_on));
allScoresTab.setContent(R.id.ScrollViewAllScores);
host.addTab(allScoresTab);
```

Объект типа TabSpec с именем allScoresTab имеет метку для ссылок allTab. На экране название вкладки представляется элементом TextView и значком (звездочкой). Наконец, содержимое вкладки определяется элементом с идентификатором ScrollViewAllScores, который содержит элемент-контейнер TableLayout с именем TableLayout_AllScores, как было указано в файле макета **scores.xml.** Затем вы добавляете вторую вкладку с меткой для ссылок friendTab в элемент TabHost. Вторая вкладка реализована почти так же, как и первая вкладка, за исключением содержимого (вместо лучших результатов игры на этой вкладке отображаются только результаты игры друзей пользователя).

Назначение вкладки, используемой по умолчанию

Теперь вы должны указать, какая вкладка будет отображаться по умолчанию. Для этого вам нужно вызвать метод setCurrentTabById() и передать в этот метод метку соответствующей вкладки:

```
host.setCurrentTabByTag("allTab");
```

Сохраните файл **QuizScoresActivity.java** и запустите приложение на эмуляторе Android. Когда вы откроете экран с результатами игры, вы увидите две вкладки, однако у вас по-прежнему нет данных с результатами игры для отображения.

ВНИМАНИЕ!
Если на экране приложения должен отображаться только элемент TabHost, рассмотрите возможность использования класса TabActivity, который упрощает работу с элементом TabHost.

.

РАБОТА С XML-ДАННЫМИ

Платформа Android имеет ряд механизмов для работы с XML-данными, включая поддержку следующих технологий:

- SAX (Simple API for XML);
- XML Pull Parser;
- Limited DOM Level 2 core support.

Выбор используемой технологии для работы с XML-данными зависит от конкретного проекта. В данном примере вы просто хотите прочитать содержимое простого XML-файла и извлечь тестовые данные с результатами игры.

Получение XML-ресурсов

Сначала вы должны получить доступ к тестовым XML-данным, которые вы сохранили в ресурсах проекта. В частности, вам нужно разобрать файл /**res/ xml/allscores.xml.**

Вы можете проинициализировать экземпляр класса `XmlResourceParser`, используя метод `getXML()`, как показано в следующем коде:

```
XmlResourceParser mockAllScores =
getResources().getXml(R.xml.allscores);
```

Разбор XML-файлов при помощи класса XmlResourceParser

Тестовые файлы с результатами игры используют очень простую схему, содержащую всего два тега:`<scores>` и `<score >`. Вам нужно найти все теги `<score>` и получить значения их атрибутов username, rank и score. Поскольку подразумевается, что вы будете работать с небольшим количеством данных, процедуру разбора XM L-данных можно реализовать при помощи простого цикла `while()`, используя для перехода между событиями метод `next()`, как показано в следующем коде:

```
int eventType = -1;
boolean bFoundScores = false;
// Find Score records from XML
while (eventType != XmlResourceParser.END_DOCUMENT) {
    if (eventType == XmlResourceParser.START_TAG) {
        // Get the name of the tag (eg scores or score)
        String strName = scores.getName();
        if (strName.equals("score")) {
           bFoundScores = true;
           String scoreValue = scores.getAttributeValue(null,
           "score");
           String scoreRank = scores.getAttributeValue(null, "rank");
           String scoreUserName =
              scores.getAttributeValue(null, "username");
           insertScoreRow(scoreTable, scoreValue, scoreRank,
              scoreUserName);
        }
    }
    eventType = scores.next();
}
```

В цикле вы ожидаете появления события START_TAG. Когда имя тега совпадет с именем тега `<score>`, вы будете знать, что обнаружена часть данных с результатами игры. После этого вы можете получить обнаруженные данные, используя метод `getAttributeValue()`. Для каждого обнаруженного результата вы добавляете новый элемент `TableRow` в соответствующий элемент-контейнер `TableLayout` (на соответствующей вкладке).

Внесение завершающих штрихов иа экраи с результатами

Написав код для разбора двух тестовых XML-файлов и заполнения двух элементов-контейнеров `TableLayout` внутри элемента `TabHost`, внести несколько незначительных изменений в класс `QuizScoresActivity`. Вы должны добавить заголовок в виде элемента `TableRow` в каждый элемент-контейнер `TableLayout`, придать заголовкам столбцов привлекательный вид, а также обработать ситуацию, когда данные о результатах игры недоступны.

Внеся эти завершающие штрихи, сохраните исходный код класса и запустите приложение на эмуляторе или на реальном устройстве. Когда откроете экран с результатами, то увидите, чтобы обе вкладки содержат данные (как показано на рис. 9.6).

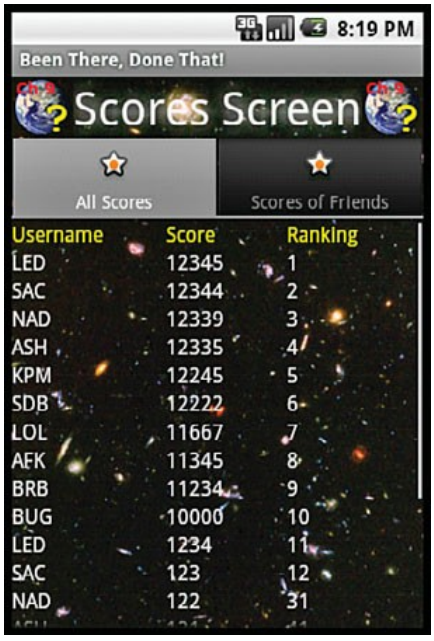

Рис. 9.6. Экран с результатами
приложения «Been There, Done That!»

ИТОГИ

В этом часе вы добавили два новых экрана в опрос-викторину «Been There, Done That!». В процессе реализации экрана с инструкциями вы узнали, как можно отображать большие объемы данных при помощи элемента `TextView` с возможностью прокрутки содержимого. Вы также узнали, как обращаться к ресурсам файлов и изменять настройки макета программным путем. В процессе реализации экрана с результатами вы

познакомились с элементом `TabHost`, а также узнали, как можно разобрать XML-данные для отображения тестовых результатов.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Почему некоторым элементам, находящимся внутри элемента `TabHost`, необходимо присваивать специальные значения атрибутов id, определенные в пакете платформы Android?

Ответ: Порой вы будете сталкиваться с ситуациями, когда элементу необходимо присвоить специальное имя для его правильного функционирования. Чем сложнее элемент, тем выше вероятность, что операционной системе Android потребуется немного «клея» (или «магии») для нагрузки правильных шаблонов и ресурсов, используемых для отображения элемента в привычном для нас виде. Обычно подобные требования к именованию описаны **в** документации инструментария Android SDK.

Вопрос: При загрузке экрана с результатами игры возникает небольшая задержка. Почему?

Ответ: Существует ряд причин, по которым этот экран действует медленнее по сравнению с другими. Во-первых, вы осуществляете разбор XML-данных, что само по себе может являться трудоемкой операцией. Во-вторых, вы создаете большое число элементов-представлений view для отображения данных о результатах игры. Вы должны всегда заботиться о том, чтобы трудоемкие вычисления выносились за пределы потока, управляющего пользовательским интерфейсом, — в этом случае приложение будет быстрее реагировать на действия пользователя, и вы сможете избежать ситуаций, когда операционная система Android завершает работу приложения из-за того, что оно вовремя не отвечает на запросы системы. Вы могли бы легко добавить рабочий поток, который бы занимался обработкой XM L-данных, а также попробовать найти другие, более эффективные, элементы для отображения данных о результатах игры. Наконец, в среде разработки Eclipse при включенном отладчике производительность приложения резко снижается.

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Элемент `TextView` способен отображать большой объем текста.

2. Какой класс может быть использован для упрощения создания экранов с вкладками?

A. Tabify.
B. TabActivity.
C. TabController.

3. Верно ли это? XML-файлы обрабатываются менеджером XML-ресурсов, поэтому в их дополнительной обработке нет никакой необходимости.

4. Элемент какого типа может быть использован для добавления возможности прокрутки содержимого?

A. Scroll Layout.
B. Scroller.
C. ScrollView.

Ответы

Верно. Элемент `TextView` способен отображать большие объемы текста с возможностью добавления горизонтальных и вертикальных полос прокрутки.

Б. Для экрана, на котором должен отображаться лишь набор вкладок, можно использовать класс `TabActivity`, который позволяет эффективно выполнять операции по настройке вкладок и другие задачи.

Неверно. XML-файлы могут быть добавлены в качестве ресурсов проекта, однако без разбора XM L-данных все равно не обойтись. В вашем распоряжении есть три механизма разбора XML-данных, при этом для разбора XML-данных из ресурсов по умолчанию используется механизм `XML Pull Parser`.

В. Элемент `ScrollView` может быть использован для размещения дочерних элементов view внутри области с возможностью прокрутки содержимого.

Упражнения

1. Запустите приложение и нажмите на каждую из ссылок, присутствующих в тексте на экране с инструкциями. Обратите внимание, насколько легко можно интегрировать Android-приложения с платформой Android. Попробуйте запустить другие типы интентов и понаблюдать за тем, что произойдет. Список существующих интентов можно найти в «Реестре интентов» по адресу **www.openintents.org/en/intentstable**.

2. Добавьте третью вкладку на экран с результатами игры, на которой будет отображаться текущий результат игры пользователя. Измените содержимое этой новой вкладки по сравнению с двумя другими вкладками, включив вместо элемента-контейнера `TableLayout` элемент `TextView`. Текущий результат игры пользователя будет храниться в виде настройки приложения.

Час 10. ПОСТРОЕНИЕ ФОРМ ДЛЯ СБОРА ВВОДИМЫХ ПОЛЬЗОВАТЕЛЕМ ДАННЫХ

Вопросы, рассматриваемые в этом часе:

- **разработка дизайна и реализация экрана с настройками приложения;**
- **работа с элементами** `EditText`**;**
- **работа с элементами** `Button`**;**
- **работа с элементами** `Spinner`**;**
- **сохранение данных формы с использованием класса** `SharedPreferences`**.**

В этом часе вы займетесь реализацией экрана с настройками приложения «Been There, Done That!». Экран с настройками представляет собой форму для ввода информации о конфигурации приложения, включая логин пользователя и настройки его профиля. Различные параметры предполагают использование различных элементов для ввода данных, включая элементы `EditText`, `Spinner` и `Button`. Наконец, вам нужно убедиться, что все настройки сохранены как часть конфигурации приложения.

РАЗРАБОТКА ДИЗАЙНА ЭКРАНА С НАСТРОЙКАМИ

Экран с настройками приложения должен позволять пользователю настраивать ряд параметров игры и сохранять их. Параметры игры могут быть представлены в виде текстовых полей ввода, раскрывающихся списков или в виде других, более сложных, элементов. (Со временем вы захотите добавить социальные возможности для совместной игры с друзьями, однако к этому требованию мы вернемся позже.) Пока же вы реализуете простой экран с настройками приложения, содержащий пять основных параметров:

- **Nickname** (Ник). Имя, которое отображается в списках с результатами игры. Количество символов, вводимых при помощи этого поля ввода, не должно превышать 20 — это значение может быть другим, однако для ваших целей этой длины вполне достаточно.

- **Email**. Уникальный идентификатор для каждого пользователя. Данный параметр представлен текстовым полем ввода.

- **Password** (Пароль). Механизм верификации пользователя. Представлен текстовым полем для ввода паролей. При вводе пароля пользователь должен указать его дважды, чтобы подтвердить правильность ввода. Сам пароль может храниться в открытом виде.

- **Date of Birth** (Дата рождения). Используется для проверки минимально допустимого возраста игрока в тех случаях, когда это необходимо. Данный параметр представлен в виде нескольких полей для ввода даты, взаимодействие с которыми не вызывает у пользователей никаких проблем.

- **Gender** (Пол). Представляет часть демографической информации, которая может быть использована для формирования специальных таблиц с результатами или для таргетинга рекламы. Этот параметр может принимать три различных значения: **Male** (Мужской) (1), **Female** (Женский) (2) или **Prefer Not to Say** (Не указано) (0).

На рис. 10.1 представлен эскиз дизайна экрана с настройками.

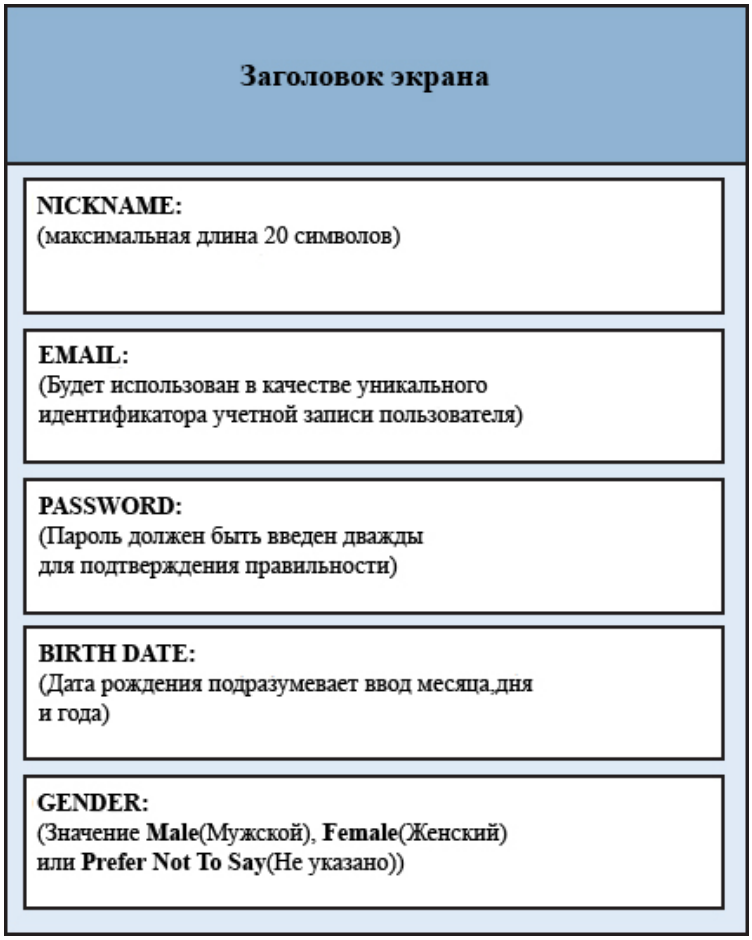

Рис. 10.1. Эскиз дизайна экрана с настройками приложения «Been There, Done That!»

Экран с настройками приложении будет содержать несколько элементов для ввода различной информации, поэтому вы должны аккуратно распределять свободное пространство экрана. Как и другие экраны приложения, этот экран в своей верхней части будет содержать заголовок.

Над заголовком экрана для каждого параметра вы добавите отдельную область. Поскольку со временем вам может потребоваться добавить новые параметры, вы должны

поместить область экрана с настройками внутрь элемента `ScrollView`. В этом случае пользователь сможет увидеть все параметры, которые не умещаются на экране, просто прокрутив содержимое данной области. Элемент `ScrollView` может иметь только один дочерний элемент, поэтому вы можете включить ваши параметры в еще один элемент-контейнер `LinearLayout` с вертикальной ориентацией.

Каждый параметр будет представлен двумя строками в элементе-контейнере `LinearLayout`: строкой, представленной элементом `TextView`, с названием параметра, и строкой, содержащей элемент для ввода значения параметра. Например, для параметра **Nickname** (Ник) потребуется строка с элементом `TextView`, в котором будет отображаться название параметра (`Nickname:`), и строка для элемента `EditText`, позволяющего пользователю ввести текстовое значение.

Теперь вам нужно определить, какой элемент лучше всего подходит для каждого параметра:

- Параметры Nickname (Ник) и Email представляют собой разновидности однострочного поля ввода, поэтому они могут быть представлены элементами `EditText`.

- Для параметра Password (Пароль) требуется использование двух элементов `EditText`. Однако сам пароль не должен вводиться непосредственно на экране с настройками. Вместо этого вы создадите элемент `Button`, который будет открывать диалоговое окно `Dialog`, позволяющее пользователю ввести пароль (при помощи двух элементов `EditText`). На основном экране с настройками может просто выводиться информация о том, был ли установлен пароль или нет, в элементе `TextView`.

- Для параметра **Date of Birth** (Дата рождения) требуется элемент `DatePicker`. Поскольку элемент `DatePicker` фактически реализован при помощи трех отдельных элементов — для выбора месяца, дня и года, он занимает много пространства на экране. Поэтому, вместо того, чтобы добавлять этот элемент непосредственно на экран с настройками, вы можете добавить элемент Button, который будет использован для отображения диалогового окна **DatePickerDialog**. Пользователь выберет подходящую дату и закроет диалоговое окно, а указанная дата будет отображена (без возможности редактирования) на основном экране с настройками при помощи элемента `TextView`.

- Параметр **Gender** (Пол) позволяет выбирать одно из трех значений, поэтому для реализации лучше всего подойдет элемент `Spinner` (раскрывающийся список).

На рис. 10.2 изображен дизайн-макет основного экрана с настройками.

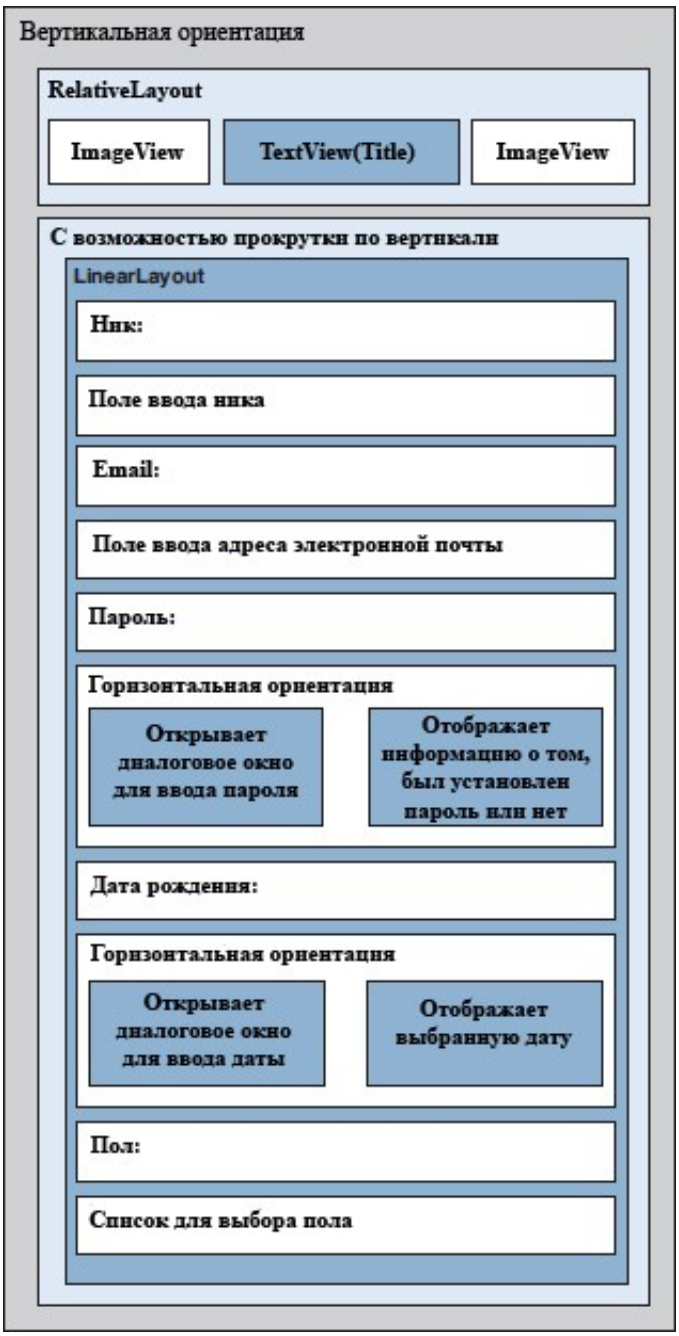

Рис. 10.2. Дизайн-макет экрана с настройками приложения «Been There, Done That!»

РЕАЛИЗАЦИЯ МАКЕТА ЭКРАНА С НАСТРОЙКАМИ

Реализацию экрана с настройками вы начнете с добавления новых ресурсов в проект. Затем вы обновите файл макета **settings.xml**, чтобы отразить подготовленный дизайн-макет экрана с настройками.

Добавление новых ресурсов в проект

В экранах с формами, предназначенными для сбора информации, применяется больше ресурсов, чем на экранах любых других типов. Вам нужно добавить ряд новых ресурсов, которые будут использованы в макете экрана с настройками. Помимо строковых ресурсов,

ресурсов размеров и цветов, вам также потребуется добавить ресурс нового типа — строковый массив.

ДОБАВЛЕНИЕ НОВЫХ СТРОКОВЫХ РЕСУРСОВ

На экране с настройками применяется множество новых строковых ресурсов. Вам нужно добавить следующие строковые ресурсы в файл ресурсов **strings.xml**:

- текст для названия каждого параметра, отображаемого при помощи элемента `TextView` (например, `NickName:`);
- текст для каждого элемента `Button` (например, `SetPassword`);
- текст, который будет отображаться в элементе `TextView` при установленном и не установленном пароле;
- текст, который будет отображаться в элементе `TextView` в том случае, когда значение параметра **Date of Birth** (Дата рождения) не установлено;
- текст, который будет отображаться в элементе `TextView` в том случае, когда значения в двух полях для ввода пароля будут отличаться;
- текст для каждого значения параметра **Gender** (Пол), используемого в элементе `Spinner` (например, «Male»).

Добавив все эти строковые ресурсы, сохраните файл ресурсов **strings.xml.**

ДОБАВЛЕНИЕ НОВОГО РЕСУРСА СТРОКОВОГО МАССИВА

Элементы `Spinner`, как и элементы `ListView`, используют адаптеры данных. Вы определили строковые ресурсы, представляющие допустимые значения для параметра **Gender** (Пол), но вам еще нужно сгруппировать эти значения, чтобы получить подходящий набор данных.

Простейший набор данных — это массив значений типа `String` (и соответствующий класс `ArrayAdapter`). Чтобы сгруппировать строковые ресурсы, представляющие пол человека (Male, Female, Prefer Not To Say), в виде отдельного массива, нужно создать ресурс нового типа, называемый строковым массивом.

Чтобы создать ресурс строкового массива, вы должны добавить новый файл ресурсов с именем **/res/values/arrays.xml**. Внутри этого файла необходимо создать элемент `string-array` с именем `genders`. В созданный элемент string-array добавляется три элемента item — по одному для каждого строкового ресурса.

Предположим, что вы добавили в файл ресурсов **strings.xml** следующие три строковых ресурса, представляющих пол пользователя и которые были определены ранее:

```
<string
    name="gender_male">Male</string>
<string
    name="gender_female">Female</string>
<string
    name="gender_neutral">Prefer Not To Say</string>
```

В файле ресурсов **arrays.xml** вы присваиваете каждому элементу `item` в строковом массиве genders соответствующий строковый ресурс. Например, первому элементу `item` в массиве (с индексом 0) будет присвоено значение `@string/gender_neutral`. Итоговое содержимое файла ресурсов **arrays.xml** будет выглядеть следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array
        name="genders">
        <item>@string/gender_neutral</item>
        <item>@string/gender_male</item>
        <item>@string/gender_female</item>
    </string-array>
</resources>
```

Сохраните файл ресурсов **arrays.xml.** Теперь, когда вы захотите загрузить ресурс строкового массива `genders` в память, вы сможете обратиться к нему из кода приложения, используя идентификатор ресурса `R.array.genders`.

Обновление макета экрана с настройками

Файл макета **settings.xml** определяет пользовательский интерфейс экрана с настройками. Как и раньше, вы открываете редактор ресурсов среды разработки Eclipse и удаляете все существующие элементы из макета. После этого выполните следующие шаги, чтобы получить желаемый макет экрана на основе подготовленного дизайн-макета.

Добавьте уже ставший привычным элемент-контейнер `LinearLayout`, присвоив его атрибуту `background` значение `@drawable/bkgrnd`. Все остальные элементы будут добавляться внутрь этого элемента-контейнера `LinearLayout`.

Добавьте те же элементы, представляющие заголовок экрана, которые были добавлены на другие экраны.

Под заголовком экрана расположите элемент `ScrollView`, который будет содержать все параметры вашего приложения. Присвойте его атрибуту `isScrollContainer` значение `true`, а атрибуту `scrollbars` — значение `vertical`. Присвойте атрибутам `layout_width` и `layout_height` этого элемента значение `fill_parent`.

Внутрь элемента `ScrollView` добавьте элемент-контейнер `LinearLayout`, в котором будут содержаться параметры приложения. Присвойте атрибуту `orientation` этого элемента значение `vertical`, а атрибутам `layout_width` и `layout_height` — значение `fill_parent`. Все последующие элементы будут добавлены внутрь данного элемента-контейнера `LinearLayout`.

Внутрь элемента-контейнера `LinearLayout` добавьте элемент `TextView`, который будет использоваться для отображения названия параметра **Nickname** (Ник). Под элементом `TextView` с названием параметра расположите элемент `EditText`. Присвойте его атрибуту `id` значение `EditText_Nickname`, атрибуту `maxLength` — значение `20`, атрибуту `maxLines` — значение 1, а атрибуту `inputType` — значение `textPersonName`.

Добавьте элемент `TextView`, который будет использоваться для отображения названия параметра **Email**. Под этим элементом расположите еще один элемент `EditText`, присвоив его атрибуту `id` значение `EditText_Email`, атрибуту `maxLines` — значение 1, а атрибуту `inputType` — значение `textEmailAddress`.

Добавьте область для параметра **Password** (Пароль), включив в макет еще один элемент `TextView`, который будет использоваться для отображения названия данного параметра. Под добавленным элементом добавьте элемент-контейнер `LinearLayout` с горизонтальной ориентацией, включив в него два дополнительных элемента: элемент `Button` и элемент `TextView`. Присвойте атрибуту `id` элемента `Button` значение `Button_Password`, а атрибуту `text` присвойте строковый ресурс с названием кнопки **Password** (Пароль). Настройте элемент `TextView` таким образом, чтобы он использовался для отображения строки состояния параметра **Password** (Пароль) (на данный момент, это строка `Password not set`).

На том же уровне вложенности, на котором находится область для параметра **Password** (Пароль), добавьте область для параметра **Date of Birth** (Дата рождении). Сначала добавьте еще один элемент-контейнер будет использоваться для отображения названия параметра **Date of Birth** (Дата рождения). Затем добавьте еще один элемент-контейнер `LinearLayout` с горизонтальной ориентацией, включив в но дополнительных элемента: элемент **Button** и элемент `TextView`. Присвойте атрибуту `id` элемента управлении `Button` значение `Button_DOB`, а атрибуту `text` присвойте строковый ресурс с названием кнопки **Date of Birth** (Дата рождения). Настройте элемент `TextView` таким образом, чтобы он использовался для отображения строки состояния параметра **Date of Birth** (Дата рождения) (на данный момент, это строка `Date not set`).

Добавьте последнюю область для параметра **Gender** (Пол), включив элемент `TextView`, который будет использоваться для отображения названия параметра **Gender** (Пол). Затем добавьте элемент `Spinner` и присвойте его атрибуту `id` значение `Spinner_Gender`.

Перед тем, как сохранить внесенные изменения, настройте размеры шрифтов, гарнитуры, цвета, высоту и ширину элементов, чтобы добиться желаемого вида экрана.

Сохраните файл макета **settings.xml**.

ИСПОЛЬЗОВАНИЕ СТАНДАРТНЫХ ЭЛЕМЕНТОВ ФОРМ

Теперь, когда файл макета **settings.xml** готов, вам понадобится обновить класс `QuizSettingsActivity`, чтобы заполнить элементы макета данными и позволить

пользователю редактировать и сохранять введенные значения. В этом разделе вы узнаете, как сохранять и восстанавливать данные формы.

Работа с элементами EditText

Элемент `EditText`, который наследуется от элемента `TextView`. применяется для сбора текстовых данных, вводимых пользователем. На рис. 10.3 изображен простой элемент `EditText`.

Рис. 10.3. Элемент EditText для ввода текстовых данных

НАСТРОЙКА ЭЛЕМЕНТОВ EDITTEXT

Все атрибуты, присущие элементу `TextView` (например, атрибуты `textColor` и `textSize`**),** доступны и в элементах `EditText`. Ниже представлен список некоторых атрибутов элемента `EditText`, которые применяются на экране с настройками.

- `inputType`. Информирует операционную систему Android о том, каким образом можно помочь пользователю ввести значение в данное поле ввода. Например, атрибуту `inputType` элемента `EditText`, связанного с параметром Email, вы присвоили значение `textEmailAddress`, которое указывает системе Android использовать программную клавиатуру, ориентированную на ввод адресов электронной почты (с символом @). Значение `textPassword` атрибута `inputType` позволяет автоматически скрывать пароль, вводимый пользователем.

- `minLines` и `maxLines`. Позволяют ограничить количество строк текста, которое можно ввести в данном элементе.

- `maxLength`. Ограничивает количество символов, которое можно ввести в данном элементе. Например, вы можете ограничить количество символов для параметра Nickname (Ник), указав для атрибута `maxLength` элемента `EditText`, связанного с этим параметром, значение `20`.

ОБРАБОТКА ВВЕДЕННОГО ТЕКСТА

Как и в случае с элементом `TextView`, вы можете обращаться к тексту, хранящемуся **в** элементе `EditText`, при помощи методов `getText()` и `setText()`. Например, чтобы получить строку, введенную в элемент `EditText` с именем `EditText_Nickname`, можно использовать метод `getText()`, как показано в следующем коде:

```
EditText nicknameText = (EditText)
findViewById(R.id.EditText_Nickname);

String strNicknameToSave = nicknameText.getText().toString();
```

Метод `getText()` возвращает объект типа `Editable`, однако в данном случае вы просто хотите получить его строковый эквивалент.

СОХРАНЕНИЕ ДАННЫХ, ВВЕДЕННЫХ В ЭЛЕМЕНТ EDITTEXT

Для обработки данных, введенных в элемент `EditText`, вы должны определить момент ввода нового текста. Для этого вы можете прослушивать события нажатий клавиш для элемента `EditText` и сохранять введенный текст, если были нажаты определенные клавиши. Например, для прослушивания события нажатия клавиши **Enter** в процессе ввода значения параметра **Nickname** (Ник) вы могли бы зарегистрировать обработчик `View.OnKeyListener`, используя метод `setOnKeyListener()` элемента `EditText`, как показано в следующем коде:

```
final EditText nicknameText =
    (EditText) findViewById(R.id.EditText_Nickname);
nicknameText.setOnKeyListener(new View.OnKeyListener() {
     public boolean onKey(View v, int keyCode, KeyEvent event) {
          if ((event.getAction() == KeyEvent.ACTION_DOWN) &&
               (keyCode == KeyEvent.KEYCODE_ENTER)) {
               String strNicknameToSave =
               nicknameText.getText().toString();
               // TODO: Save Nickname setting (strNicknameToSave)
               return true;
          }
          return false;
     }
});
```

ПРОСЛУШИВАНИЕ СОБЫТИЙ НАЖАТИЯ КЛАВИШ ДЛЯ ЭЛЕМЕНТА EDITTEXT

Скоро вы займетесь разработкой диалогового окна для ввода пароля. Представим, что в процессе ввода данных пользователем вы хотите проверять на совпадение строки в двух элементах `EditText`, предназначенных для ввода и подтверждения пароля, с именами `EditText_Pwdl` и `EditText_Pwd2`. Третий элемент `TextView` с именем `TextView_PwdProblem` будет отображать информацию о том, совпадают введенные пароли или нет.

Сначала вы должны получить экземпляр каждого из элементов:

```
final EditText p1 = (EditText) findViewById(R.id.EditText_Pwd1);
final EditText p2 = (EditText) findViewById(R.id.EditText_Pwd2);
final TextView error = (TextView)
findViewById(R.id.TextView_PwdProblem);
```

После этого вы связываете экземпляр класса `TextWatcher` со вторым элементом `EditText`, используя метод `addTextChangedListener()`, как показано в следующем коде:

```
p2.addTextChangedListener(new TextWatcher() {
```

```
    @Override
    public void afterTextChanged(Editable s) {
        String strPass1 = p1.getText().toString();
        String strPass2 = p2.getText().toString();
        if (strPass1.equals(strPass2)) {
            error.setText(R.string.settings_pwd_equal);
        } else {
            error.setText(R.string.settings_pwd_not_equal);
        }
    }
// Other required overrides do nothing
});
```

Пользователь может вводить пароль элемент `EditText` с именем `EditText_Pwd1` в обычном режиме. Однако всякий раз, когда пользователь вводит очередной символ в элемент с именем `EditText_Pwd2`, вы сравниваете текст в ооoич длементах `EditText` и присваиваете соответствующий текст длеченгу `TextView` с именем `TextView _PwdProblem`, чтобы показать, совпадают введенные пароли или нет (рис. 10.4).

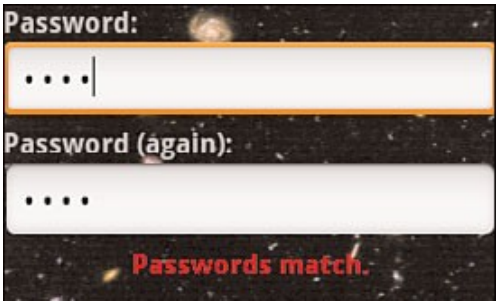

Рис. 10.4. Диалоговое окно **Password** (Пароль)
с двумя элементами EditText и элементом TextView

Работа с элементами Button

Элемент `Button` платформы Android достаточно прост в использовании, впрочем, как и другие элементы форм. Вообще говоря, элемент `Button` представляет собой область с текстовой строкой, на которую можно нажимать. На рис. 10.5 изображен элемент `Button`.

Рис. 10.5. Элемент Button

НАСТРОЙКА ЭЛЕМЕНТОВ BUTTON

Многие из атрибутов, присущих элементам `TextView`, например атрибуты `textColor` и `textSize`, доступны и для текста элемента `Button`. Для экрана с настройками вам

потребуются два простых элемента Button: один для открытия диалогового окна Dialog, предназначенного для ввода пароля, и один для открытия диалогового окна DatePickerDialog. Чтобы настроить эти элементы Button, вы назначите каждому из них уникальный идентификатор и присвоите соответствующее значение атрибуту text, Кроме того вы присвоите атрибутам layout_wtdth и layout_height каждого элемента Button значение wrap_content, чтобы размеры каждой кнопки изменялись пропорционально размеру ее названия.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

В платформе Android фактически поддерживается два типа элементов Button; обычный элемент Button и элемент ImageButton. Элемент ImageButton во многом ведет себя так же, как обычный элемент Button, за исключением того, что вместо текстового названия кнопки отображается графический ресурс типа Drawable.

Вы можете использовать атрибуты, чтобы изменить внешний вид элемента Button. Например, вы можете изменить форму кнопки (по умолчанию используется светлый прямоугольник со скругленными углами), присвоив атрибутам background, drawableTop, drawableBottom, drawableLeft и drawableRight элемента Button подходящие графические ресурсы типа Drawable.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО

Попробуйте изменить внешний вид элемента Button с именем Button_DCB, внеся следующие изменения в файл макета **settings.xml**:

1. Измените значение атрибута background элемента Button, указав в качестве значения этого атрибута графический ресурс типа Drawable с названием @drawable/textured.
2. Измените значение атрибута drawableTop элемента Button, указав в качестве значения этого атрибута графический ресурс типа Drawable с названием @drawable/divider.
3. Измените значение атрибута drawableBottom элемента Button, указав в качестве значения этого атрибута графический ресурс типа Drawable с названием @drawable/divider. Обратите внимание, что сейчас на экране элемент Button выглядит как уродливое создание оранжевого цвета. Вы создали монстра!
4. Верните внешний вид элемента Button к его предыдущему состоянию, удалив значения атрибутов background, drawableTop и drawableBottom элемента с именем Button_DOB.

ОБРАБОТКА СОБЫТИЙ НАЖАТИЙ НА КНОПКУ

Обработка событий нажатий для элемента Button осуществляется при помощи метода setOnClickListener(). В частности, вы должны реализовать метод onClick() класса View.OnClickListener. Именно здесь должна происходить обработка событий нажатий.
Например, чтобы обработать ситуацию, когда пользователь нажимает на элемент Button с именем Button_DOB, нужно добавить следующий код в метод onCreate() класса QuizSettingsActivity для обработки соответствующего события:

```
Button pickDate = (Button) findViewById(R.id.Button_DOB);
pickDate.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
```

```
            // Handle date picking dialog
        }
});
```

Вы получаете экземпляр элемента `Button`, используя метод `findViewByld()`, и затем устанавливаете для него экземпляр класса `View.OnClickListener`, используя метод `setOnClickListener()`. Внутри метода `onClick()` будет происходить открытие диалогового окна **DatePickerDialog**. Однако вы еще не совсем готовы для реализации диалогового окна **Dialog**. Вместо этого вы можете отобразить небольшое всплывающее сообщение-подсказку, называемое всплывающим уведомлением.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Всплывающее уведомление — это представление с определенным сообщением, которое отображается на экране в течение нескольких секунд и затем исчезает.

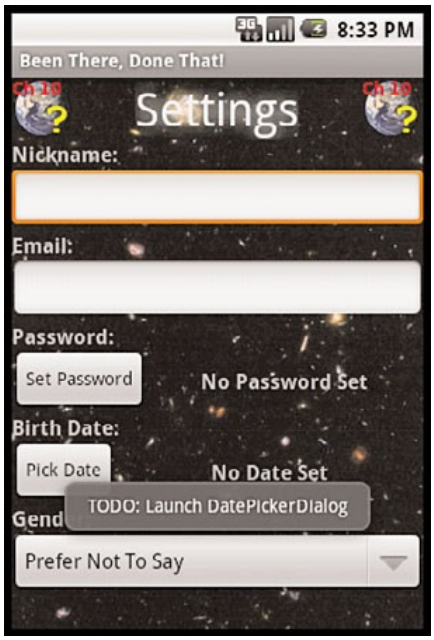

Рис. 10.6. Всплывающее уведомление, отображаемое при нажатии на кнопку

Например, вы могли бы добавить всплывающее уведомление в методе `onClick()` для элемента с именем `Button_DOB`, используя код наподобие следующего:

```
Toast.makeText(QuizSettingsActivity.this,
    "TODO: Launch DatePickerDialog",
    Toast.LENGTH_LONG).show();
```

На рис. 10.6 изображено результирующее всплывающее уведомление.

Работа с элементами Spinner

Элемент `Spinner` — это специфическая реализация раскрывающегося списка на платформе Android. В закрытом состоянии элемент во многом схож с обычным раскрывающимся списком (рис. 10.7, слева), но когда элемент `Spinner` находится в активном состоянии, на экране отображается диалоговое окно с вариантами выбора (рис. 10.7, справа) вместо отображения раскрывающегося списка на основном экране.

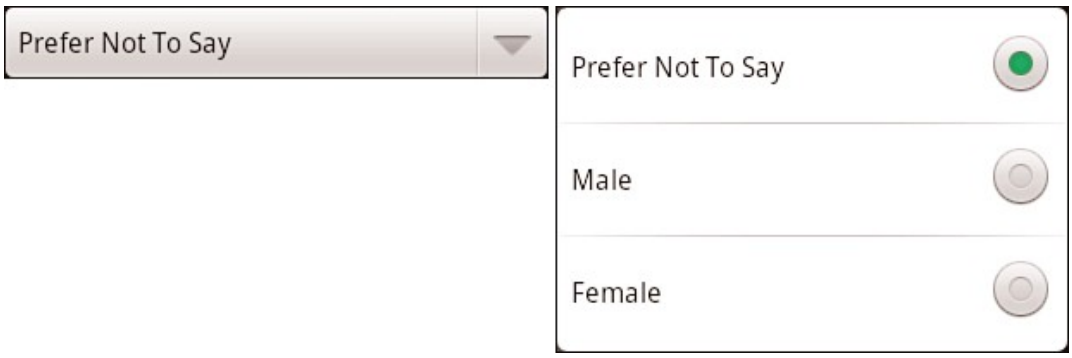

Рис. 10.7. Элемент Spinner в закрытом (слева) и активном (справа) состояниях

НАСТРОЙКА ЭЛЕМЕНТОВ SPINNER

Большая часть настройки элемента `Spinner` должна осуществляться программным путем. Как и в случае с элементом `ListView`, элемент `Spinner` использует адаптеры данных для связывания содержимого из набора данных с каждым представлением, отображаемым в элементе. Чтобы загрузить данные в элемент `Spinner`, необходимо выполнить следующие шаги:

1. Получить экземпляр элемента `Spinner` из макета.
2. Настроить адаптер данных для связывания необходимых данных с элементом.
3. Вызвать метод `setAdapter()` элемента `Spinner`.

Чтобы получить экземпляр Spinner из макета, используется уже знакомый вам метод `findViewById()`:

```
final Spinner spinner = (Spinner) findViewById(R.id.Spinner_Gender);
```

Теперь нужно настроить ваш адаптер данных. Элемент Spinner отображает данные в закрытом и активном состояниях по-разному. Поэтому вы должны создать макеты шаблонов для обоих состояний. К счастью, платформа Android включает несколько специальных ресурсов макетов, которые помогают создавать элементы Spinner, содержащие только текст. В частности, вы можете использовать ресурс макета с именем `android.R.layout.simple_spinner_item` для создания подходящего представления для каждого элемента списка в стандартном элементе `Spinner`. Вы можете использовать ресурс макета с именем `android.R.layout.simple_spinner_dropdown_item` в качестве шаблона представления для раскрывающегося списка.

Используя эти удобные встроенные макеты шаблонов, вы можете загрузить ваш строковый массив с именем **genders** в экземпляр класса `ArrayAdapter`, используя метод `createFromResource()`:

```
ArrayAdapter<?> adapter = ArrayAdapter.createFromResource(this,
      R.array.genders, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(
    android.R.layout.simple_spinner_dropdown_item);
```

Наконец, вы вызываете метод `setAdapter()` элемента `Spinner`, чтобы связать загруженные данные с этим элементом:

```
spinner.setAdapter(adapter);
```

РАБОТА СО ЗНАЧЕНИЯМИ В ЭЛЕМЕНТЕ SPINNER

После того, как в элемент `Spinner` будут загружены данные, вы сможете узнать, какое значение выбрал пользователь, используя метод `setSelection()`. Например, вы знаете, что значение, соответствующее мужскому полу, хранится в строковом массиве под индексом 2 (поскольку в строковом массиве применяется индексация с нуля). Поскольку вы выбрали вариант сопоставления пола пользователя непосредственно с индексами в строковом массиве, вы можете выбрать в элементе `Spinner` значение **Female** (Женский), используя следующий код:

```
spinner.setSelection(2);
```

Класс элемента `Spinner` также предоставляет ряд методов для получения выбранного пользователем значения.

ПРОСЛУШИВАНИЕ СОБЫТИЙ ВЫБОРА ЗНАЧЕНИЙ В СПИСКЕ

Вам нужно сохранить выбранное значение в элементе управления `Spinner` сразу, как только пользователь выберет его. Для этого используется метол `setOnItemSelectedListener()` элемента `Spinner`, чтобы установить слушателя событий выбора значений. В частности, вам нужно реализовать метод `onItemSelected()` класса AdapterView`.setOnItemSelectedListener()`, как показано в следующем коде:

```
spinner.setOnItemSelectedListener(
    new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View
        itemSelected,
            int selectedItemPosition, long selectedId) {
            // TODO: Save item index (selectedItemPosition) as Gender
            setting
        }
        // … Other required overrides
    });
```

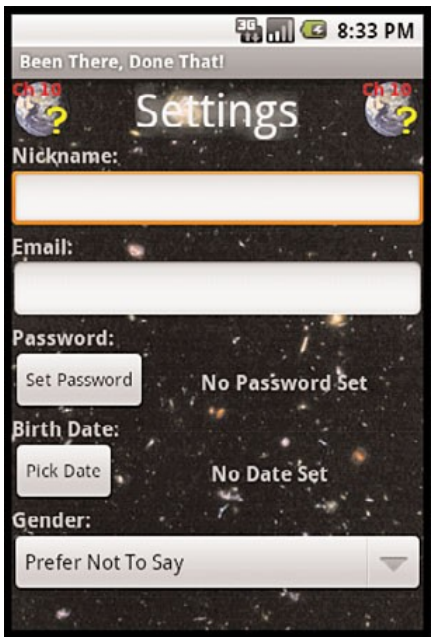

Рис. 10.8. Экран с настройками приложения «Been There, Done That!»

КСТАТИ

В некоторых версиях инструментария Android SDK может также потребоваться реализовать соответствующие «заглушки» для других методов класса `AdapterView.OnItemSelectedListener`.

СОХРАНЕНИЕ ДАННЫХ ФОРМЫ С ИСПОЛЬЗОВАНИЕМ КЛАССА SHAREDPREFERENCES

Для сохранения настроек приложения вы можете использовать механизм постоянного хранения данных, реализуемый классом `SharedPreferences`. При помощи этого механизма вы можете сохранить все значения, указанные пользователем в форме на экране с настройками.

Определение записей для экземпляра класса SharedPreferences

Ранее вы добавили в базовый класс `QuizActivity` строку для настроек вашей игры:

```
public static final String GAME_PREFERENCES = "GamePrefs";
```

Теперь вам нужно добавить в класс `QuizActivity` название каждого параметра, значение которого вы хотите сохранить в настройках:

```
public static final String GAME_PREFERENCES_NICKNAME = "Nickname";
// Тип String

public static final String GAME_PREFERENCES_EMAIL = "Email";
//Тип String

public static final String GAME_PREFERENCES_PASSWORD = "Password";
```

```
//Тип String
public static final String GAME_PREFERENCES_DOB = "DOB";
//Тип Long

public static final String GAME_PREFERENCES_GENDER = "Gender";
//Тип Int
```

Сохранение настроек параметров с использованием экземпляра класса SharedPreferences

Теперь, когда вы определили названия параметров, вы можете сохранить в настройках игры любые значения этих параметров, вводимые пользователем. Внутри класса `QuizSettingsActivity` сначала нужно объявить переменную-член, представляющую экземпляр класса `SharedPreferences`:

```
SharedPreferences mGameSettings;
```

Внутри метода `onCreate()` деятельности вы инициализируете эту переменную-член следующим образом:

```
mGameSettings =
getSharedPreferences(GAME_PREFERENCES, Context.MODE_PRIVATE);
```

Вы передаете и указанный метод название вашего экземпляра класса `SharedPreferences` (строковая константа с именем `GAME_PREFERENCES`, которую вы создали в классе `QuizActivity`). Режим `MODE_PRIVATE` представляет стандартное разрешение, используемое для закрытых файлов приложения.

Теперь, в любом месте кода, где нужно сохранить какую-либо настройку игры, вы просто получаете экземпляр класса `SharedPreferences.Editor`, присваиваете нужному параметру желаемое значение и сохраняете произведенное изменение. Например, чтобы сохранить информацию из элемента `EditText` параметра **Nickname** (Ник), сначала вы получаете введенный пользователем текст, используя метод `getText()` элемента `EditText`:

```
final EditText nicknameText =
   (EditText) findViewById(R.id.EditText_Nickname);
String strNickname = nicknameText.getText().toString();
```

После того, как вы получите значение типа `String` из поля ввода, представленного элементом `EditText`, вы можете сохранить это значение в экземпляре класса `SharedPreferences.Editor`, используя метод `putString()`:

```
Editor editor = mGameSettings.edit();
editor.putString(GAME_PREFERENCES_NICKNAME, strNickname);
editor.commit();
```

Настройки параметров **Nickname** (Ник), **Email** и **Password** (Пароль) могут быть сохранены в виде значений типа **string,** а настройки параметров **Date of Birth** (Дата рождения) и **Gender** (Пол) — в виде значений типа `long` и `integer` соответственно. Чтобы сохранить значения этих параметров, вы должны получить введенное пользователем значение из соответствующего элемента, при необходимости преобразовать это значение, и затем сохранить его с использованием методов `putLong()` и `putInt()` класса `Shared Preferences.Editor`.

Пока вы можете сохранить введенные пользователем значения в поля ввода **Nickname** (Ник), **Email** и **Gender** (Пол). С параметрами **Date of Birth** (Дата рождения) и **Password** (Пароль) вы будете работать в следующем часе, когда будете реализовывать диалоговые окна **DatePickerDialog** (для ввода даты) и **Dialog** (для ввода пароля). Если вы вернетесь к исходному коду класса `QuizSettingsActivity` и найдете строки кода, отмеченные комментариями `TODO`, вы увидите, где должно происходить сохранение значений параметров.

Чтение значений параметров из экземпляра класса SharedPreferences

Приступив к сохранению значений параметров с использованием механизмов постоянного хранения данных, вам понадобится способ для чтения сохраненных значений и загрузим их в поля ввода формы (для последующего редактирования). Для этого вам нужно получить доступ к настройкам игры и проверить. существует ли среди этих настроек нужный вам параметр. Например, вы можете захотеть проверить, было ли указано значение для параметра Nickname (Ник), и, если значение было указано, загрузить это значение в элемент `EditText` с именем `EditTextNickname`. Для этого вы можете воспользоваться методами `contains()` и `getString()` класса `SharedPreferences`:

```
final EditText nicknameText =
    (EditText) findViewById(R.id.EditText_Nickname);
if (mGameSettings.contains(GAME_PREFERENCES_NICKNAME)) {
    nicknameText.setText(mGameSettings.getString(
        GAME_PREFERENCES_NICKNAME, ""));
}
```

В этом коде вы проверяете существование параметра с именем, определяемым константой `GAME_PREFERENCES_NICKNAME`, в экземпляре класса `SharedPreferences`, используя метод `contains()`. Если метод `contains()` возвращает значение `true`, вы извлекаете значение данного параметра (значение типа `String`) из экземпляра класса `SharedPreferences`, используя метод `getString()`.

Параметры **Nickname** (Ник), **Email** и **Password** (Пароль) представляются строковыми значениями и могут извлекаться при помощи метода `getString()`. Вместе с тем значение параметра **Date of Birth** (Дата рождения) должно извлекаться при помощи метода `getLong()`, а для получения значения параметра **Gender** (Пол) требуется использование метода getInt().

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Настройки приложения хранятся в файловой системе Android в виде XML-файлов. К файлам настроек можно обращаться с использованием панели **File Explorer** (Проводник) перспективы DDMS среды разработки Eclipse. Файлы, создаваемые экземплярами класса `SharedPreferences`, находятся в следующем каталоге:
`/data/data/<имя пакета>/shared_prefs/<имя файла настроек>.xml`.

Наконец для отладочных целей вы переопределяете метод `onDestroy()` класса `QuizSettingsActivity` таким образом, чтобы при уничтожении экрана с настройками происходило журналирование всех текущих значений параметров:

```
@Override
protected void onDestroy() {
   Log.d(DEBUG_TAG, "SHARED PREFERENCES");
   Log.d(DEBUG_TAG, "Nickname is: "
   + mGameSettings.getString(GAME_PREFERENCES_NICKNAME, "Not set"));
   Log.d(DEBUG_TAG, "Email is: "
   + mGameSettings.getString(GAME_PREFERENCES_EMAIL, "Not set"));
   Log.d(DEBUG_TAG, "Gender (M=1, F=2, U=0) is: "
   + mGameSettings.getInt(GAME_PREFERENCES_GENDER, 0));
   // We are not saving the password yet
   Log.d(DEBUG_TAG, "Password is: "
   + mGameSettings.getString(GAME_PREFERENCES_PASSWORD, "Not set"));
   // We are not saving the date of birth yet
   Log.d(DEBUG_TAG, "DOB is: "
   + DateFormat.format("MMMM dd, yyyy", mGameSettings.getLong(
   GAME_PREFERENCES_DOB, 0)));
   super.onDestroy();
}
```

Теперь всякий раз, когда будет происходить уничтожение экземпляра класса `QuizSettingsActivity` (например, когда пользователь нажимает кнопку **Back** на телефоне), сохраненные настройки параметров будут отображаться на панели **LogCat**.

ИТОГИ

В этом часе вы добавили форму на экран с настройками приложения-викторины «Been There, Done That!». Форма содержит разнообразные параметры, включая поля ввода различных типов, реализуемых с использованием элементов `EditText`, и раскрывающийся список, реализуемый при помощи элемента `Spinner`. Также вы сохранили свободное пространство на экране, воспользовавшись двумя элементами `Button`, которые в будущем будут использованы для отображения диалоговых окон Dialog. Наконец, вы реализовали простейший механизм загрузки и сохранения настроек игры при помощи класса `SharedPreferences`.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Почему бы не воспользоваться обычными кнопками **Save** (Сохранить) и **Cancel** (Отменить), которые используются на веб-формах?

Ответ: Формы для ввода данных в приложениях, функционирующих на мобильных устройствах, действительно могут быть разработаны с применением этого традиционного подхода, но во внимание должны приниматься накладные расходы, связанные с механизмом управления состояниями. (События жизненного цикла деятельности, например пауза и восстановление, могут требовать немедленного сохранения и восстановления введенных данных.). Распространенный подход при разработке форм для ввода данных для мобильных устройств основан на сохранении значений полей формы по мере ввода данных пользователем.

Вопрос: Элемент **Spinner** должен заполняться данными исключительно из массива?

Ответ: Нет, данные, используемые элементом `Spinner`, могут загружаться из различных источников при помощи адаптера данных. Например, содержимое элемента `Spinner` может загружаться из базы данных.

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Элементы `EditText` наследуются от элементов `TextView`.

2. Какие типы кнопок доступны в платформе Android?
   A. `Button`.
   B. `TextButton`.
   C. `ImageButton`.

3. Верно ли это? Данные типа `Calendar` можно непосредственно сохранить в экземпляре класса `SharedPreferences`.

Ответы

1. Верно. Класс `TextView` вместе с его известными атрибутами и методами, например, методами `getText()` и `setText()`, — это базовый класс для класса `EditText`.

2. A и B. В платформе Android существует два типа кнопок: элемент Button представляет простую кнопку с надписью. Элемент `ImageButton` представляет кнопку с графическим и изображением типа `Drawabe`.

3. Неверно. Классом `SharedPreferences` поддерживаются следующие типы данных: `Boolean`, `float`, `int`, `long` и `String`. Для сохранения даты или времени можно рассмотреть вариант использования значений типа `long` (выраженных в миллисекундах с начала Эпохи).

Упражнения

1. Добавьте отображение всплывающего уведомления в обработчике событий нажатий для элемента `Button` параметра **Password** (Пароль). Всплывающее уведомление с текстом «Clicked!» (Нажато!) должно отображаться после того, как пользователь нажмет на элемент `Button`.

2. Измените каждый элемент `EditText` так, чтобы сохранение его значения происходило при нажатии кнопки вверх (`KEYCODE_DPAD_UP`) или кнопки вниз (`KEYCODE_DPAD_DOWN`) многопозиционного джойстика, помимо нажатия клавиши Enter (`KEYCODE_ENTER`).

3. Реализуйте кнопку **Clear** (Очистить), которая будет удалять все настройки игры, используя метод `clear()` класса `SharedPreferences.Editor`. Не забудьте вызвать метод `commit()`.

Час 11. ИСПОЛЬЗОВАНИЕ ДИАЛОГОВЫХ ОКОН ДЛЯ СБОРА ДАННЫХ, ВВОДИМЫХ ПОЛЬЗОВАТЕЛЕМ

Вопросы, рассматриваемые в этом часе:

- **работа с диалоговыми окнами деятельностей;**
- **использование диалогового окна DatePickerDialog;**
- **обработка и форматирование информации о дате;**
- **построение пользовательских диалоговых окон.**

В этом часе вы завершите работу над экраном с настройками приложения «Been There, Done That!», добавив несколько диалоговых окон **Dialog** в класс `QuizSettingsActivity`. Каждое диалоговое окно **Dialog** предназначено для сбора данных определенного типа, вводимых пользователем. Сначала вы добавите диалоговое окно **Dialog,** реализуемое классом `DatePickerDialog` и предназначенное для ввода даты, которое позволит пользователю указать его дату рождения, а затем вы создадите собственное диалоговое окно **Dialog**, чтобы предоставить пользователю возможность изменять его пароль.

РАБОТА С ДИАЛОГОВЫМИ ОКНАМИ ДЕЯТЕЛЬНОСТЕЙ

Деятельность может использовать диалоговые окна **Dialog** для организации информации и обработки событий, возникающих в результате взаимодействия с пользователем. Например, деятельность может отобразить диалоговое окно с сообщением об ошибке или с предложением подтвердить некую операцию, например удаление некоторой информации. Использование диалоговых окон **Dialog** для решения простых задач позволяет сократить количество классов **Activity** в приложении.

Знакомство с различными типами диалоговых окон Dialog

В инструментарии Android SDK доступно несколько различных типов диалоговых окон **Dialog**, включая следующие:

- **Dialog** — базовый класс для всех типов диалоговых окон **Dialog** (рис. 11.la).

- **AlertDialog** — диалоговое окно с одним, двумя или тремя элементами `Button` (рис. 11.1 b).

- **Character PickerDialog** – диалоговое окно, позволяющее выбрать символ с ударением, связанный с базовым символом (рис. 11.1c).

- **DatePickerDialog** — диалоговое окно с элементом `DatePicker` (рис 11.1d).

- **ProgressDialog** — диалоговое окно с элементом `ProgressBar`, представляющим индикатор непрерывного хода выполнения процесса или индикатор хода выполнения процесса на основе значений (рис. 11.1e)

- **TimePickerDialog** — диалоговое окно с элементом `TimePicker` (рис. 11.1f)

Рис. 11.1. Различные типы диалоговых окон **Dialog**, доступных в платформе Android

Если ни один из существующих типов диалоговых окон **Dialog** не подходит для решения поставленной задачи, вы можете создать ваши собственные диалоговые окна Dialog, отвечающие специфическим требованиям к макету. Пользовательские диалоговые окна Dialog будут рассмотрены далее в этом часе.

Отслеживание жизненного цикла диалогового окна деятельности

Каждое диалоговое окно **Dialog** должно быть определено внутри деятельности, в которой используется это окно. Отдельное диалоговое окно **Dialog** может открываться как единожды, так и использоваться регулярно. Понимание того, как деятельность управляет жизненным циклом диалогового окна **Dialog**, имеет решающее значение для правильной реализации диалогового окна **Dialog**. Давайте рассмотрим ключевые методы, которые должны использования деятельностью для управления диалоговым окном **Dialog:**

- Метод `showDialog()` применяется для отображения диалогового окна **Dialog.**

- Метод `dismissDialog()` используется для прекращения отображения диалогового окна **Dialog** на экране. Диалоговое окно **Dialog** продолжает оставаться в пуле диалоговых окон **Dialog** данной деятельности. При повторном отображении диалогового окна **Dialog** на экране с применением метода `showDialog()` будет использована кэшированная версия этого окна.

- Метод `removeDialog()` применяется для удаления диалогового окна Dialog из пула диалоговых окон **Dialog** данной деятельности. Диалоговое окно **Dialog** больше не будет доступно для дальнейшего использования. При повторном вызове метода `showDialog()` диалоговое окно **Dialog** необходимо создавать снова.

ОПРЕДЕЛЕНИЕ ДИАЛОГОВОГО ОКНА

Диалоговое окно, используемое в деятельности, должно быть определено заранее. Каждое диалоговое окно **Dialog** имеет свой идентификатор (целое число). При вызове метода `showDialog()` вы передаете этот идентификатор диалогового окна **Dialog** в качестве параметра данного метода. После этого происходит вызов метода `onCreateDialog()`, который возвращает экземпляр диалогового окна подходящего типа.

Вы должны переопределить метод `onCreateDialog()` деятельности и вернуть экземпляр подходящего диалогового окна в соответствии с заданным идентификатором. Если в деятельности используется несколько диалоговых окон **Dialog,** в методе `onCreateDialog()` может применяться инструкция switch для возврата экземпляра подходящего диалогового окна **Dialog** на основании переданного параметра — идентификатора диалогового окна **Dialog.**

ИНИЦИАЛИЗАЦИЯ ДИАЛОГОВОГО ОКНА

Поскольку диалоговое окно **Dialog** может сохраняться в пуле диалоговых окон **Dialog** деятельности, существенное значение может иметь возможность повторной инициализации диалогового окна при каждом его отображении, а не только при создании. С этой целью вы можете переопределить метод `onPrepareDialog()` деятельности.

В то время как метод `onCreateDialog()` может быть вызван лишь единожды для инициализации диалогового окна при его создании, метод `onPrepareDialog()` вызывается каждый раз при вызове метода `showDialoq()`, предоставляя деятельности возможность выполнить повторную инициализацию диалогового окна всякий раз, когда это окно показывается пользователю.

ОТОБРАЖЕНИЕ ДИАЛОГОВОГО ОКНА

Вы можете отобразить любое диалоговое окно, определенное в деятельности, вызвав метод `showDialog()` этой деятельности и передай **в** него существующий идентификатор диалогового окна Dialog — идентификатор, который будет распознан методом `onCreateDialog()`.

СОКРЫТИЕ ДИАЛОГОВОГО ОКНА

Для большинства типов диалоговых окон **Dialog** определены условия, при которых эти диалоговые окна будут автоматически скрыты. Тем не менее, если вы хотите самостоятельно скрыть диалоговое окно, вы можете просто вызвать метод `dismissDialog()` и передать в него идентификатор скрываемого диалогового окна **Dialog.**

УДАЛЕНИЕ ДИАЛОГОВОГО ОКНА

Сокрытие диалогового окна **Dialog** не приводит к уничтожению этого окна. При повторном отображении диалогового окна используется его кэшированное содержимое. Если вы

хотите, чтобы деятельность удалила диалоговое окно и в дальнейшем не использовала его, вызовите метод `removeDialog()` и передайте в него существующий идентификатор диалогового окна **Dialog.**

ИСПОЛЬЗОВАНИЕ ДИАЛОГОВОГО ОКНА DATEPICKERDIALOG

Теперь вы должны добавить диалоговое окно **Dialog** в класс `QuizSettingsActivity`. В частности, вы должны добавить диалоговое окно **DatePickerDialog**, которое предназначено для ввода даты рождения пользователя. Добавление диалогового окна **DatePickerDialog** в класс `QuizSettingsActivity` включает несколько шагов.

1. Определить уникальный идентификатор для диалогового окна в классе деятельности.

2. Реализовать метод `onCreateDialog()` деятельности, который будет возвращать экземпляр диалогового окна **DatePickerDialog** при указании уникального идентификатора, определенного на шаге 1.

3. Реализовать метод `onPrepareDialog()` деятельности, который будет инициализировать диалоговое окно **DatePickerDialog** с использованием выбранной даты рождения или текущей даты.

4. Отобразить диалоговое окно **DatePickerDialog** при помощи метода `showDialog()`, передав в него уникальный идентификатор диалогового окна **Dialog**.

Добавление диалогового окна DatePickerDialog в класс QuizSettingsActivity

Чтобы счищать диалоговое окно **DatePickerDialog**, вы должны сначала объявить уникальный идентификатор, который будет представлять данное диалоговое окно, как показано в следующем коде:

```
static final int DATE_DIALOG_ID = 0;
```

Затем вам нужно реализовать метод `onCreateDialog()` класса `QuizSettingsActivity` и включить инструкцию case для нового идентификатора диалогового окна **Dialog**:

```
@Override
protected Dialog onCreateDialog(int id) {
   switch (id) {
   case DATE_DIALOG_ID:
        // TODO: Return a DatePickerDialog here
        Using DatePickerDialog 185
        }
   return null;
}
```

Теперь давайте подробно рассмотрим, как создать экземпляр диалогового окна **DatePickerDialog**. Внутри инструкции switch для идентификатора DATE_ DIALOG_ID вы должны вернуть экземпляр диалогового окна **DatePickerDialog** для его последующего отображения. Конструктор диалогового окна **DatePickerDialog** имеет параметр `DatePickerDialog.OnDateSetListener`, используя который, вы можете определить реализацию метода `onDateSet()`, вызываемого в тот момент, когда пользователь выберет свою дату рождения, благодаря чему вы сможете сохранить эту дату в экземпляре класса `SharedPreferences`:

```
DatePickerDialog dateDialog =
    new DatePickerDialog(this,
        new DatePickerDialog.OnDateSetListener() {
            public void onDateSet(DatePicker view, int year,
                int monthOfYear, int dayOfMonth) {
                Time dateOfBirth = new Time();
                dateOfBirth.set(dayOfMonth, monthOfYear, year);
                long dtDob = dateOfBirth.toMillis(true);
                dob.setText(DateFormat
                    .format("MMMM dd, yyyy", dtDob));
                Editor editor = mGameSettings.edit();
                editor.putLong(GAME_PREFERENCES_DOB, dtDob);
                editor.commit();
        }
        }, 0, 0, 0);
```

Элемент DatePicker состоит из трех отдельных элементов для ввода месяца, дня и года. Таким образом, чтобы создать экземпляр диалогового окна **DatePickerDialog**, вы должны указать значения для всех этих элементов. Поскольку диалоговое окно **DatePickerDialog** может отображаться произвольное число раз, нет необходимости инициализировать дату, отображаемую в этом диалоговом окне, в методе `onCreateDialog()`, поэтому для инициализации используются значения по умолчанию (три нуля). Наконец, вы возвращаете новый экземпляр диалогового окна `DatePickerDialog`, созданный в инструкции `switch` метода `onCreateDialog()`:

```
return dateDialog;
```

Инициализация диалогового окна DatePickerDialog

Диалоговое окно **DatePickerDialog** должно инициализироваться всякий раз, когда оно отображается на экране, поэтому вы переопределяете метод `onPrepareDialog()`, чтобы установить для элемента `DatePicker` либо сегодняшнюю дату, либо дату рождения пользователя, сохраненную в настройках игры.

ВНИМАНИЕ!

Чтобы получить текущую дату на устройстве, вы можете использовать класс `Calendar`. Класс `Calendar` имеет отдельные поля для каждой «части» даты — дня, месяца и года. Вы можете использовать эту возможность класса `Calendar`, чтобы установить определенную дату в диалоговом окне **DatePickerDialog**.

В метод `onPrepareDialog()` передается как идентификатор диалогового окна **Dialog**, так и экземпляр диалогового окна **Dialog**, поэтому при необходимости вы можете изменить этот экземпляр. Например, в данном случае вы должны обновить дату, отобра-

жаемую в диалоговом окне **DatePickerDialog**, поэтому вы используете метод `updateDate()`:

```
@Override
protected void onPrepareDialog(int id, Dialog dialog) {
    super.onPrepareDialog(id, dialog);
    switch (id) {
    case DATE_DIALOG_ID:
        // Handle any DatePickerDialog initialization here
        DatePickerDialog dateDialog = (DatePickerDialog) dialog;
        int iDay, iMonth, iYear;
        // Check for date of birth preference
        if (mGameSettings.contains(GAME_PREFERENCES_DOB)) {
            // Retrieve Birth date setting from preferences
            long msBirthDate =
            mGameSettings.getLong(GAME_PREFERENCES_DOB, 0);
            Time dateOfBirth = new Time();
            dateOfBirth.set(msBirthDate);
            iDay = dateOfBirth.monthDay;
            iMonth = dateOfBirth.month;
            iYear = dateOfBirth.year;
        } else {
            Calendar cal = Calendar.getInstance();
            // Today's date fields
            iDay = cal.get(Calendar.DAY_OF_MONTH);
            iMonth = cal.get(Calendar.MONTH);
            iYear = cal.get(Calendar.YEAR);
        }
        // Set the date in the DatePicker to the date of birth OR to
        the
        // current date
        dateDialog.updateDate(iYear, iMonth, iDay);
        return;
    }
}
```

Отображение диалогового окна DatePickerDialog

Вы настроили диалоговое окно **DatePickerDialog**, однако оно не отобразится на экране до тех пор, пока пользователь не нажмет на нужный элемент `Button` на основном экране с настройками. Пользователь открывает диалоговое окно **DatePickerDialog**, нажимая на элемент `Button` с именем `Button_DOB`.

Вы зарегистрировали обработчик событий нажатий для элемента с именем `Button_DOB`, и в настоящий момент в этом обработчике событий вызывается метод класса Toast. Теперь вы можете изменить код обработчика, чтобы вызвать метод `showDialog()` для отображения диалогового окна **DatePickerDialog**, изображенного на рис. 11.2:

```
Button pickDate = (Button) findViewById(R.id.Button_DOB);
pickDate.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        showDialog(DATE_DIALOG_ID);
    }
});
```

Рис. 11.2. Диалоговое окно **DatePickerDialog**,
используемое для ввода даты рождения пользователя

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Вы можете использовать класс `DateFormat` для получения строкового представления даты. Класс `DateFormat` позволяет работать как с датами, представленными в виде объемов класса `Calendar`, так и с датами, представленными в виде значений типа `long` (выраженными в миллисекундах с начала Эпохи). Например, чтобы представить дату, которая хранится в виде значения типа `long`, в формате «January 1,2010», вы могли бы использовать метод `format()` класса `DateFormat`, как показано ниже:

```
String strDate = DateFormat.format("MMMM dd, yyyy", dtDob);
```

РАБОТА С ПОЛЬЗОВАТЕЛЬСКИМИ ДИАЛОГОВЫМИ ОКНАМИ

Если основные типы диалоговых окон **Dialog** не удовлетворяют вашим требованиям, вы можете создать пользовательское диалоговое окно. Один из простейших способов создания пользовательского диалогового окна заключается в использовании встроенного диалогового окна **AlertDialog** и класса AlertDialog.Builder для переопределения его стандартного макета. Чтобы создать пользовательское диалоговое окно подобным образом, выполните следующие шаги:

1. Разработайте ресурс макета пользовательского диалогового окна, который будет отображаться при помощи диалогового окна **AlertDialog**.

2. Определите в деятельности идентификатор пользовательского диалогового окна Dialog.

3. Обновите метод `onCreateDialog()` деятельности, чтобы создать и вернуть экземпляр подходящего пользовательского диалогового окна **AlertDialog**.

4. Отобразите диалоговое окно, используя метод `showDialog()`.

Добавление пользовательского диалогового окна на экран с настройками

В приложении «Been There, Done That!» вы хотите использовать пользовательское диалоговое окно для ввода и подтверждения нового пароля. На рис. 11.3 изображены состояния диалогового окна (для совпадающих и несовпадающих паролей).

Пользовательское диалоговое окно, предназначенное для ввода пароля, которое вы хотите создать, должно содержать два текстовых поля, используемых для ввода и подтверждения пароля. Если два введенных пароля будут совпадать, указанный пароль будет сохранен в настройках приложения. На рис. 11.4 изображен эскиз дизайна диалогового окна для ввода пароля.

Диалоговое окно для ввода пароля — это всего лишь упрощенный вариант экрана с настройками, который содержит два элемента `EditText`. Вам также потребуется расположить элемент `TextView` под полями ввода, чтобы сообщать пользователю информацию о том, что совпадают введенные пароли или нет.

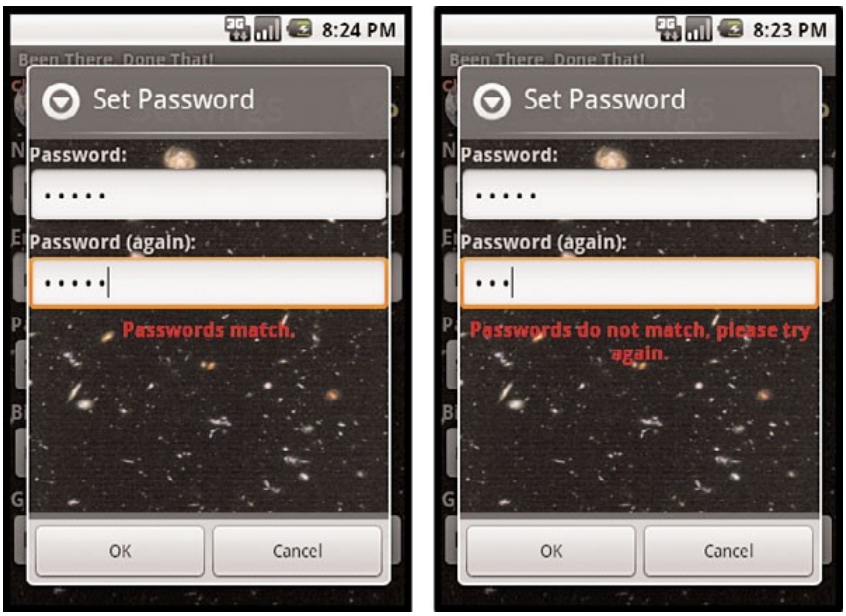

Рис. 11.3. Пользовательское диалоговое окно для ввода пароля

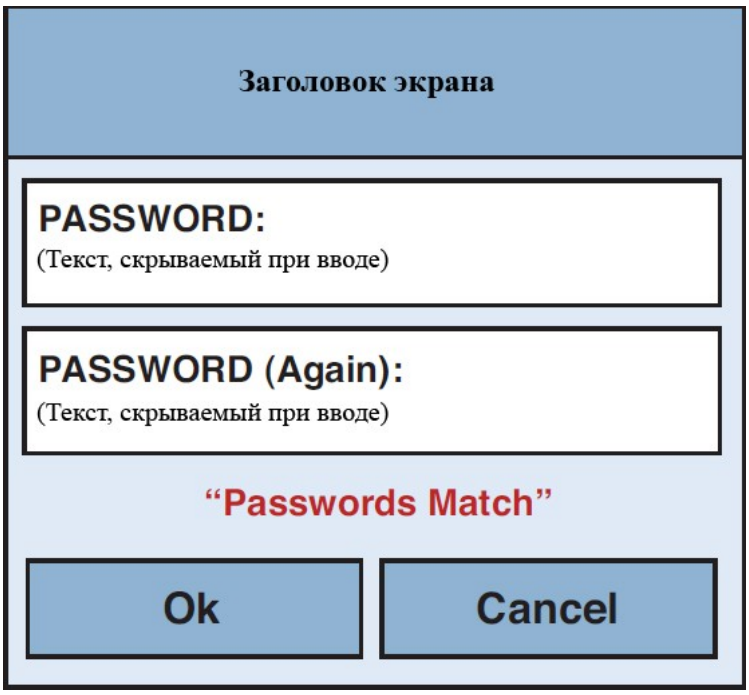

Рис. 11.4. Эскиз дизайна диалогового окна для ввода пароля приложения «Been There, Done That!»

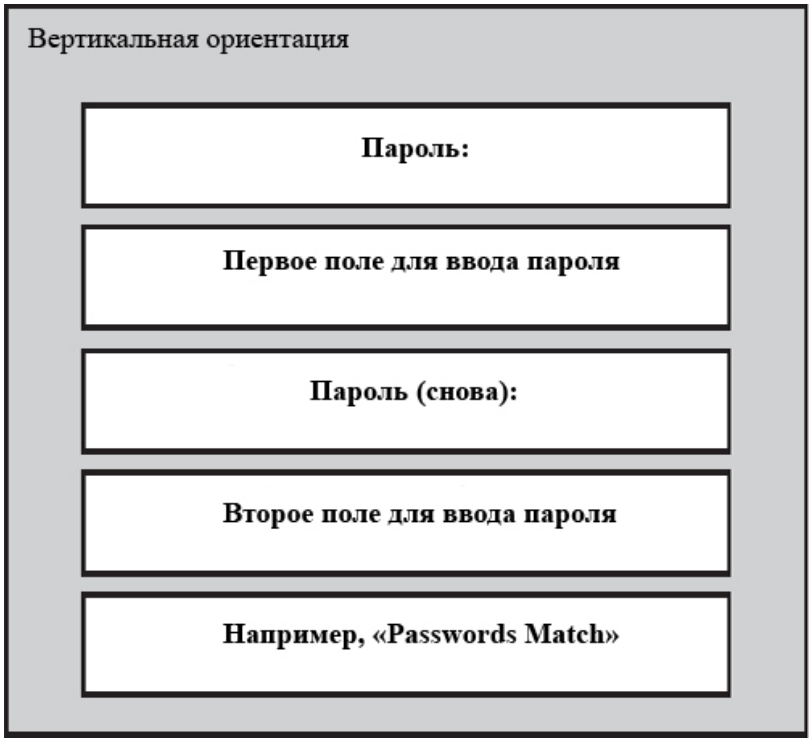

Рис. 11.5. Дизайн-макет диалогового окна для ввода пароля приложения «Been There, Done That!»

На рис. 11.5 изображен дизайн-макет диалогового окна для ввода пароля.

Вы можете воспользоваться преимуществами встроенных элементов `Button`, которые могут быть настроены для использования вместе с диалоговым окном **AlertDialog**. Нет необходимости включать три кнопки в ваш дизайн-макет.

РЕАЛИЗАЦИЯ МАКЕТА ДИАЛОГОВОГО ОКНА ДЛЯ ВВОДА ПАРОЛЯ

Теперь настало время реализовать новый макет, который будет использован в диалоговом окне для ввода пароля. Сначала вы создадите новый файл ресурса макета с именем **password_dialog.xml.** Этот файл макета определяет пользовательский интерфейс диалогового окна. Чтобы создать этот файл, выполните следующие шаги:

Откройте редактор ресурсов среды разработки Eclipse и добавьте в проект новый файл с именем **/res/layout/password_dialog.xml**.

Добавьте элемент-контейнер `LinearLayout`. Присвойте его атрибуту id значение root, а атрибуту orientation — значение `vertical`. Атрибутам `layout_width` и `layout_height` присвойте значение `fill_parent`. Все остальные элементы будут добавлены внутрь этого элемента-контейнера `LinearLayout`.

Добавьте элемент `TextView` для отображения строки «Password» (Пароль). Затем добавьте элемент `EditText` и присвойте его атрибуту `id` знамение `EditText_Pwdl`, атрибуту `maxLines` — значение `1`, а атрибуту `inputType` — значение `textPassword`.

Добавьте еще один элемент `TextView` для отображения строки «Password (Again)» (Пароль (подтверждение)). Затем добавьте еще один элемент `EditText` и присвойте его атрибуту `id` значение `EditText_Pwd2`, атрибуту `maxLines` — значение `1`, а атрибуту `inputType` — значение `textPassword`.

Наконец, добавьте элемент `TextView` с атрибутом `id`, установленным в значение `TextView_PwdProblem`, который будет использоваться для отображения информации о состоянии паролей. В этом элементе `TextView` будет отображаться сообщение о том, совпадают два введенных пароля или нет.

Теперь сохраните файл макета **password_.dialog.xml**.

ДОБАВЛЕНИЕ ДИАЛОГОВОГО ОКНА ДЛЯ ВВОДА ПАРОЛЯ В КЛАСС QUIZSETTINGSACTIVITY

Чтобы создать пользовательское диалоговое окно **AlertDialog**, вы сначала должны объявить уникальный идентификатор, представляющий это диалоговое окно, как показано ниже:

```
static final int PASSWORD_DIALOG_ID = 1;
```

Затем вам нужно обновить метод `onCreateDialog()` класса `QuizSettingsActivity`, добавив инструкцию case для нового идентификатора диалогового окна Dialog:

```
case PASSWORD_DIALOG_ID:
    // Build Dialog
    // Return Dialog
```

Теперь давайте подробно рассмотрим процесс создания диалогового окна для ввода пароля. Вы начнете с наполнения элемента `View` созданным макетом (т. е. загрузки макета в этот элемент):

```
LayoutInflater inflater =
    (LayoutInflater)
    getSystemService(Context.LAYOUT_INFLATER_SERVICE);
final View layout =
    inflater.inflate(R.layout.password_dialog,
    (ViewGroup) findViewById(R.id.root));
```

Чтобы загрузить файл макета **password_dialog.xml** н элемент `View`, вы должны получить экземпляр класса `LayoutInflater` и кием пытать метод atop, передан в качестве параметра идентификатор ресурса макета, а также идентификатор корневого элемента макета (в данном случае — элемент-контейнер `LinearLayout` с именем root, содержащий остальные элементы диалогового окна).

После того, как макет будет загружен в элемент view, его можно изменять программным путем, как и обычный макет. Теперь можно заполнять элементы данными и регистрировать слушателей событий.

Например, чтобы получить экземпляр элемента с именем `EditText_Pwd1` из переменной типа `View` с именем `layout`, можно использовать метод `findViewByid()`, как показано в следующем коде:

```
final EditText p1 =
    (EditText) layout.findViewById(R.id.EditText_Pwd1);
final EditText p2 =
    (EditText) layout.findViewById(R.id.EditText_Pwd2);
```

Теперь можно зарегистрировать необходимых слушателей событий для элементов `EditText`, например, тех, которые использовались ранее для наблюдения за вводимыми значениями в элементах `EditText` и сравнения строк в процессе ввода значений пользователем.

Например, вы можете зарегистрировать слушателя для событий изменения текста во втором элементе `EditText`, используя класс `TextWatcher`, и сравнивать содержимое этого поля ввода с содержимым первого элемента `EditText`. После этого вы можете отобразить результат сравнения паролей в третьем элементе `TextView` с именем `TextView_PwdProblem`, который мы создали ранее:

```
final TextView error =
    (TextView) layout.findViewById(R.id.TextView_PwdProblem);
p2.addTextChangedListener(new TextWatcher() {
    @Override
    public void afterTextChanged(Editable s) {
        String strPass1 = p1.getText().toString();
        String strPass2 = p2.getText().toString();
        if (strPass1.equals(strPass2)) {
        error.setText(R.string.settings_pwd_equal);
        } else {
            error.setText(R.string.settings_pwd_not_equal);
```

```
        }
    }
    // ... other required overrides do nothing
});
```

Класс `TextWatcher` имеет несколько методов, которые нужно реализовать. Тем не менее, нас больше всего интересует метод `afterTextChanged()`.

Теперь, когда вы заполнили элемент `View` ресурсом макета и настроили его для дальнейшего использования, вы можете присоединить его к диалоговому окну **AlertDialog**. Для этого используется класс `AlertDialog.Builder`:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setView(layout);
builder.setTitle(R.string.settings_button_pwd);
```

Сначала вы устанавливаете в качестве элемента-представления экземпляра класса `AlertDialog.Builder` загруженный вами макет. Затем вы устанавливаете заголовок диалогового окна Dialog при помощи метода `setTitle()`.

Ваше диалоговое окно будет иметь два элемента `Button`: кнопку положительного ответа (**OK**) и кнопку отрицательного ответа (**Cancel**). Поскольку вы не хотите, чтобы это диалоговое окно кэшировалось для дальнейшего использования деятельностью, в обоих обработчиках событий для элементов `Button` должен вызываться метод `removeDialog()`, который уничтожает диалоговое окно:

```
QuizSettingsActivity.this
    .removeDialog(PASSWORD_DIALOG_ID);
```

Кнопка положительного ответа (**OK**) подразумевает некоторую обработку данных. Когда пользователь нажимает на нее, вам нужно извлечь текст паролей из элементов `EditText`, сравнить полученные значения и, если две строки совпадают, сохранить новый пароль в настройках приложения:

```
builder.setPositiveButton(android.R.string.ok,
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
        TextView passwordInfo =
            (TextView) findViewById(R.id.TextView_Password_Info);
        String strPassword1 = p1.getText().toString();
        String strPassword2 = p2.getText().toString();
        if (strPassword1.equals(strPassword2)) {
            Editor editor = mGameSettings.edit();
            editor.putString(GAME_PREFERENCES_PASSWORD,
                strPassword1);
            editor.commit();
            passwordInfo.setText(R.string.settings_pwd_set);
        } else {
            Log.d(DEBUG_TAG, "Passwords do not match. "
                + "Not saving. Keeping old password (if set).");
            }
            QuizSettingsActivity.this
            .removeDialog(PASSWORD_DIALOG_ID);
        }
```

```
    });
```

Кнопка отрицательного ответа, **Cancel**, просто возвращает пользователя на основной экран с настройками. Вы настраиваете поведение кнопки отрицательного ответа при помощи метода `setNegativeButton()` класса `Builder`:

```
builder.setNegativeButton(android.R.string.cancel,
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            QuizSettingsActivity.this
                .removeDialog(PASSWORD_DIALOG_ID);
        }
    });
```

Когда ваше диалоговое окно будет полностью сконфигурировано при помощи класса `Builder`, вы вызываете метод `create()`, чтобы создать и вернуть экземпляр пользовательского диалогового окна **AlertDialog**:

```
AlertDialog passwordDialog = builder.create();
return passwordDialog;
```

ОТОБРАЖЕНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ДИАЛОГОВОГО ОКНА ДЛЯ ВВОДА ПАРОЛЯ

Пользовательское диалоговое окно, как, например, ваше диалоговое окно для ввода пароля, отображается таким же способом, что и обычное диалоговое окно, — при помощи метода `showDialog()` деятельности. На экране с настройками приложения «Been There, Done That!» пользователь выбывает открытие диалогового окна для ввода пароля, нажимая на элемент `Button` с именем `Button_Password`. Таким образом, вы можете зарегистрировать обработчик событий нажатия для этого элемента `Button` и отображать диалоговое окно для ввода пароля соответствующим образом:

```
Button setPassword = (Button) findViewById(R.id.Button_Password);
setPassword.setOnClickListener(new View.OnClickListener() {
   public void onClick(View v) {
       showDialog(PASSWORD_DIALOG_ID);
   }
});
```

На рис. 11.6 изображен готовый экран с настройками, включая элементы для отображения диалоговых окон Dialog.

Рис. 11.6. Готовый экран с настройками приложения «Been There, Done That!»

ИТОГИ

В этом часе вы ознакомились с тем, как деятельность может использовать диалоговые окна **Dialog** для упрощения функциональности экрана и его макета на примере экрана с настройками приложения «Been There, Done That!». Диалоговое окно может использоваться для отображения необходимой информации и виде всплывающего окна. Существуют стандартные типы диалоговых окон **Dialog** для ввода даты, времени и специальных символов, а также вспомогательные диалоговые окна **Dialog** для отображения индикаторов хода выполнения процесса или предупреждений. Кроме того, вы также можете создавать пользовательские диалоговые окна **Dialog**.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Каким образом сохраняется информация о диалоговых окнах в деятельности?

Ответ: Каждая деятельность имеет пул используемых диалоговых окон **Dialog**, и именно из этого пула будет извлекаться диалоговое окно **Dialog** при его повторном отображении. По существу, диалоговые окна отображаются при помощи метода `showDialog()`, после чего они помещаются в пул. При закрытии диалоговое окно остается в пуле до тех пор, пока не будет уничтожена деятельность или пока не будет явно вызван метод `removeDialog()`.

Вопрос: Как определить, из какой деятельности было вызвано диалоговое окно **Dialog**?

Ответ: Чтобы определить родительскую деятельность определенного диалогового окна **Dialog**, вы можете использовать метод `getOwnerActivity()` класса **Dialog**.

ПРАКТИКУМ
Контрольные вопросы

1. Какой класс может быть использован внутри деятельности для создания всплывающих окон?

   A. Popup.
   B. ActivityWindow.
   C. Dialog.

2. Верно ли что? Вы можете использовать одно и то же диалоговое окно многократно, если макет этого диалогового окна не изменяется.

3. Верно ли что? Только определенные макеты могут быть использованы для таких диалоговых окон, как `Alert` и `ContinueOrCancel`.

Ответы

1. B. Класс `Dialog` и производные от него классы могут быть использованы внутри деятельности для создания всплывающих окон при помощи методой `onCreateDialog()` и `showDialog()`.

2. Верно. Но для отображения друг их данных в диалоговом окне вам потребуется переопределить внутри деятельности метод `onPrepareDialog()`.

3. Неверно. Вы можете использовать любой макет для вашего диалогового окна.

Упражнения

1. Обновите метод `onDataSet()` класса `DatePickerDialog` таким образом, чтобы введенная пользователем дата рождения отображалась в элементе `TextView` на основном экране с настройками.

2. Обновите диалоговое окно для ввода пароля таким образом, чтобы в элементе `TextView` на основном экране с настройками отображалась информация о состоянии пароля (указан или не указан).

Час 12. РЕАЛИЗАЦИЯ ЛОГИКИ ПРИЛОЖЕНИЯ

Вопросы, рассматриваемые в этом часе:

- **разработка дизайна игрового экрана; работа с элементами** `ViewSwitcher`**;**
- **структуры данных и разбор XML-данных;**
- **реализация логики игры и хранение состояния игры.**

В этом часе вы реализуете главный экран приложения «Been There, Done Thai!» — игровой экран. Этот экран предлагает пользователю ответить на ряд вопросов опроса-викторины и сохраняет информацию о набранных им очках. Поскольку на этом экране должна отображаться динамическая последовательность изображений и текстовых строк, вы можете воспользоваться несколькими новыми элементами-представлениями `View`, включая элементы `imageSwitcher` и `TextSwitcher`, которые помогут реализовать переходы между вопросами викторины. Кроме того, *вы* также сможете реализовать в классе `QuizGameActivity` логику игры и добавить информацию о промежуточном состоянии игрового процесса, включая получение наборов новых вопросов по мере прохождения игры пользователем.

РАЗРАБОТКА ДИЗАЙНА ИГРОВОГО ЭКРАНА

Игровой экран должен позволять пользователю отвечать на вопросы викторины и отслеживать количество положительных ответов (набранные очки). Каждый вопрос викторины имеет соответствующее изображение, которое должно отображаться на экране. Например, на игровом экране может отображаться картинка горы, и пользователю будет задан вопрос, совершал ли он когда-либо восхождение на гору.

В отличие от предыдущих разработанных вами экранов, игровой экран не должен иметь заголовка. Вместо этого предполагается использовать все пространство экрана для отображения компонентов игры. На рис. 12.1 изображен эскиз дизайна игрового экрана.

Рис. 12.1. Эскиз дизайна игрового экрана приложения «Been There, Done That!»

Желательно сделать так, чтобы игровой экран обладал некоторыми общими чертами, присущими остальной части приложения: в нем должно использоваться то же фоновое изображение, шрифт и цветовая схема, которые используются на других экранах. Чтобы преобразовать эскиз дизайна в соответствующий дизайн-макет, вам понадобится обновить файл макета **/res/ tayout/game.xml** и класс `QuizGameActivity`.

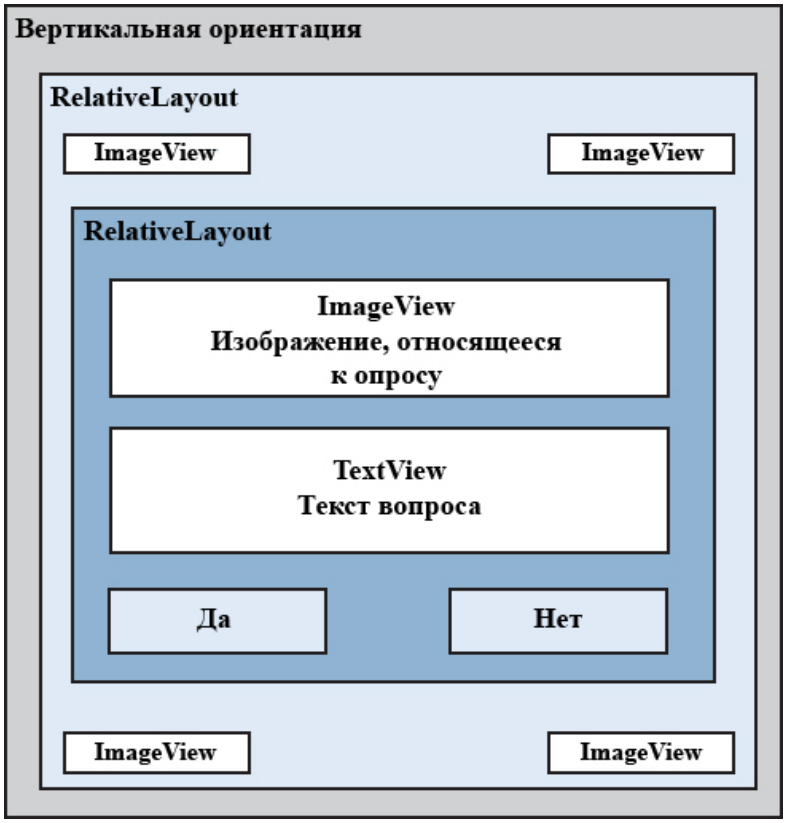

Рис. 12.2. Дизайн-макет игрового экрана приложения «Been There, Done That!»

Элемент-контейнер `RelativeLayout` отлично подойдет для отображения значков в каждом из четырех углов экрана. Кроме того, вы также можете использовать еще один элемент-контейнер `RelativeLayout` для отображения вопросов, внутри которого будут размещены: один элемент `ImageView`, один элемент `TextView` и два элемента `Button` для отображения ответов пользователя.

На рис 12.2 изображен первоначальный вариант дизайн-макета игрового экрана.

Каждый раз, когда пользователь нажимает на один из элементов `Button`, происходит обновление элементов `ImageView` и `TextView` на игровом экране для отображения следующего вопроса. Для плавного (и анимированного) перехода от одного вопроса к другому вы можете использовать специальные элементы `ImageSwitcher` и `TextSwitcher`, которые наследуются от класса `ViewSwitcher`.

Элемент `ViewSwitcher` позволяет анимировать переход между двумя дочерними элементами-представлениями `View`: текущим элементом `View` и элементом `View`, который нужно отобразить на экране. Одновременно на экране может отображаться только один элемент `View`, однако во время перехода между элементами `View` могут

использоваться анимации, например, плавные появления/исчезновения и вращения. Дочерние элементы `View` создаются при помощи экземпляра класса `ViewFactory`. Например, элемент `ImageSwitcher` и его соответствующий экземпляр класса `ViewFactory` могут быть использованы для генерации элемента `ImageView` текущего вопроса и для последующего переключения на элемент `ImageView` следующего вопроса, когда пользователь щелкнет по одному из элементов `Button`. Подобным образом, элемент `TextSwitcher` имеет два дочерних элемента-представления `TextView`, при переключении между которыми анимация применяется к тексту.

На рис. 12.3 изображен обновленный дизайн-макет игрового экрана, в котором используются элемент `ImageSwitcher` и элемент `TextSwitcher`.

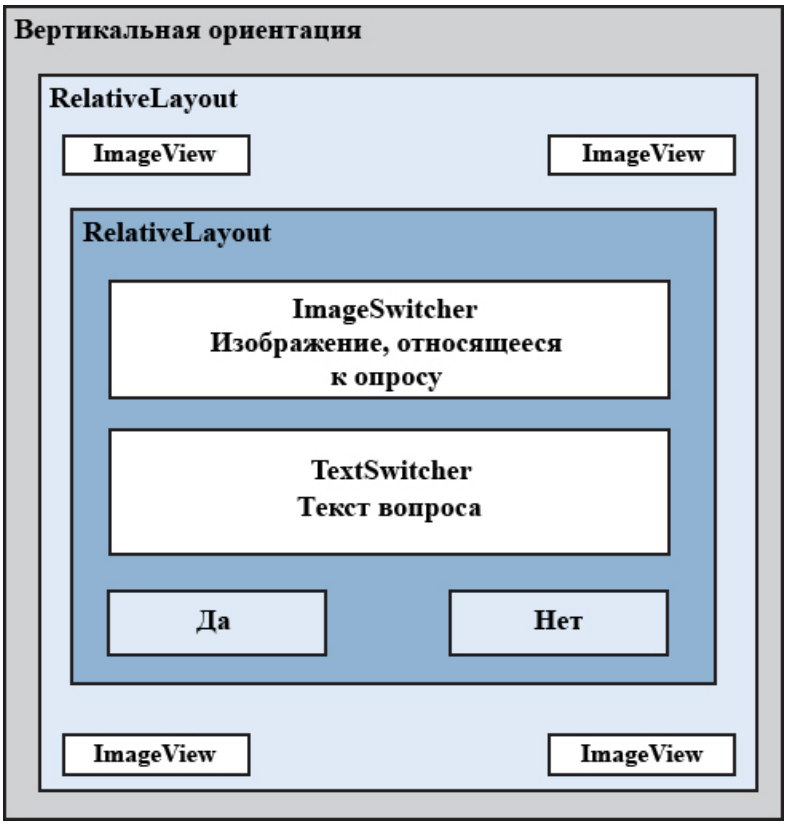

Рис. 12.3. Измененный дизайн-макет игрового экрана приложения «Been There, Done That!» с элементами ImageSwitcher и TextSwitcher

РЕАЛИЗАЦИЯ МАКЕТА ИГРОВОГО ЭКРАНА

Реализацию игрового экрана вы начнете с добавления новых ресурсов в проект. Затем вы обновите файл макета **game.xml**, чтобы отразить подготовленный дизайн-макет игрового экрана.

Добавление новых ресурсов в проект

Для реализации игрового экрана вам потребуется несколько новых ресурсов:

- строковые ресурсы (типа `string`) для текста, который будет отображаться на элементах `Button`, а также для текста, который будет отображаться в случае отсутствия доступных вопросов;
- разнообразные ресурсы размеров (типа `Dimension`) и цветов (типа `Color`), необходимые для реализации дизайна элементов игрового экрана;

- два ресурса с XML-данными, которые будут содержать наборы тестовых вопросов.

Вы уже без особого труда должны уметь создавать любые ресурсы типа `String`, `Dimension` и `Color`, необходимые для макета экрана, поэтому давайте поговорим о наборах тестовых вопросов.

Со временем вопросы, используемые в приложении «Been There, Done That!», будут загружаться с сервера в Интернете. Тем не менее пока вы создадите два набора тестовых вопросов, которые будут храниться локально в виде XML-файлов: **/res/xml/samplequestions.xml** и **/res/xml/smaplequestions2.xml**. В дальнейшем, когда вы реализуете сетевую поддержку в вашем приложении, эти XML-данные будут загружаться с удаленного сервера. Используя наборы тестовых вопросов, вы можете прямо сейчас заняться реализацией логики игры, не беспокоясь о функциональности, связанной с сетевым взаимодействием.

Но зависимо от источника получения списки вопросов - локальная файловая система или удаленный сервер - XML-данные будут выглядеть одинаково. Вот пример этих данных:

```
<?xml version="1.0" encoding="utf-8"?>
<!— This is a mock question XML chunk —>
<questions>
    <question
        number="1"
        text=
           "Have you ever been on an African safari?"
        imageUrl=
           "http://www.perlgurl.org/Android/BeenThereDoneThat/Questions/q1.png"
    />
    <question
        number="2"
        text=
           "Have you ever climbed a mountain?"
        imageUrl=
           "http://www.perlgurl.org/Android/BeenThereDoneThat/Questions/q2.png"
    />
    <question
        number="3"
        text=
           "Have you ever milked a cow?"
        imageUrl=
           "http://www.perlgurl.org/Android/BeenThereDoneThat/Questions/q3.png"
/>
</questions>
```

Как можно заметить, структура XML-данных очень проста. В ней присутствует один тег с именем <questions>, который может содержать несколько тегов <question>. Каждый тег <question> имеет три атрибута: идентификатор вопроса (number), текст вопроса **(**text) и адрес URL изображения, связанного с данными (imageUrl). Обратите внимание, что вы используете изображения с удаленных источников в Интернете вместо того, чтобы добавлять все изображения для опросов в качестве ресурсов приложения.

Обновление макета игрового экрана

Файл макет **game.xml** определяет пользовательский интерфейс игрового экрана. Как и раньше, откройте редактор ресурсов среды разработки Eclipse и удалите все существующие элементы из макета. Далее выполните следующие шаги, чтобы создать желаемый макет на базе ранее подготовленного дизайн-макета экрана.

1. Добавьте новый элемент-контейнер LinearLayout и присвойте его атрибуту background значение @drawable/bkgrnd. Все остальные элементы будут добавляться внутрь этого элемета-котейнера LinearLayout;

2. Добавьте элемент-контейнер RelativeLayout и присвойте сто атрибутам layout_width и layout_height значение wrap_content.

3. Добавьте четыре элемента ImageView внутрь элемента-контейнера RelativeLayout — но одному элементу для каждого угла экрана. Присвойте атрибуту src каждого элемента ImageView значение @drawable/quizicon. Присвойте атрибуту id каждого элемента ImageView уникальные значения @+id/ImageView_Header, @+id/ImageView_Header2, @+id/ImageView_Header3 и @+ id/ImageView_Header4.

4. Найдите элемент ImageView с атрибутом id, установленным в значение ImageView_Header, и присвойте его атрибутам layout_alignParentLett и layout_alignParentTop значение true.

5. Найдите элемент ImageView с атрибутом id, установленным в значение ImageView_Header2, и присвойте его атрибутам layout_ alignParentRight и layout_alignParentTop значение true.

6. Найдите элемент ImageView с атрибутом id, установленным в значение ImageView_Header3, и присвойте его атрибутам layout_alignParentLeft и layout_alignParentBottom значение true.

7. Найдите элемент ImageView с атрибутом id, установленным в значение ImageView_Header4, и присвойте его атрибутам layout_alignParentRight и layout_alignParentBottom значение true.

8. Добавьте еще один элемент-контейнер RelativeLayout внутрь существующего элемента-контейнера RelativeLayout, расположив его под элементом ImageView, чтобы создать область для вопросов викторины. Присвойте его атри-

буту id значение @+id/RelativeLayout_Content. Присвойте атрибутам layout_width и layout_height значение wrap_content. Кроме того, также присвойте его атрибуту gravity значение center, а атрибуту layout_margin — значение 45px.

9. Внутрь только что добавленного элемента-контейнера RelativeLayout добавьте элемент ImageSwitcher с атрибутом id, установленным в значение @+id/ImageSwitcher_QuestionImage. Присвойте его атрибутам layout_width и layout_height значение wrap_content. Также присвойте его атрибутам layout_alignParentTop и layout_centerInParent значение true.

10. Добавьте момент TextSwitcher с атрибутом id, установленным в значение @+id/TextSwitcher_QuestionText, под элементом ImageSwitcher. Присвойте его атрибутам layout_width и layout_height значение wrap_content. Также присвоите его атрибуту layout_centerInParent значение true, а атрибуту layout_below — значение @+id/ ImageSwitcher_QuestionImage.

11. Добавьте элемент Button с атрибутом id, установленным в значение @+id/Button_Yes, расположив его под элементом TextSwitcher. Присвойте его атрибутам layout_width и layout_height значение wrap_content. Также присвойте его атрибутам layout_alignParentBottom и layout_alignParentLeft значение true. Присвойте атрибуту text этого элемента строковый ресурс (значение Yes) и настройте его другие атрибуты так, чтобы элемент Button стал выглядеть привлекательно.

12. Добавьте еще один элемент Button с атрибутом id, установленным в значение @+id/Button_No, расположив его под предыдущим элементом Button. Присвойте его атрибутам layout_width и layout_height значение wrap_content. Также присвойте его атрибутам layout_ alignParentBottom и layout_alignParentRight значение true. Присвойте атрибуту text этого элемента строковый ресурс (значение No) и настройте его другие атрибуты так, чтобы элемент Button стал выглядеть привлекательно.

Сохраните файл макета **game.xml**.

ВНИМАНИЕ!

Редактор ресурсов среды разработки Eclipse не отображает элементы TextSwitcher и ImageSwitcher в режиме дизайна. Просмотр результирующих элементов TextView и ImageView, генерируемых указанными элементами, должен осуществляться на эмуляторе Android. В данном случае редактор ресурсов не отражает реальный внешний вид приложения.

РАБОТА С ЭЛЕМЕНТАМИ VIEWSWITCHER

Для тех ситуаций, когда планируется, что деятельность будет постоянно обновлять содержимое элемента `View`, инструментарий Android SDK предоставляет специальный механизм, реализуемый элементом `ViewSwitcher`. Элемент `ViewSwitcher` — это эффективный способ для обновления содержимого экрана. Элемент `ViewSwitcher` имеет два дочерних элемента-представления и позволяет управлять переходом между текущим дочерним элементом-представлением и дочерним элементом-представлением, который должен отображаться следующим. Дочерние элементы-представления `View` элемента `ViewSwitcher` могут генерироваться программным путем при помощи класса `ViewFactory`.

У класса `ViewSwitcher` есть два производных класса;

- **Класс** `TextSwitcher`. Представляет элемент `ViewSwitcher`, который позволяет переключаться между двумя элементами `TextView`.

- **Класс** `ImageSwitcher`. Представляет элемент `ViewSwitcher`, который позволяет переключаться между двумя элементами `ImageView`.

И хотя в любой произвольный момент времени элемент `ViewSwitcher` может иметь только два дочерних элемента-представления, последовательно он может отображать любое количество элементов-представлений `View`. Класс `ViewFactory` генерирует содержимое следующего элемента-представления, так что элементы `ImageSwitcher` и `TextSwitcher` могут последовательно отображать изображения и тексты вопросов.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Вы можете создать пользовательский элемент `ViewSwitcher`. реализовав ваш собственный класс, производный от класса `ViewSwitcher`.Использование класса `ViewFactory` для генерации элементов-представлений `View` элемента `ViewSwitcher`.

Использование класса ViewFactory для генерации элементов-представлений View элемента ViewSwitcher

При создании экземпляра элемента `ViewSwitcher` вы можете указать подходящий класс `ViewFactory`, используя метод setFactory(). Класс `ViewFactory` имеет всего один обязательный метод — метод `makeView()`. Этот метод должен возвращать элемент-представление View подходящего типа. Например, класс `ViewFactory` для элемента `TextSwitcher` должен возвращать правильно настроенный элемент-представление `TextView`, а класс `ViewFactory` для элемента `ImageSwitcher` — правильно настроенный элемент-представление `ImageView`.

Ниже представлена реализация класса `ViewFactory` для элемента `ImageSwitcher`, которая может быть использована для генерации изображений к каждому вопросу на игровом экране:

```
private class MyImageSwitcherFactory implements
ViewSwitcher.ViewFactory {
    public View makeView() {
        ImageView imageView = new ImageView(QuizGameActivity.this);
        imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
        imageView .setLayoutParams(new ImageSwitcher.LayoutParams(
            LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
        return imageView ;
    }
}
```

Обратите внимание, что источник, или содержимое, элемент-представления не устанавливается в методе `makeView()`. Вместо этого, вы можете рассматривать возвращаемый элемент-представления как шаблон для элемента `ViewSwitcher`, который будет применяться для отображения каждого дочернего элемента-представления.

Создавая экземпляр элемента `ViewSwitcher`, вы можете указать используемый им класс `VlewFactory` при помощи метода `setFactory()`. Например, чтобы указать в качестве класса `ViewFactory` для элемента `ImageSwitcher` созданный вами класс `MyImageSwitcherFactory`, вы можете сделать следующее:

```
ImageSwitcher questionImageSwitcher =
    (ImageSwitcher) findViewById(R.id.ImageSwitcher_QuestionImage);
questionImageSwitcher.setFactory(new MyImageSwitcherFactory());
```

Подобным образом вы должны создать производный класс от класса `ViewFactory`, который будет генерировать элементы-представления `TextView` для каждого вопроса на игровом экране. Ниже представлена реализация класса `MyTextSwitcherFactory`, производного от класса `ViewFactory`, который делает именно то, что нужно:

```
private class MyTextSwitcherFactory implements
ViewSwitcher.ViewFactory {
   public View makeView() {
        TextView textView = new TextView(QuizGameActivity.this);
        textView.setGravity(Gravity.CENTER);
        Resources res = getResources();
        float dimension = res.getDimension(R.dimen.game_question_size);
        int titleColor = res.getColor(R.color.title_color);
        int shadowColor = res.getColor(R.color.title_glow);
        textView.setTextSize(dimension);
        textView.setTextColor(titleColor);
        textView.setShadowLayer(10, 5, 5, shadowColor);
        return textView;
   }
}
```

Обратите внимание, что, как и в случае с реализацией класса `MyImageSwitcherFactory`, в классе `MyTextSwitcherFactory` также реализуется метод `makeView()` – на этот раз данный метод генерирует подходящий элемент-представление `TextView` с заданными значениями атрибутов размера текста, цвета и притяжения.

Работа с элементом TextSwitcher

Элемент TextSwitcher позволяет деятельности анимировать переход между двумя элементами-представлениями TextView. Вам нужно добавить элемент TextSwitcher с именем TextSwitcher QuestiorT^x* на макет игрового экрана для отображения вопросов викторины пользователю.

ИНИЦИАЛИЗАЦИЯ ЭЛЕМЕНТА TEXTSWITCHER

Для инициализации элемента TextSwitcher вы просто указываете подходящую реализацию класса ViewFactory и затем используете метод setCurrentText(), как показано в следующем коде:

```
TextSwitcher questionTextSwitcher = (TextSwitcher)
    findViewById(R.id.TextSwitcher_QuestionText);
questionTextSwitcher.setFactory(new MyTextSwitcherFactory());
questionTextSwitcher.setCurrentText("First Text String");
```

ОБНОВЛЕНИЕ СОДЕРЖИМОГО ЭЛЕМЕНТА TEXTSWITCHER

Когда вы хотите обновить содержимое элемента TextSwitcher, добавив новый элемент-представление TextView, вы можете вызвать метод setText():

```
TextSwitcher questionTextSwitcher = (TextSwitcher)
    findViewById(R.id.TextSwitcher_QuestionText);
questionTextSwitcher.setText("Next Text String");
```

Вызов метода setText() заставляет экземпляр класса MyTextSwitcherFactory сгенерировать новый элемент-представление TextView и заполнить его содержимым типа String, переданным в качестве параметра метода setText().

Работа с элементом ImageSwitcher

Элемент ImageSwitcher позволяет деятельности анимировать переход между двумя элементами-представлениями ImageView. Вы добавили элемент ImageSwitcher с именем ImageSwitcher_QuestionText на макет игрового экрана для отображения изображения, связанного с каждым вопросом викторины, пользователю.

ИНИЦИАЛИЗАЦИЯ ЭЛЕМЕНТА IMAGESWITCHER

Для инициализации элемента ImageSwitcher вы просто указываете подходящую реализацию класса ViewFactory и затем используете один из трех методов, чтобы установить изображение. В следующем коду продемонстрировано использование метода setImageDrawable():

КСТАТИ

К сожалению, в настоящее время использовать метод setlmageURL() элемента ImageSwitcher для указания адресов URL ресурсов, размещенных в Интернете, невозможно. Вместо этого вам нужно проделать дополнительную работу, связанную с загрузкой изображения по указанному адресу URL и сохранением этого изображения в виде объекта типа Drawable. Представленная ниже реализация метода getQuestionImageDrawable() делает именно то, что нужно:

```
private Drawable getQuestionImageDrawable(int questionNumber) {
Drawable image;
URL imageUrl;

try {
// Create a Drawable by decoding a stream from a remote URL
imageUrl = new URL(getQuestionImageUrl(questionNumber));
Bitmap bitmap = BitmapFactory.decodeStream(imageUrl.openStream());
image = new BitmapDrawable(bitmap);
} catch (Exception e) {
Log.e(DEBUG_TAG, "Decoding Bitmap stream failed.");
image = getResources().getDrawable(R.drawable.noquestion);
}
return image;
}
```

Метод getQuestionImageUrl() — это простой вспомогательный метод, который получает подходящий веб-адрес изображения для указанного вопроса. Эта информация хранится в объекте типа Hashtable с вопросами. (Мы поговорим о том, как обрабатываются вопросы, далее.) Полная реализация метода getQuestionlmageUrl() доступна в исходном коде на диске, прилагаемом к данной книге.

Класс URL используется для хранения удаленного адреса файла изображения в формате PNG, которое вы хотите загрузить в элемент ImageSwitcher. После этого вы выгружаете полученные данные в объект типа BitmapDrawable. Наконец, чтобы использовать методы для работы с потоком, приложению требуется разрешение android.permission.INTERNET, которое должно быть добавлено в файл манифеста Android.

ОБНОВЛЕНИЕ СОДЕРЖИМОГО ЭЛЕМЕНТА IMAGESWITCHER

Когда вы хотите обновить содержимое элемента ImageSwitcher, добавив новый элемент-представление ImageView, вы можете вызвать метод setImageDrawable():

```
ImageSwitcher questionImageSwitcher =
    (ImageSwitcher) findViewById(R.id.ImageSwitcher_QuestionImage);
Drawable image = getQuestionImageDrawable(nextQuestionNumber);
questionImageSwitcher.setImageDrawable(image);
```

Вызов метода `setlmageDrawable()` заставляет экземпляр класса `MyImageSwitcher Factory` сгенерировать новый элемент-представление irageView с объектом типа `Drawable`, переданным в качестве параметра метола `setImageDrawable()`.

Использование анимации для элемента ViewSwitcher

Чтобы анимировать переход между дочерними элементами-представлениями `View` элемента `ViewSwitcher`, применяются методы `setInAnimation()` и `setOutAnimation()`. Например, чтобы добавить анимацию плавного появления и плавного исчезновения элементов-предстаатений в элементе `TextSwitcher`, вы могли бы загрузить и использовать встроенные анимации платформы Android, как показано в следующем коде:

```
Animation in = AnimationUtils.loadAnimation(this,
android.R.anim.fade_in);
Animation out = AnimationUtils.loadAnimation(this,
android.R.anim.fade_out);
TextSwitcher questionTextSwitcher =
    (TextSwitcher) findViewById(R.id.TextSwitcher_QuestionText);
questionTextSwitcher.setInAnimation(in);
questionTextSwitcher.setOutAnimation(out);
```

Теперь при каждом вызове метода `setText()` или метода `setCurrentText()` элемента `TextSwitcher` будет запускаться анимация, имитирующая плавное появление или исчезновение текста. Вы можете еще больше улучшить процесс перехода между вопросами, добавив предыдущие анимации к элементу `ImageSwitcher`, который используется для отображения изображений, связанных с вопросами.

РЕАЛИЗАЦИЯ ЛОГИКИ ИГРЫ

Приложение «Been There, Done That!» основано на расширяемом списке вопросов викторины. И именно поэтому вы не можете хранить все вопросы в виде ресурсов, а должны придумать простой способ получения новых вопросов на лету. Кроме того, храня весь список вопросов викторины на удаленном сервере, вы оптимизируете приложение, функционирующее на мобильном телефоне, путем экономии дискового пространства.

В окончательной версии приложения вы будете получать новые наборы вопросов из Интернета. Пока же вы можете загрузить несколько наборов вопросов из XML-файлов. хранящихся в локальной файловой системе. Приложение может хранить используемый набор вопросов в памяти, а новые наборы вопросов могут загружаться по мере необходимости. Чтобы реализовать логику игры для игрового экрана, выполните следующие шаги:

1. Добавьте параметры, представляющие текущее состояние игры, в экземпляр класса `SharedPreferences`.

2. Реализуйте получение и разбор наборов вопросов викторины (в формате XML) в подходящую структуру данных.

3. Реализуйте обработку событий нажатий по элементу Button, чтобы обновлять содержимое элементов `ImageSwitcher` и `TextSwitcher`, а также управлять логикой игры.

4. Обработайте крайние случаи, например, ситуацию, когда больше нет доступных вопросов.
5. В следующих разделах эти шаги описываются более подробно.

ДОБАВЛЕНИЕ ПАРАМЕТРОВ, ХРАНЯЩИХ ТЕКУЩЕЕ СОСТОЯНИЕ ИГРЫ, В ЭКЗЕМПЛЯР КЛАССА SHAREDPREFERENCES

Чтобы отслеживать текущее состояние игры, вы должны добавить два дополнительных параметра типа `Integer` в экземпляр класса `SharedPreferences` приложения: промежуточный результат игры и текущий номер вопроса. Для добавления этих параметров вы сначала должны объявить строковые значения с названиями параметров в классе `QuizActivity`:

```
public static final String GAME_PREFERENCES_SCORE = "Score";
public static final String GAME_PREFERENCES_CURRENT_QUESTION =
"CurQuestion";
```

Затем вы объявляете объект типа `SharedPreferences` в качестве переменной-члена класса `QuizGameActivity`:

```
SharedPreferences mGameSettings;
```

Переменная-член `mGameSettings` инициализируется в методе `onCreate()` класса `QuizGameActivity`:

```
mGameSettings = getSharedPreferences(GAME_PREFERENCES,
Context.MODE_PRIVATE);
```

Теперь, по мере необходимости, и любом месте класса *ни* можете /'.пользовать объект типа `SharedPreferences` для чтения и сохранения состояния игры, а именно номера текущего вопроса и промежуточною результата игры. Например, вы можете получить номер текущего («опроса, используя метод `getInt()` класса `SharedPreferences`, как показано ниже:

```
int startingQuestionNumber =
    mGameSettings.getInt(GAME_PREFERENCES_CURRENT_QUESTION, 0);
```

ПОЛУЧЕНИЕ, РАЗБОР И ХРАНЕНИЕ ДАННЫХ С ВОПРОСАМИ ВИКТОРИНЫ

Когда у приложения «Been There, Done That!» заканчиваются вопросы для отображения, оно пытается получить новый набор вопросов. Выбранная нами архитектура упрощает реализацию сетевой поддержки в приложении, рассматриваемую в следующих часах, поскольку процедура разбора XML- данных останется прежней.

Каждый набор вопросов представлен простым XML-файлом, содержимое которого должно быть разобрано. Вы можете сохранить текущий набор вопросов в памяти, используя простую, но очень мощную структуру данных — в данном случае, это переменная-член типа `Hashtable`.

ОБЪЯВЛЕНИЕ ПОЛЕЗНЫХ СТРОКОВЫХ ЛИТЕРАЛОВ ДЛЯ РАЗБОРА ВОПРОСОВ

Уделите немного времени, чтобы изучить формат XML-данных, используемый для хранения наборов вопросов, который был рассмотрен ранее. Чтобы разобрать наборы вопросов, вам потребуется добавить несколько строковых литералов, представляющих XML-теги и атрибуты, в класс `QuizActivity`:

```
public static final String XML_TAG_QUESTION_BLOCK = "questions";
public static final String XML_TAG_QUESTION = "question";
public static final String XML_TAG_QUESTION_ATTRIBUTE_NUMBER =
"number";
public static final String XML_TAG_QUESTION_ATTRIBUTE_TEXT = "text";
public static final String XML_TAG_QUESTION_ATTRIBUTE_IMAGEURL =
"imageUrl";
```

И, раз уж вы работаете с этим файлом, можете также определить размер набора вопросов по умолчанию, чтобы упростить процесс выделения памяти для хранения вопросов в процессе разбора XML-данных:

```
public static final int QUESTION_BATCH_SIZE = 15;
```

СОХРАНЕНИЕ ТЕКУЩЕГО НАБОРА ВОПРОСОВ В ОБЪЕКТЕ ТИПА HASHTABLE

Теперь внутри класса `QuizGameActivity` вы можете реализовать простой вспомогательный класс с именем `Question`, который будет использоваться им представления каждого вопроса викторины:

```
private class Question {
    int mNumber;
    String mText;
    String mImageUrl;
    public Question(int questionNum, String questionText, String
        questionImageUrl) {
        mNumber = questionNum;
        mText = questionText;
        mImageUrl = questionImageUrl;
    }
}
```

Вы не будете хранить все вопросы локально. Вместо этого вы будете подгружать наборы вопросов по мере необходимости (пока из локальных тестовых XML-файлов, в

дальнейшем — из Интернета). Вам необходимо место для хранения этих вопросов, поэтому объявите переменную-член типа `Hashtable` внутри класса `QuizGameActivity`, которая будет хранить набор объектов типа `Question` в памяти после завершения разбора XML-данных:

```
Hashtable<Integer, Question> mQuestions;
```

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Инструментарий Android SDK включает множество классов, широко используемых при разработке приложений на платформе Java. Например, в пакете j**ava.util** вы найдете многие знакомые вам структуры данных (например, класс Hashtable) и вспомогательные классы, а в пакете **android.util** доступны дополнительные специализированные классы.

Вы можете создать экземпляр класса Hashtable и присвоить его соответствующей переменной-члену в методе onCreate () класса QuizGameActivity, как показано в следующем коде:

```
mQuestions = new Hashtable<Integer, Question>(QUESTION_BATCH_SIZE);
```

Теперь, предположив, что у нас есть некие XML-данные, представляющие новый набор вопросов, мы можем создать объект типа `XmlResourceParser` с именем `questionBatch`. Объект типа `XmlResourceParser` может быть использован для извлечения данных для каждого вопроса и сохранения этих данных в виде экземпляра класса Question в переменной-члене типа Hashtable при помощи `put()`, как показано ниже:

```
String questionNumber =
    questionBatch.getAttributeValue(null,
    XML_TAG_QUESTION_ATTRIBUTE_NUMBER);
Integer questionNum =
    new Integer(questionNumber);
String questionText =
    questionBatch.getAttributeValue(null,
    XML_TAG_QUESTION_ATTRIBUTE_TEXT);
String questionImageUrl =
    questionBatch.getAttributeValue(null,
    XML_TAG_QUESTION_ATTRIBUTE_IMAGEURL);

// Save data to our hashtable
mQuestions.put(questionNum,
    new Question(questionNum, questionText, questionImageUrl));
```

Вы можете проверить существование определенного вопроса в переменной- члене типа `Hashtable` по номеру этого вопроса при помощи метода `containsKey()`. Вы также можете получить определенный экземпляр класса `Question` по номеру вопроса, используя метод `get()`:

```
Question curQuestion = (Question) mQuestions.get(questionNumber);
```

ОБРАБОТКА СОБЫТИЙ НАЖАТИЙ НА КНОПКИ

Элементы Button на игровом экране используются для управления элементами ImageSwitcher и TextSwitcher. Каждый раз, когда пользователь нажимает любой из элементов Button, происходит сохранение промежуточного результата игры и обновление элементов ViewSwitcher для отображения следующего вопроса. Таким образом, элементы Button заставляют деятельность «продвигаться» вперед, перемещая пользователя по вопросам викторины.

Существует небольшое отличие между обработкой событий нажатий на кнопки **Yes** (Да) и **No** (Нет). Давайте рассмотрим метод OnClickListener.OnClick() для элемента Button, представляющего кнопку **Yes** (Да):

```
Button yesButton = (Button) findViewById(R.id.Button_Yes);
yesButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        handleAnswerAndShowNextQuestion(true);
    }
});
```

Метод View.onClick.Listener() для элемента Button, представляющего кнопку **No** (Нет), практически идентичен обработчику событий нажатий на кнопки **Yes** (Да), показанному выше. Единственное отличие заключается в том, что в метод handleAnswerAndShowNextQuestion() передается значение false. Это пользовательский метод, который вы реализуете для обработки промежуточных результатов игры и управления любой логикой, связанной с элементами ViewSwitcher.

Теперь давайте более подробно рассмотрим метод handleAnswerAndShowNextQuestion(), принимающий один параметр — boolean bAnswer. Сна- чата взгляните на следующий псевдокод, демонстрирующий то, что происходит в этом обработчике событий нажатий на элемент Button:

```
private void handleAnswerAndShowNextQuestion(boolean bAnswer) {
// Загрузить состояние игры, включая промежуточный результат
// Обновить промежуточный результат игры, если пользователь нажал
кнопку "yes"
// Загрузить следующий вопрос, обработав ситуацию, когда вопросов не
осталось
}
```

Теперь давайте последовательно разберем этот псевдокод и реализуем данный метод. Сначала вы должны получить текущее состояние игры, включая промежуточный результат игры и номер следующего вопроса, из экземпляра класса SharedPreferences:

```
int curScore =
    mGameSettings.getInt(GAME_PREFERENCES_SCORE, 0);
```

```
int nextQuestionNumber =
    mGameSettings.getInt(GAME_PREFERENCES_CURRENT_QUESTION, 1) + 1;
```

После этого вам нужно увеличить значение номера следующего вопроса, хранящегося в экземпляре класса SharedPreferences:

```
Editor editor = mGameSettings.edit();
editor.putInt(GAME_PREFERENCES_CURRENT_QUESTION, nextQuestionNumber);
```

Если пользователь щелкнул по кнопке **Yes** (Да), вам также нужно обновить промежуточный результат игры и сохранить его в экземпляре класса `SharedPreferences`:

```
if (bAnswer == true) {
    editor.putInt(GAME_PREFERENCES_SCORE, curScore + 1);
}
```

После того, как вы изменили все необходимые значения в экземпляре класса `SharedPreferences`, вы сохраняете все изменения при помощи метода `commit()` класса `Editor`:

```
editor.commit();
```

Далее вы проверяете, доступен ли в объекте типа `Hashtable` следующий вопрос. Если вам нужно получить новый набор вопросов, сделайте это сейчас:

```
if (mQuestions.containsKey(nextQuestionNumber) == false) {
    // Загружаем следующий набор вопросов
    try {
         loadQuestionBatch(nextQuestionNumber);
    } catch (Exception e) {
         Log.e(DEBUG_TAG, "Loading updated question batch
         failed", e);
    }
}
```

Наконец, вы обновляете элементы `TextSwitcher` и `ImageSwitcher`, используя текст и изображение для следующего вопроса:

```
if (mQuestions.containsKey(nextQuestionNumber) == true) {
    // Обновляем текст вопроса
    TextSwitcher questionTextSwitcher =
        (TextSwitcher) findViewById(R.id.TextSwitcher_QuestionText);
    questionTextSwitcher.setText(getQuestionText(nextQuestionNumber));

    // Обновляем изображение связанное с вопросом
    ImageSwitcher questionImageSwitcher =
        (ImageSwitcher)
        findViewById(R.id.ImageSwitcher_QuestionImage);
    Drawable image = getQuestionImageDrawable(nextQuestionNumber);
    questionImageSwitcher.setImageDrawable(image);
} else {
    handleNoQuestions();
}
```

Рис. 12.4. Игровой экран приложения «Been There, Done That!»

Когда вы запустите приложение и откроете игровой экран, он должен выглядеть наподобие экрана, изображенного на рис. 12.4.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Существует два простых способа «сбросить» текущее состояние викторины для отладочных целей. Первый метод заключается в удалении файла, используемого экземпляром класса `SharedPreferences` приложения, из файловой системы Android и перезапуске эмулятора. При помощи перспективы **DDMS** среды разработки Eclipse вы открываете каталог с данными приложениями и удаляете связанный с приложением файл, который используется экземпляром класса `SharedPreferences`. Также вы можете удалить и повторно установить приложение, используя пункт меню **Settings** (Настройки) на домашнем экране операционной системы Android. Добавление механизма сброса настроек приложения «Been There, Done That!» дается читателю в качестве упражнения в конце этого часа.

ОБРАБОТКА КРАЙНИХ СЛУЧАЕВ

Когда больше нет доступных вопросов, вы должны сообщить об этом пользователю. Этот случай обрабатывается методом `handleNoQuestions()` Сначала в качестве текста вопроса и связанного с ним изображения вы устанавливаете соответствующие значения, как показано в следующем коде:

```
TextSwitcher questionTextSwitcher =
(TextSwitcher) findViewById(R.id.TextSwitcher_QuestionText);
questionTextSwitcher.setText(getResources().getText(R.string.no_questi
ons));
ImageSwitcher questionImageSwitcher =
(ImageSwitcher) findViewById(R.id.ImageSwitcher_QuestionImage);
questionImageSwitcher.setImageResource(R.drawable.noquestion);
The Been There,
Done That!
game screen.
Также вы должны использовать эту возможность для отключения элементов:
```

```
Button yesButton =
(Button) findViewById(R.id.Button_Yes);
yesButton.setEnabled(false);
Button noButton =
(Button) findViewById(R.id.Button_No);
noButton.setEnabled(false);
```

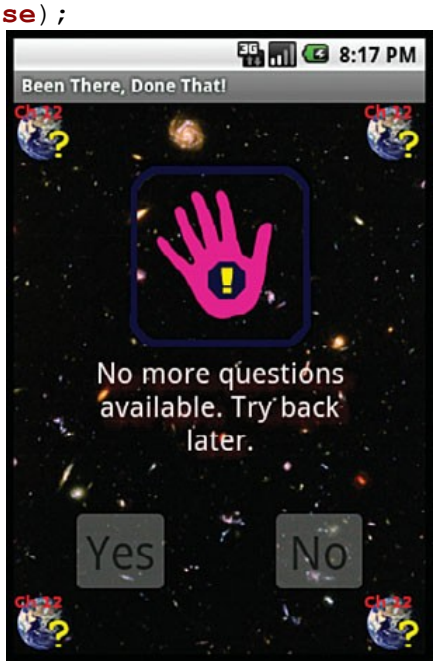

Рис. 12.5. Игровой экран приложения «Been There, Done That!», когда больше нет доступных вопросов

Когда у приложения заканчиваются вопросы, игровой экран выглядит так, как показано на рис. 12.5. Пользователю сообщается о том, что доступных вопросов больше нет и он больше не может нажимать на элементы `Button`. Вместо этого он должен нажать кнопку `Back` на мобильном телефоне и вернуться в основное меню.

ИТОГИ

В этом часе вы реализовали наиболее важный экран приложения «Been There. Done That!» — игровой экран. Вы узнали, как можно анимировать переходы между элементами-представлениями view при помощи элементов `ImageSwitcher` и `TextSwitcher`. Также вы бегло познакомились с различными структурами данных, доступными в инструментарии Android SDK, и использовали переменную-член типа `Hashtable` для хранения набора вопросов, полученных в результате разбора XML-данных. Наконец, вы использовав экземпляр класса `SharedPreferences` для хранения настроек, представляющих текущее состояние игры.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Если я храню изображения локально, могу ли я использовать метод `setImageURL()` элемента `ImageSwitcher` вместо метода `setImageDrawable()`?

Ответ: Безусловно. На самом деле, мы даже рекомендуем использовать именно этот подход. Если изображение хранится локально, использование метода `setImageURL()` существенно упрощает код, реализующий загрузку изображения в элемент `ImageSwitcher` (или `ImageView`). В данном случае нет необходимости создавать потоки или объекты типа `Drawable` в памяти.

Вопрос: Могу ли я при использовании элемента `ViewSwitcher` устанавливать свою собственную анимацию?

Ответ: Да, вы можете создать любую анимацию, которую хотите использовать для перехода между текущим элементом-представлением `View` и новым элементом-представлением `View`. Тем не менее необходимо помнить, что, как только произошел переход между элементами-представлениями `View`, единственный способ вернуться к прежнему элементу-представлению — это установить прежний элемент-представление в качестве следующего элемента-представления `View`. Для элемента `ViewSwitcher` не существует понятия «предыдущий элемент-представление `View`».

ПРАКТИКУМ
Контрольные вопросы

1. Какие классы наследуются от класса `ViewSwitcher`?

A. `TextSwitcher`;
B. `VideoSwitcher`;
C. `ImageSwitcher`;
D. `AudioSwitcher`.

2. Верно ли это? Элементы `TextView`, используемые элементом `TextSwitcher`, должны быть определены до начала использования элемента `TextSwitcher`.

3. Верно ли это? Стандартные пакеты, например java.io, java.math, java.net и java.util, доступны в инструментарии Android SDK.

Ответы

1. A и B. Класс `ViewSwitcher` имеет два производных класса; `TextSwitcher` (для анимации переходов между двумя элементами `Textview`) и `ImageSwitcher` (для анимации переходов между двумя элементами `ImageView`).

2. Неверно. Элементы `TextView`, отображаемые при помощи элемента `TextSwitcher`, могут создаваться на лету с использованием экземпляра класса `ViewFactory`.

3. Верно. Многие стандартные пакеты платформы Java доступны в инструментарии Android SDK. Также существует ряд пакетов, разработанных специально для платформы Android, которые доступны в дереве пакетов android.*.

Полный список доступных пакетов можно найти в документации к инструментарию Android SDK.

Упражнения

1. Добавьте дополнительный пункт в меню опций игрового экрана, позволяющий сбросить текущее состояние викторины. Убедитесь, что вы очищаете только необходимые настройки игры в экземпляре класса `SharedPreferences`, а не все настройки.

2. Измените приложение таким образом, чтобы вместо объекта типа `Hashtable` в нем использовалась другая структура данных, например связанный список.

ЧАСТЬ III. УЛУЧШЕНИЕ ВАШЕГО ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ МОЩНЫХ ВОЗМОЖНОСТЕЙ ПЛАТФОРМЫ ANDROID

Час 13. РАБОТА С ИЗОБРАЖЕНИЯМИ И КАМЕРОЙ

Вопросы, рассматриваемые в этом часе:

- **подготовка дизайна для использования аватара;**
- **работа с элементами** ImageButton**;**
- **запуск деятельностей и обработка результатов;**
- **работа с камерой и галереей изображений;**
- **работа с растровыми изображениями.**

В этом часе вы добавите новый функционал в приложение «Been There, Done That!» — возможность добавлять пользовательский аватар или небольшую картинку на экране с настройками. Пользователь сможет добавлять аватар двумя способами: либо при помощи камеры, встроенной в мобильный телефон, либо путем выбора изображения, которое уже сохранено на мобильном телефоне.

ПОДГОТОВКА ДИЗАЙНА ДЛЯ ИСПОЛЬЗОВАНИЯ АВАТАРА

Многие современные приложения для мобильных телефонов обладают возможностями сетевого взаимодействия и имеют ту или иную составляющую, присущую социальным сетям. Один из способов отличия одних пользователей от других состоит в использовании пользовательских значков — «аватарок». Вы можете реализовать функционал добавления аватара на экране с настройками. Аватары могут быть самыми разными: это может быть фотография лица пользователя крупным планом или забавная картинка, которая отражает его индивидуальность.

Чтобы реализовать функционал добавления аватара на экране с настройками приложения «Been There, Done That!», вам потребуется немного модифицировать дизайн экрана для размещения картинки, а также добавить некий механизм, благодаря которому пользователь сможет ее изменять. На рис. 13.1 изображен эскиз дизайна экрана, который демонстрирует, как на экране с настройками будут размещены элементы, связанные с функционалом добавления аватара.

Пространство экрана в приложениях для мобильных телефонов ценится на вес золота, и вы хотите максимально упростить использование настроек приложения «Been There, Done That!». К функционалу добавления аватара предъявляются два требования: пользователь должен иметь возможность выбирать изображение и отображать его на экране с настройками. Изо всех элементов пользовательского интерфейса платформы Android элемент `ImageButton` лучше других подходит для решения обеих задач.

Чтобы внести изменения, связанные с функционалом добавления аватара в файл макета **/res/layout/settings.xml**, вам нужно изменить область экрана, в которой размещаются элементы для параметра **Nickname** (Ник).

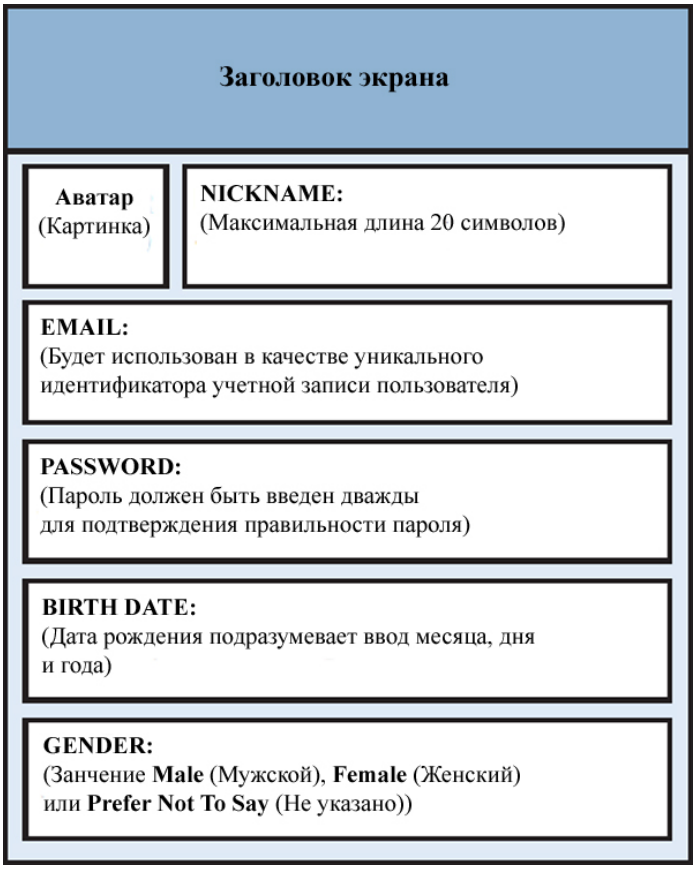

Рис. 13.1. Эскиз дизайна экрана для включения функционала добавления аватара
в приложении «Been There, Done That!»

Поскольку вы хотите добавить новый элемент слова от элементов для параметра **Nickname** (Ник), вам понадобится поместить все три элемента (элемент `ImageButton` с аватаром, элемент `TextView` с названием параметра **Nickname** (Ник) и элемент `EditText` для параметра **Nickname** (Ник)) внутрь контейнера, например `LibearLayout` (с горизонтальной ориентацией). Если поместить элементы для параметра **Nickname** (Ник) в отдельный элемент-контейнер `LinearLayout` с вертикальной ориентацией, вы получите желаемый результат. На рис. 13.2 представлены изменения дизайн-макета, необходимые для реализации функционала добавления аватара.

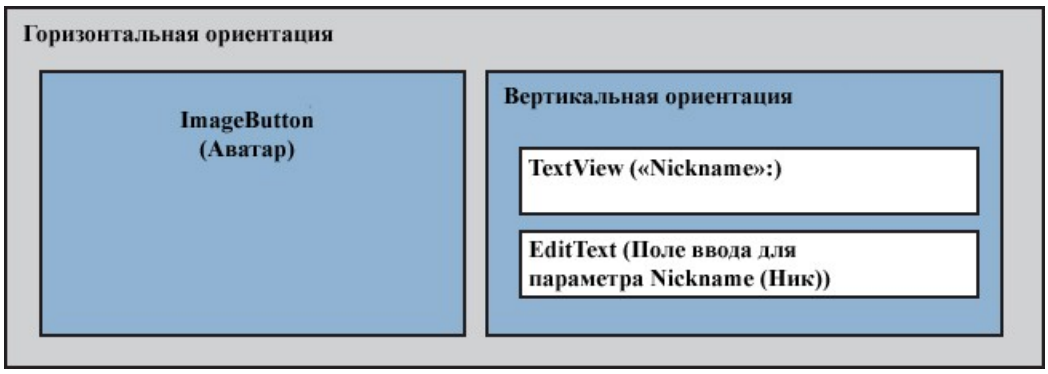

Рис. 13.2. Необходимые изменения дизайн-макета экрана с настройками для реализации функционала добавления аватара

Подождите! Вы хотите, чтобы пользователь мог настраивать свой аватар либо при помоши камеры (чтобы снять новую картинку), либо путем выбора изображения, которое уже сохранено на мобильном телефоне. Никаких проблем. Для этого достаточно обрабатывать события нажатий и продолжительных нажатий для одного и того же элемента Button. Пользователь может нажать на элемент `ImageButton`, чтобы открыть приложение для работы с камерой, либо нажать и удерживать тот же элемент `ImageButton`, чтобы запустить приложение для выбора изображений в галерее.

ДОБАВЛЕНИЕ АВАТАРА НА МАКЕТ ЭКРАНА С НАСТРОЙКАМИ

Для обновления экрана с настройками вы сначала добавите новые ресурсы в проект, а затем обновите файл макета **settings.xml**, чтобы отразить новый дизайн экрана с настройками.

Чтобы реализовать функционал, связанный с добавлением аватара, в приложении «Been There, Done That!», потребуется добавить несколько новых ресурсов, включая необходимые ресурсы типа `String`, `Dimension`, `Color` и `Drawable`. Например, вам нужно добавить новый графический ресурс, который будет использоваться в качестве аватара по умолчанию.

Обновление макета экрана с настройками

Файл макета settings.xml определяет пользовательский интерфейс экрана с настройками. Вам нужно снова открыть этот файл макета в редакторе ресурсов среды разработки Eclipse и внести следующие изменения:

Сначала найдите в файле элемент `TextView` с именем `TextView_Nickname`. Добавьте новый элемент-контейнер `LinearLayout` над этим элементом и внутри элемента `ScrollView`, а также присвойте атрибуту `orientation` нового элеента-контейнера `LinearLayout` значение `horizontal`. Присвойте его атрибутам `layout_width` и `layout_height` значение `fill_parent`.

Добавьте элемент `ImageButton` с именем `ImageButton_Avatar` внутрь добавленного элемента-контейнера `LinearLayout`. Присвойте его атрибутам `layout_width` и `layout_height` значение `wrap_content`. Вам потребуется

возможность изменять изображение аватара с сохранением соотношения его сторон, поэтому присвойте атрибуту `adjustViewBounds` значение true, а атрибуту `scaleType` – значение `fitXY`. Вы также можете присвоить атрибутам `maxHeight` и `minHeight` этого элемента определенное значение, которое будет задавать допустимые размеры изображения аватара для экрана с настройками (например, значение `75px`).

Добавьте еще один элемент-контейнер `LinearLyaout` под элементом `ImageButton`. Присвойте его атрибуту orientation значение `vertical`. Присвойте атрибутам `layout_width` и `layout_height` значение `fill_parent`. Теперь переместите элементы пользовательского интерфейса для параметра **Nickname** (Ник) (элемент `TextView` с именем `TextView_Nickname` и элемент `EditText` с именем `EditText_Nickname`) внутрь созданного элемента-контейнера `LinearLayout`.

Сохраните файл **settings.xml**. Если вы повторно запустите приложение на эмуляторе, экран с настройками будет выглядеть примерно так, как показано на рис. 13.3

РАБОТА С ЭЛЕМЕНТАМИ УПРАВЛЕНИЯ IMAGEBUTTON

Элемент ImageButton – это особый тип кнопки, на которой вместо текста отображается изображение типа Drawable. На рис. 13.3 продемонстрирован элемент ImageButton, используемый для отображения изображения аватара, как часть экрана с настройками.

Оба элемента, ImageButton и Button, наследуются от класса View, однако больше они никак не связаны между собой. Класс Button – прямой потомок класса TextView (элемент Button можно рассматривать как строку текста, отображаемую поверх фонового изображения, выглядевшего как кнопка), а класс ImageButton – прямой потомок класса ImageView.

Рис. 13.3. Экран с настройками,
на который было добавлено изображение аватара

КСТАТИ

Любое изображение, отображаемое при помощи элемента `ImageButton`, должно быть сохранено непосредственно на мобильном телефоне. Использование удаленных адресов типа Uri не рекомендуется, поскольку это может привести к снижению производительности и быстроты реакции приложения на действия пользователя.

Установка изображения для элемента ImageButton

Как и в случае с элементом `ImageView`, существует несколько различных вариантов установки изображения, отображаемого при помощи элемента `ImageButton`, включая следующие:

- `setImageBitmap()`. Используйте этот метод, чтобы указать в качестве изображения, отображаемого при помощи элемента `ImageButton`, существующий экземпляр класса Bitmap.

- `setlImageDrawable()`. Используйте этот метод, чтобы указать в качестве изображения, отображаемого при помощи элемента `ImageButton`, существующий экземпляр класса Drawable.

- `setImageResource()`. Используйте этот метод, чтобы указать в качестве изображения, отображаемого при помощи элемента `ImageButton`, существующий идентификатор ресурса.

- `setlmageURI()`. Используйте этот метод, чтобы указать в качестве изображения, отображаемого при помощи элемента ImageButton, существующий адрес типа Uri.

ВНИМАНИЕ!

В некоторых ситуациях элемент `ImageButton` кэширует отображаемое изображение и продолжает делать это, даже если вы используете один из методов изменения графики. Один из вариантов решения этой проблемы - использование `setImageURI(null)`, чтобы удалить кэшированную версию повторного вызова метода setImageURI() с аргументом Uri, содержащего адрес нового изображения, которое должно отображаться при помоги элемента `ImageButton`.

Ниже представлен удобный способ, позволяющий обращаться к ресурсам приложения, например к ресурсам типа `Drawable`, с использованием адреса типа Uri особого вида. Этот способ позволяет использовать метол `setImageURI()` элемента `ImageButton` как для графических ресурсов, так и для других изображений, хранящихся на мобильном телефоне. Обращаться к ресурсам при помощи адресов URI можно по идентификатору ресурса или по типу ресурса/его имени. Формат адреса типа Uri для варианта с использованием идентификатора ресурса выглядит следующим образом:

```
android.resource://[пакет]/[идентификатор ресурса]
```

Например, вы могли бы использовать следующий адрес типа Uri для обращения к ресурсу типа `Drawable` с именем `avatar.png` по его идентификатору ресурса:

Формат адреса типа Uri для обращения к ресурсу по его типу/имени выглядит следующим образом:

```
android.resource://[пакет]/[тип ресурса]/[имя ресурса]
```

Например, вы могли бы использовать следующий адрес типа Uri для обращения к ресурсу типа `Drawable` с именем `avatar.png` по его типу/имени:

```
Uri path = Uri.parse(
   "android.resource://com.androidbook.triviaquiz13/drawable/avatar");
```

Когда у вас есть существующий адрес типа Uri для ресурса типа `Drawable`, вы можете использовать этот адрес для вызова метода `setImageURI()` элемента `ImageButton`, как показано в следующем коде:

```
ImageButton avatarButton= (ImageButton)
findViewById(R.id.ImageButton_Avatar);
avatarButton.setImageURI(path);
```

Обработка событий элемента ImageButton

Обработка событий элемента `ImageButton`, например, событий нажатий, осуществляется точно так же, как и для любого другого элмента-представления `View` – при помощи слушателей событий нажатий. Для элемента `ImageButton`, используемого для отображения картинки аватара, вам нужно обрабатывать обычные нажатия и продолжительные нажатия.

ОБРАБОТКА НАЖАТИЙ ПРИ ПОМОЩИ МЕТОДА SETONCUCKUSTENER

Для прослушивания и обработки событий нажатий по элементу `ImageButton`, используемого для отображения аватара, вы должны реализовать класс `View.OnClickListener`:

```
avatarButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO: Launch the Camera and Save the Photo as the Avatar
    }
});
```

Этот код должен быть вам уже знаком, поскольку вы реализовали ряд слушателей для элементов `Button`.

ОБРАБОТКА ПРОДОЛЖИТЕЛЬНЫХ НАЖАТИЙ ПРИ ПОМОЩИ МЕТОДА SETONLONGCLICKLISTENER

Продолжительное нажатие — это особый тип нажатий, доступный на платформе Android. В основном событие продолжительного нажатия происходит в том случае, когда пользователь нажимает и удерживает элемент в течение приблизительно одной секунды. Этот тип нажатия обрабатывается независимо от обычного, «быстрого» нажатия.

Для обработки продолжительных нажатий вам потребуется реализовать класс `View.OnLongClickListener` и передать его в метод `setOnLongClickListener()` элемента `ImageButton`. Класс `OnLongClickListener` имеет один обязательный метод, который должен быть реализован — метод `onLongClick()`. Вот пример реализации класса `OnLongClickListener` для элемента `ImageButton`, используемого для отображения аватара:

```
avatarButton.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
    // TODO: Launch Image Picker and Save Image as Avatar
       return false;
    }
});
```

Метод `onLongClick()` во многом похож на метод `onClick()` класса `OnClickListener`. Тем не менее метод `onLongClick()` имеет возвращаемое значение, которое должно равняться значению `true`, если будут обрабатываться события продолжительных нажатий.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО

Потратьте немного времени, чтобы поработать с событиями нажатий и продолжительных нажатий для элемента `ImageButton`:

1. Откройте файл класса QuizSettingsActivity.java и добавьте слушателей событий нажатий и продолжительных нажатий для элемента с именем ImageButton_ Avatar.
2. Внутри метода onClick () класса OnLickListener добавьте отображение всплывающего уведомления с текстом Short click.
3. Внутри метода onLongClick () класса OnLongClickListener добавьте отображение всплывающего уведомления с текстом Long click.
4. Сохраните изменения и перезапустите приложение. Понажимайте на элемент ImageButton, используемому для отображения аватара, на экране с настройками и обратите внимание, в каких случаях возникают события нажатий и продолжительных нажатий.

РАБОТА С ГАЛЕРЕЕЙ ИЗОБРАЖЕНИЙ

Теперь, когда вы настроили элемент `ImageButton` для отображения аватара, можно приступать к обработке событий нажатий. Когда пользователь щелкает по кнопке с аватаром, вы хотите запускать соответствующую деятельность (при помощи интента) и затем обрабатывать результирующее изображение и сохранять его в качестве аватара.

Пока вы можете сохранить изображение аватара в локальной файловой системе мобильного телефона. Вам также потребуется добавить новую настройку в экземпляр класса `sharePreferences` приложения. Сначала вы определите эту новую настройку в классе `QuizActivity`, как показано ниже:

```
public static final String GAME_PREFERENCES_AVATAR = "Avatar";
```

Запуск деятельностей и обработке результатов

Если вернуться к часу 3, где мы рассматривали жизненный цикл приложения, вы вспомните, что для запуска деятельностей существует несколько способов – в частности, метод `startActivityForResult()`, который позволяет запускать деятельность при помощи интента и затем обрабатывать полученный результат, вызванный методом `onActivityResult()` деятельности.

Метод `startActivityForResult()` принимает два параметра: интент для запуска деятельности и код запроса, определяемый разработчиком. В данном случае вызывается деятельность `OuizSettingsActivity`. Запуск ноной деятельности должен происходить в двух случаях. Когда пользователь щелкает по элементу `ImageButton`, должна запускаться деятельность, позволяющая создать снимок при помощи камеры. Когда пользователь продолжительно щелкает по элементу `ImageButton`, должна открываться галерея изображений. Таким образом, вам нужно определить два кода запроса в классе `QuizSettingsActivity`:

```
static final int TAKE_AVATAR_CAMERA_REQUEST = 1;
static final int TAKE_AVATAR_GALLERY_REQUEST = 2;
```

Более подробно мы остановимся на методе `startActivityForResult()` далее, а пока давайте сосредоточимся на том, как класс `QuizSettingsActivity` обрабатывает результаты, возвращаемые запущенной деятельностью. Для обработки результатов, возвращаемых деятельностью, необходимо реализовать метод `onActivityResult()` класса `QuizSettingsActivity`. Поскольку определено более одного кода запроса, необходимо использовать инструкцию `switch` с двумя условиями — одно для результатов, полученных при помощи камеры, и одно для результата выбора изображения в галерее изображений:

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    switch(requestCode) {
    case TAKE_AVATAR_CAMERA_REQUEST:
        if (resultCode == Activity.RESULT_CANCELED) {
            // Avatar camera mode was canceled.
        } else if (resultCode == Activity.RESULT_OK) {
            // TODO: HANDLE PHOTO TAKEN
        }
        break;
    case TAKE_AVATAR_GALLERY_REQUEST:
```

```
            if (resultCode == Activity.RESULT_CANCELED) {
                // Avatar gallery request mode was canceled.
            } else if (resultCode == Activity.RESULT_OK) {
                // TODO: HANDLE IMAGE CHOSEN
            }
            break;
        }
    }
```

Обратите внимание, что пользователь может запустить деятельность и затем отменить ее выполнение. В данном случае параметр `resultCode` метода `onActivityResult()` будет равен значению `Activity.RESULT_CANCELED`. Когда параметр `resultCode` равен значению `Activity.RESULT_OK`, результат пригоден для дальнейшей обработки.

Поскольку в обоих случаях результатом является изображение, которое вы хотите сохранить в качестве аватара, можно создать вспомогательный метод с именем `saveAvatar()`. Этот метод будет принимать растровое изображение и сохранять его в виде локального файла. Псевдокод для метода `saveAvatar()` выглядит следующим образом:

```
private void saveAvatar(Bitmap avatar)
{
    // TODO: Save the Bitmap as a local file called avatar.jpg
    // TODO: Determine the Uri to the local avatar.jpg file
    // TODO: Save the Uri path as a String preference
    // TODO: Update the ImageButton with the new image
}
```

РАБОТА С КАМЕРОЙ

Существует множество способов интегрировать камеру мобильного телефона с вашим приложением. Вы можете реализовать поддержку камеры непосредственно в программе или же добавить существующую функциональность, используя механизм интентов.

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Для получения наиболее подробного доступа к возможностям камеры мобильного телефона вы можете использовать класс `android.hardware.Camera`, чтобы подключится к службе **Camera** на устройстве, настроить ее параметры и затем снимать фотографии и видеоклипы. Для доступа к камере мобильного телефона вашему приложению потребуется разрешение android.permission.CAMERA.

На данный момент простейший способ добавить в приложение возможность съемки фотографий при помощи камеры – запуск интента `ACTION_IMAGE_CAPTURE`. Например, вы смогли бы добавить следующий код в метод `onClick()` класса `OnClickListener` элемента `ImageButton`, используемого для отображения аватара:

```
Intent pictureIntent = new Intent(
    android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(pictureIntent, TAKE_AVATAR_CAMERA_REQUEST);
```

В эмуляторе Android отсутствует поддержка настоящей камеры. Вместо этого отображается тестовый экран камеры, и каждый раз, когда пользователь делает снимок, сохраняется предопределенное изображение. Это позволяет протестировать функциональность приложения, связанную с использованием камеры, в эмуляторе Android. Если вы запустите приложение и нажмете на элемент `ImageButton` с изображением аватара, появится экран эмулятора наподобие того, что изображен на рис. 13.4.

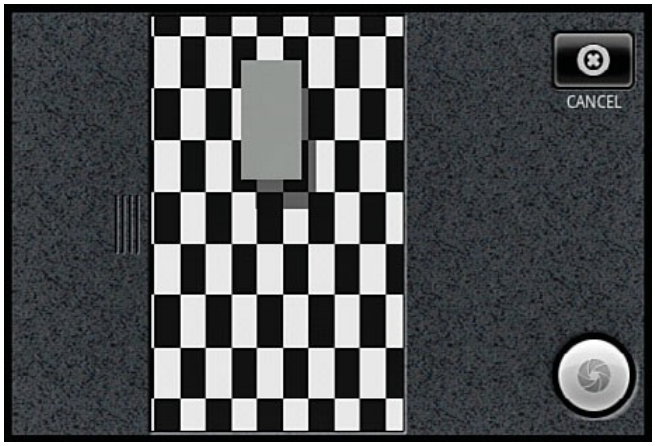

Рис. 13.4. Съемка фотографии с использованием приложения для работы с камерой в эмуляторе Android

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

При запуске «удаленной» деятельности, т.е. деятельности, не обязательно являющейся частью вашего приложения, вы фактически сообщаете следующее: «Мне нужно выполнить некую операцию. Кто может сделать это для меня?». Ряд других приложений на мобильном телефоне могут иметь возможность выполнить данную операцию. Android пытается найти наиболее подходящую деятельность для выполнения вашего запроса. Если вы хотите, чтобы пользователь мог выбрать для обработки запроса желаемую деятельность или приложение из списка допустимых, просто включите ваш интент в другой интент с именем ACTION_CHOOSER. Этот механизм зачастую используется в приложениях для повседневной работы, например для обмена сообщениями (например «Какое приложение вы хотите использовать, чтобы отправить это сообщение?»). Вы можете включить интент в другой интент, обеспечивающий выбор подходящей деятельности из списка, при помощи метода createChooser(), как показано в следующем коде:

```
Intent.createChooser(innerIntent,
"Choose which application to handle this");
```

И хотя в большинстве мобильных телефонов имеется всего одно устройство для захвата изображений, вам, как разработчику, лучше не делать подобных продложений.

Интент `ACTION_IMAGE_CAPTURE` вызывает запуск приложения для работы с камерой, предоставляет пользователю возможность сделать снимок и затем возвращает полученную фотографию. По умолчанию возвращается небольшое растровое изображение, которое вполне подойдет для вашего аватара. Используя специальную инструкцию case в методе `onActivityResult` для кода запроса `TAKE_AVATAR_CAMERA_REQUEST`, вы можете по-

лучить растровое изображение, обратившись к параметру типа intent с именем data, как показано ниже:

```
Bitmap cameraPic = (Bitmap) data.getExtras().get("data");
```

После этого вы можете передать полученное изображение в ваш вспомогательный метод `saveAvatar()`.

РАБОТА С ГАЛЕРЕЕЙ ИЗОБРАЖЕНИЙ

Платформа Android предоставляет стандартный интент с именем `ACTION_ PICK`, который позволяет пользователю выбрать некий элемент из набора элементов. Этот тип интента часто используется для адресов URI. однако он может применяться и в других ситуациях. Данный тип интента также может быть использован для получения списка всех файлов определенного MIME-типа, хранящихся на мобильном телефоне, и предоставления пользователю возможности выбора желаемого файла из этого списка.

Например, вы можете создать интент, который будет использоваться в методе `onLongClick()` для отображения пользователю всех изображений, как показано в следующем коде:

```
Intent pickPhoto = new Intent(Intent.ACTION_PICK);
pickPhoto.setType("image/*");
startActivityForResult(pickPhoto, TAKE_AVATAR_GALLERY_REQUEST);
```

Когда вы запустите приложение и выберете элемент `ImageButton` аватара, удерживая его, на экране появится галерея изображений, доступных на мобильном телефоне (рис. 13.5).

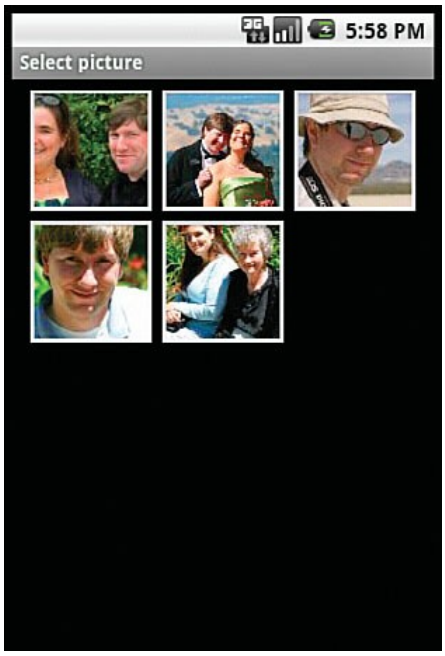

Рис. 13.5. Выбор изображения из галереи изображений на эмуляторе Android

Интент ACTION_PICK вызывает отображение галереи всех изображений, хранящихся на мобильном телефоне, позволяя пользователю выбрать одно изображение, и возвращает адрес URI, определяющий местоположение выбранного изображения. Таким образом, при помощи специальной инструкции case в методе `onActivityResult()` для кода запроса `TAKE_AVATAR_GALLERY_REQUEST` вы можете получить адрес UR! для выбранного изображения, обратившись к параметру типа `Intent` с именем `data`, как показано в следующем коде:

```
Uri photoUri = data.getData();
```

Затем, чтобы преобразовать адрес типа `Url` в соответствующий экземпляр класса Bitmap, вы можете использовать, следующий метод:

```
Bitmap galleryPic = Media.getBitmap(getContentResolver(), photoUri);
```

После этого вы можете передать полученное растровое изображение в ваш вспомогательный методе именем `saveAvatar()`.

РАБОТА С РАСТРОВЫМИ ИЗОБРАЖЕНИЯМИ

Теперь у вас есть два способа получения растрового изображения, которое будет сохранено в качестве аватара в вашем приложении. Вы можете использовать класс `Bitmap` (android.graphics.Bitmap) для создания, обработки и сохранения изображений на мобильном телефоне.

ВНИМАНИЕ!

Класс `Bitmap` инкапсулирует функциональность для работы с различными форматами растровых изображений, включая форматы PNG и JPG. Не путайте этот класс с форматом файла растровых изображений (**image.bmp**). Для создания и обработки изображений в форматах PNG и JPG на мобильных телефонах под управлением операционной системы Android также используется класс `Bitmap`.

Сохранение растровых изображений

Вы можете использовать метод `compress()` класса `Bitmap` для сохранения растровых изображений в различных формат. Например, чтобы сохранить аватар в виде JPG-файла, принадлежащего данному приложению, можно использовать следующий код:

```
String strAvatarFilename = "avatar.jpg";
avatar.compress(CompressFormat.JPEG,
    100, openFileOutput(strAvatarFilename, MODE_PRIVATE));
```

Вы можете определить адрес URI файла, используя метод `fromFile()` класса `Uri`. Например, для аватара, который вы только что сохранили при помощи метода `compress()`, можно использовать следующий код:

```
Uri imageUri = Uri.fromFile(new File(getFilesDir(),
strAvatarFilename));
```

Сохранив изображение аватара в виде файла и получив его адрес URI, вы можете обновить содержимое элемента `ImageButton`. Теперь, если вы запустите приложение и выберете изображение аватара (сделав снимок при помощи камеры или выбрав из галереи), содержимое элемента `ImageButton` будет обновлено с использованием выбранного изображения, как показано на рис. 13.6.

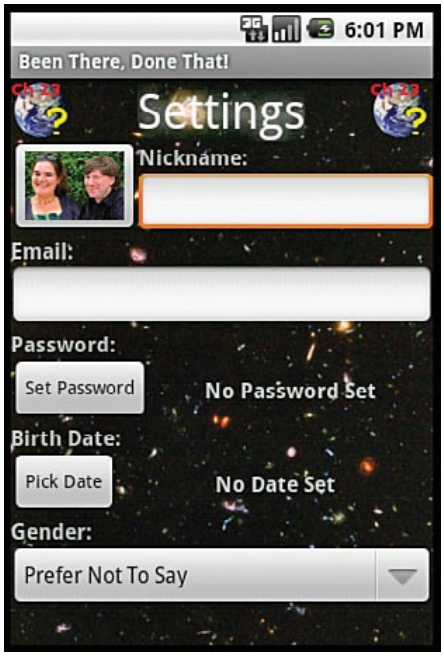

Рис. 13.6. Экран с настройками
приложения «Been There, Done That!»
на котором отображается пользовательский аватар

Масштабирование растровых изображений

Вы можете использовать метод `createScaledBitmap()` класса `Bitmap` для создания миниатюр изображений и изображений с измененными размерами.

ВНИМАНИЕ!
Вы должны правильно рассчитывать конечную ширину и высоту масштабируемого изображения, чтобы сохранить пропорции исходного растрового изображения. В противном случае масштабируемое изображение будет искажено (растянуто или сжато).

Генерация растровых изображений из различных источников

Вы можете использовать класс `BitmapFactory` (`android.graphics.BitmapFactory`), чтобы декодировать растровые объекты из различных источников, включая файлы, байтовые массивы, потоки и ресурсы.

Выполнение трансформаций растровых изображений

Вы можете выполнять различные трансформации – например, операции вращения – над растровыми объектами, используя класс `Matrix`(`android.graphics.Matrix`)

ИТОГИ

В этом часе вы реализовали новый функционал добавления аватара экране с настройками приложения «Been There. Done That!». Пользователь может создать аватар, либо сделав снимок встроенной камерой, либо выбрав существующее изображение из галереи. Вы узнали, как запускать деятельность и получать результаты ее выполнения при помощи методов `startActivityForResult()` и `onActivityResult()`. В конце урока вы познакомились с некоторыми классами `Android SDK`, предназначенными для работы с графическими изображениями.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: По умолчанию интент `ACTION_IMAGE_CAPTURE` возвращает небольшое растровое изображение фотографии, снятой на камеру. Однако оригинальное изображение, снятое камерой, имеет гораздо большие размеры. Могу ли я получить это полноразмерное изображение?

Ответ: Вы можете управлять данными, возвращаемыми приложением для работы с камерой, указав для интента некоторые дополнительные параметры (в частности, заполнив поле `EXTRA_OUTPUT`).

Вопрос: Как сохранить пропорции исходного изображения при масштабировании изображения, представленного экземпляром класса `Bitmap`?

Ответ: Чтобы сохранить пропорции изображения, просто применяйте один и тот же коэффициент к каждой оси (`x` и `y`). Не забывайте, что если вы масштабируете изображения с использованием одного и того же кода, масштаб одних из них может быть уменьшен, а других — увеличен.

ПРАКТИКУМ
Контрольные вопросы

1. С помощью какого параметра можно отличить друг от друга результаты деятельности, обрабатываемые по методу `onActivityResult()`?

A. requestCode.
B. resultCode.
C. data.

2. Верно ли это? Элемент `ImageButton` – производный от элемента `Button`.

3. Верно ли это? Класс `Bitmap` позволяет создавать только традиционные растровые изображения с расширением **.bmp**.

Ответы

1. A. Параметр roqestCode, значение которого указывается разработчиком, используется и для определения того, какая деятельность (запущенная при помощи метода `startActivityForResult()`) возвращает результат. Параметр `resultCode` предоставляет информацию об этой деятельности, например, была ли она успешно завершена или отменена пользователем.

2. Неверно. Элемент `ImageButton` на самом деле происходит от элемента `ImageView`. Тем не менее, поведение элемента `Button` во многом похоже на поведение апеменга `ImageButton`, поскольку оба они — потомки элемента-представления `View`.

3. Неверно. Класс `Bitmap` инкапсулирует функциональность для работы со всеми форматами растровых изображений — в частности, с форматами PNG (рекомендуется) и JPG (поддерживается).

Упражнения

1. Используйте класс `Bitmap`, чтобы изменить изображение аватара каким-либо образом перед тем, как сохранить его в виде файла приложения. Например, используйте класс Matrix, чтобы инвертировать все цвета в изображении (эта операция называется инверсией).

2. Используйте метод `createScaledBitmap()` класса `Bitmap`, чтобы создать масштабированную версию (миниатюру) изображения аватара.

Час 14. ДОБАВЛЕНИЕ ПОДДЕРЖКИ ГЕОЛОКАЦИОННЫХ СЕРВИСОВ

Вопросы, рассматриваемые в этом часе:

- **подготовка дизайна для функционала, связанного с указанием избранного места пользователя;**
- **использование геолокационных сервисов;**
- **использование сервисов, основанных на геокодировании;**
- **работа с картами.**

В этом часе вы добавите новый функционал в приложение «Been There. Done That!» — возможность указывать пользователю на экране с настройками выбранное им место. Это можно будет делать одним из двух способов: либо используя текущее местоположение, предоставляемое геолокационными сервисами на мобильном телефоне (LBS — location-based services), либо указав название места, которое может быть преобразовано в соответствующие GPS- координаты с использованием инструментария Android SDK.

ПОДГОТОВКА ДИЗАЙНА ДЛЯ ФУНКЦИОНАЛА, СВЯЗАННОГО С УКАЗАНИЕМ ИЗБРАННОГО МЕСТА ПОЛЬЗОВАТЕЛЯ

Мобильные пользователи находятся в постоянном движении и приложения для мобильных устройств, которые обеспечивают интеграцию с LBS- сервисами, обрели необычайную популярность. Платформа Android упрощает добавление поддержки LBS-сервисов в приложения. Степень интеграции LBS-сервисов с приложением зависит от решения разработчика, и для этого существуют различные варианты.

Поскольку приложение «Been There, Done That!» - это игра, вы хотите использовать в нем некоторые из наиболее распространенных функций, предоставляемых LBS-сервисами. Для этого вы добавите на экран с настройками Функционал, связанный с указанием избранного места пользователя. В ходе работы вы познакомитесь с другими возможностями LBS-сервисов, которые могут быть использованы для построения более мощных приложений.

На экране с настройками пользователь может выбрать и сохранить и качестве своего избранного места либо последнее известное местоположение, сохраненное в памяти телефона, либо ввести ключевые слона, описывающие что место, например его адрес, город или название достопримечательности (например, New York City, Iceland,

Yellowstone National Park, 90210). После этого приложение обращается к любому доступному поставщику услуг, основанных на геокодировании, чтобы преобразовать указанное местоположение в соответствующие GPS-координаты.

Чтобы реализовать подобный функционал на экране с настройками приложения «Been There, Done That!», вам потребуется немного изменить дизайн экрана. На рис. 14.1 изображен эскиз, который показывает, каким образом будет модифицирован экран с настройками.

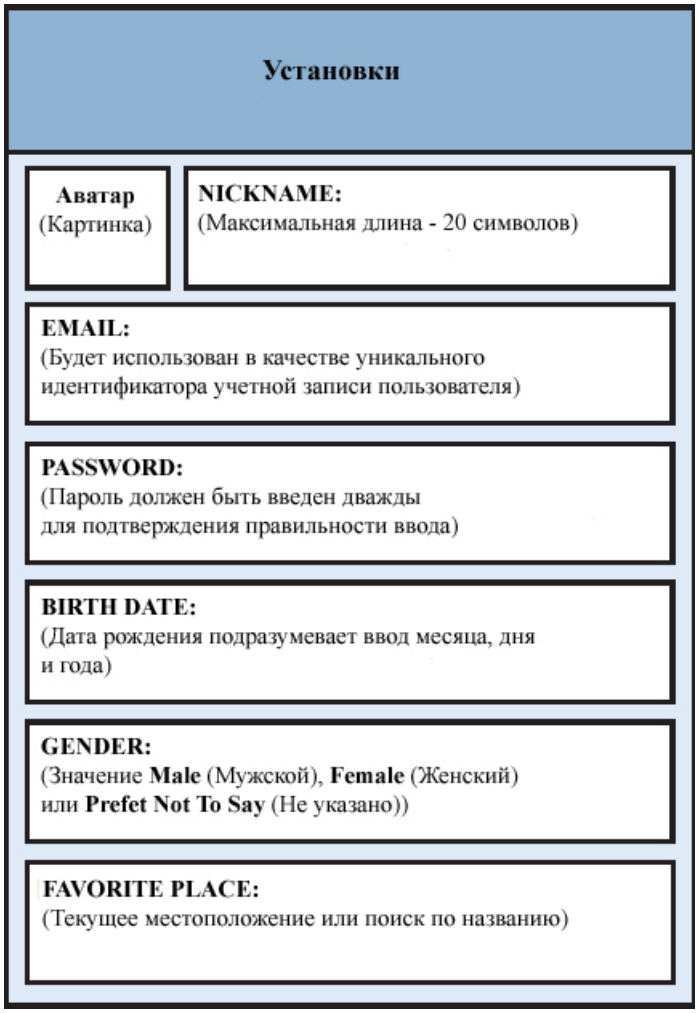

Рис. 14.1. Эскиз дизайна для функционала, связанного с указанием избранного места пользователя

Определение изменений макета для реализации геолокационного функционала Favorite Place

Напомним, что все поля, отображаемые на экране с настройками, размешаются внутри элемента `ScrollView`. Благодаря лому добавить новый параметр в нижнюю часть экрана не составит никакого труда. Функциональность, связанная с указанием избранного места пользователя, во многом будет работать так, как работают параметры **Date of Birth** (Дата рождения) и **Password** (Пароль).

Чтобы внести необходимые изменения в файл макета **/res/layout/settings.xml** для добавления параметра **Favorite Place** (Избранное место), вам потребуется создать новую область на экране с настройками, расположив ее под элементом `Spinner` параметра **Gender** (Пол).

Сначала вы добавите элемент `TextView`, который будет использоваться для отображения названия нового параметра. Затем — внутренний элемент- контейнер `LinearLayout` с элементом `Button`, используемым для отображения диалогового окна, и элементом управления `TextView`, который будет применяться для отображения выбранного значения параметра **Favorite Place** (Избранное место).

На рис. 14.2 представлены необходимые изменения макета.

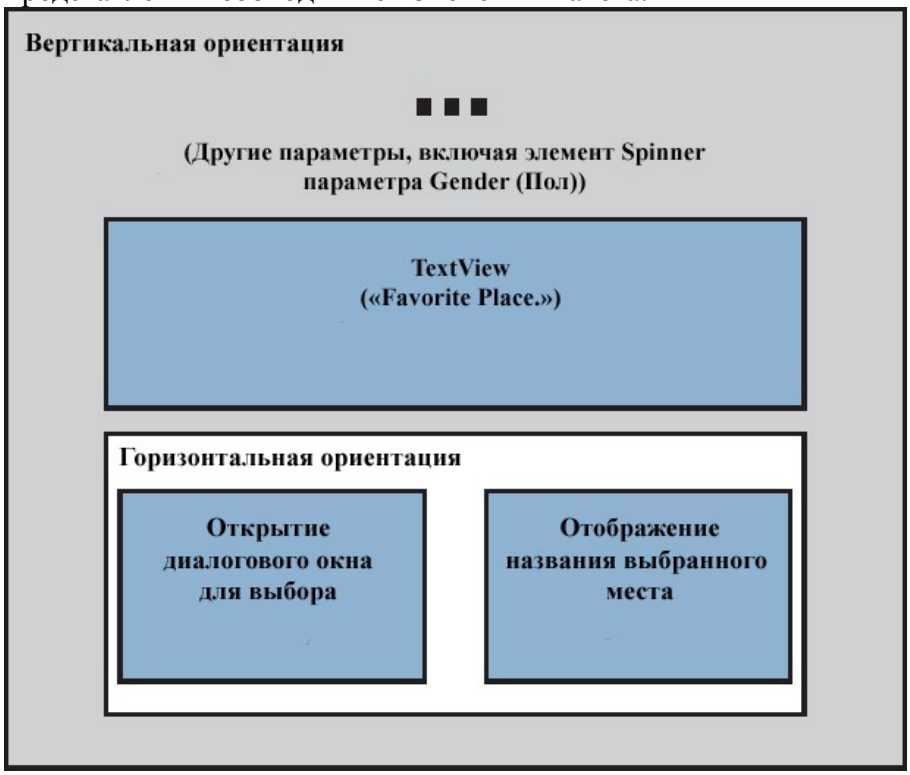

Рис. 14.2. Измененный макет экрана с настройками для реализации функциональности, связанной с указанием избранного места пользователя

Разработка дизайна диалогового окна для выбора избранного места пользователя

Как и раньше, вы построите пользовательское диалоговое окно на базе класса `AlertDialog`.

Значение параметра **Favorite Place** (Избранное место) в файле с настройками приложения будет храниться в виде трех частей:

- название местоположения (значение типа String);
- широта местоположения (значение типа float);

- долгота местоположения (значение типа float).

КСТАТИ

Технически можно получить и сохранить значение высоты над уровнем моря для выбранного местоположения, однако в большинстве современных приложений для работы с картами применяется вид с высоты птичьего полета в двухмерном пространстве.

Чтобы максимально упростить работу с этим диалоговым окном, вы можете предоставить пользователю две возможности; использовать последнее известное местоположение (при условии, что GPS-провайдер предоставляет необходимую информацию) или ввести строку в элемент `EditText`, которая будет преобразована в GPS-координаты при помощи инструментария Android SDK.

Рис. 14.3. Эскиз дизайна диалогового окна для выбора избранного места пользователя

Получив информацию о широте и долготе выбранного местоположения, вы можете включить возможность просмотра этого местоположения в приложении для работы с картами (если таковое имеется). На рис. 14.3 изображен эскиз дизайна диалогового окна для выбора избранного места пользователя.

Реализация макета диалогового окна для выбора избранного места пользователя

Вам нужно создать новый файл макета, в котором будет сохранен макет диалогового окна для выбора избранного места пользователя. Для этого вы добавите в проект новый ресурс макета с именем **/res/layout/fav_place_dialog.xml**.

Здесь нет ничего сложного. Все элементы диалогового окна размещаются внутри контейнера `LinearLayout` с вертикальной ориентацией. Сначала идет элемент

`TextView`, используемый для отображения текста с приглашением выбрать предпочтительное местоположение. Затем нужно отобразить элемент `EditText`, который будет использоваться для ввода названия места, а рядом с ним расположить элемент `Button`, позволяющий запустить приложение для работы с картами. Вы можете легко разместить элементы и `EditText Button` друг за другом, используя элемент-контейнер `RelativeLayout`. Наконец, нужно добавить два элемента `TextView`: один будет использоваться для отображения названия местоположения с указанными GPS-координатами, а другой— для отображения самих GPS-координат (в приложении «Been There. Done That!» эти координаты можно только просматривать, но не изменять).

На рис. 14.4 изображен дизайн-макет диалогового окна для выбора значения параметра **Favorite Place** (Избранное место).

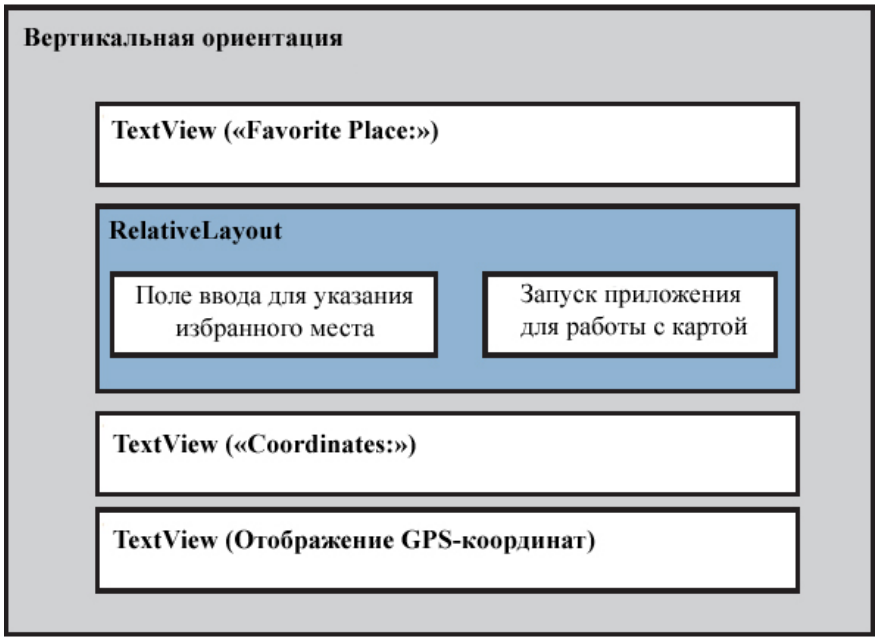

Рис. 14.4. Дизайн-макет диалогового окна
для выбора значения параметра **Favorite Place** (Избранное место)

РЕАЛИЗАЦИЯ ИНФРАСТРУКТУРЫ ДЛЯ ФУНКЦИОНАЛЬНОСТИ, СВЯЗАННОЙ С ВЫБОРОМ ИЗБРАННОГО МЕСТА ПОЛЬЗОВАТЕЛЯ

Перед тем, как приступить к добавлению поддержки LBS-сервисов и приложение «Been There. Done That!», нужно разработать инфраструктуру для функциональности, связанной с выбором избранного места. Здесь вам пригодятся навыки, полученные в предыдущих часах. Чтобы реализовать эту функциональность, выполните следующие шаги:

1. Добавьте новые ресурсы типа String, Dimension, Color и Drawable, которые потребуются для реализации макета.

2. Обновите файл макета **/res/layout/settings.xml**, чтобы добавить новую область в нижнюю часть экрана с настройками для открытия диалогового окна, как показано на рис. 14.5.

3. Добавьте в проект файл макета **/res/layout/fav_place_dialog.xml** и включите в него элементы `TextView`, `EditText` и `Button`, необходимые для диалогового окна (рис. 14.6).

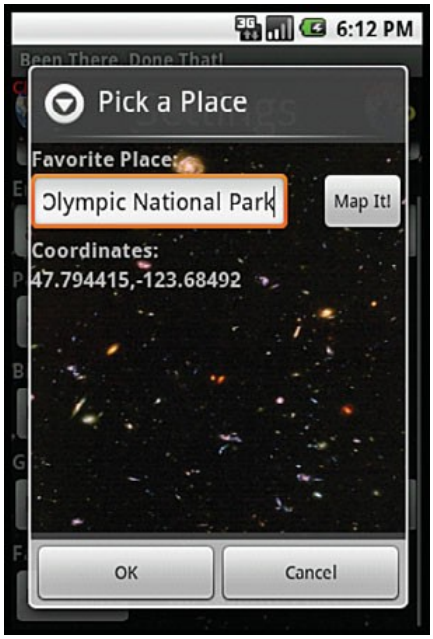

Рис. 14.5. Экран с настройками, содержащий параметр **Favorite Place** (Избранное место)

Рис. 14.6. Диалоговое окно для выбора избранного места пользователя

4. Определите три новых значения типа `String` в классе `QuizActivity` для дополнительных настроек приложения. Они будут использоваться в экземпляре класса `SharedPreferences` приложения для хранения названия избранного места пользователя (значение типа `String`), а также его географической широты (значение типа `float`) и долготы (значение типа `float`).

5. Обновите класс `QuizSettingsActivity`, включив в него новое диалоговое окно. Сначала определите в этом классе идентификатор диалогового окна (например, `PLACE_DIALOG_ID`). Затем обновите метолы `onCreateDialog()` и `onPrepareDialog()` класса, чтобы создать, инициализировать и управлять новым диалоговым окном для выбора избранного места пользователя.

Поскольку каждый из этих шагов был разобран выше, нет необходимости снова подробно рассматривать их. Тем не менее вот несколько подсказок, которые помогут реализовать необходимую функциональность:

- Добавьте вспомогательный метод `initFavoritePlacePicker()` для отображения названия избранного места пользователя (если оно было указано ранее) и обработки событий нажатий на элемент `Button`, чтобы открывать диалоговое окно

для выбора избранного места пользователя. Этот метод во многом должен быть похож на метолы `initPasswordChooser()` и `initDatePicker()`.

- Создайте новое диалоговое окно для выбора избранного места пользователя, во многом похожее на диалоговое окно для ввода пароля, которое вы реализовали ранее. Одно из ключевых отличий состоит в том, что в новом диалоговом окне присутствует элемент `Button`, используемый для запуска приложения для работы с картами.

- Шаг за шагом реализуйте диалоговое окно **Dialog** для выбора избранного места пользователя. Сначала реализуйте функциональность, позволяющую сохранять введенный в диалоговом окне **Dialog** текст в качестве названия избранного места пользователя. Затем добавьте сохранение некоторой тестовой информации о широте и долготе вместе с названием местоположения. Когда все это будет работать, добавьте реализацию класса `View.OnClickListener` для элемента управления Button, используемого для открытия приложения для работы с картами, и в соответствующем методе этого класса отображайте всплывающее уведомление с некоторым текстом, например, «Map Button Clicked».

КСТАТИ

Реализацию этих советов вы найдете на диске, прилагаемом к книге.

Реализовав инфраструктуру для функциональности, выбора избранного места пользователя, вы можете заняться более интересными вещами, например вычислением последнего известного системе местоположения и отображения на карте по GPS-координатам.

ИСПОЛЬЗОВАНИЕ ГЕОЛОКАЦИОННЫХ СЕРВИСОВ

Разработчики, реализующие поддержку LBS-сервисов, должны принимать во внимание несколько моментов. Первый и самый важный момент: местоположение пользователя — это конфиденциальная информация, для использования которой необходимо его согласие. Второй момент: LBS-сервисы на мобильном телефоне потребляют большое количество энергии и передают большой объем данных по сети.

Система Android позволяет частично решить эти проблемы при помощи разрешений. Вот несколько рекомендации по использованию LBS- и подобных сервисов:

- Активируйте LBS-функции только тогда, когда они действительно необходимы, и отключайте их сразу же, когда необходимость в них отпадает.

- Поставьте пользователя в известность о том, что вы собираете и используете конфиденциальные данные. Многие пользователи считают свое текущее или предыдущие местоположения конфиденциальной информацией.

- Предоставьте пользователю возможность настраивать и отключать функции, которые могут отрицательно повлиять на впечатление от использования вашего приложения. Например, разработайте режим «роуминга» для вашего приложения, чтобы позволить пользователю работать с вашим приложением, не неся серьезные затраты.

- Обрабатывайте такие события системы, как, например, предупреждения о разряде батареи, и соответствующим образом изменяйте работу вашего приложения.

- Подумайте над добавлением раздела, посвященного конфиденциальности, в соглашение об использовании вашего приложения, чтобы объяснить, как будут (или не будут) использоваться любые данные, указываемые пользователем, включая его имя и информацию о его местоположении.

ВНИМАНИЕ!
Не каждое устройство, работающее под управлением операционной системы Android, имеет аппаратное обеспечение для определения местоположения, поэтому вы не должны полагать, что абсолютно все устройства смогут предоставить информацию о местоположении пользователя.

Часть функциональности, связанной с LBS-сервисами, входит в состав Android SDK, а некоторые наиболее интересные функции — часть дополнения Google API. Это дополнение позволяет, например, встраивать функциональность сервиса Google Maps непосредственно в Android-приложения.

Разработчики, которые планируют использовать Google API, должны зарегистрироваться, чтобы получить специальный аккаунт разработчика Google, и использовать, специальный API ключ.

Тестирование функциональности, связанной с определением местоположения, на эмуляторе

Многие LBS-функции доступны разработчикам и без использования специальных аккаунтов разработчиков Google и API-ключей. Например, вак нужен специальный API-ключ дня использования сервиса Google Maps непосредственно из приложения, но, чтобы запустить объект типа `Intent` для просмотра местоположения с помощью приложений для работы с картами, специальные разрешения не требуются.

СОЗДАНИЕ AVD-УСТРОЙСТВЛ С ДОПОЛНЕНИЕМ GOOGLE API И ПРИЛОЖЕНИЯМИ

Возможно, вы заметили, что базовая конфигурация системы Android (целевая платформа, выбранная при создании AVD-устройства *зля* использования в качестве эмулятора) не имеет приложения для работы с картами. Чтобы использовать приложение Google Maps на эмуляторе, нужно создать AVD-устройство с дополнением Google API. Поскольку вы хотите добавить некоторую функциональность для работы с картами в приложение «Been There, Done That!», вам нужно создать новое AVD-устройство для этой целевой платформы.

НАСТРОЙКА МЕСТОПОЛОЖЕНИЯ НА ЭМУЛЯТОРЕ

К сожалению, эмулятор Android просто имитирует реальное устройство — естественно, поэтому он не может аппаратно определить свое текущее местоположение. Это придется

сделать вам самим. Простейший способ настроить ваш эмулятор — использовать перспективу **DDMS** в среде разработки Eclipse. Введите широту и долготу местоположения, которое вы хотите использовать на эмуляторе.

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Для получения GPS-координат вы можете использовать сервис Google Maps. Чтобы определить требуемые координаты, перейдите по адресу **maps.google.com** и найдите желаемое местоположение. Отцентруйте карту по этому местоположению, щелкнув правой кнопкой мыши по карте, и затем нажмите на кнопку для создания ссылки на карту (обычно она находится в правом верхнем углу экрана). Скопируйте полученный адрес URL в текстовый файл. Найдите последнюю переменную `ll` в запросе — это и есть широта и долгота. Например, значение переменной `ll`, представляющей берег Йеллоустонского озера в Йеллоустонском национальном парке, равно 51.845959, 104.908447. Значение 44.427896.-110.585632 переменной `ll` обозначает широту 44.427896 и долготу -110.585632. Вы можете еще раз проверить эти координаты, вставив их в строку поиска сервиса Google Maps и убедившись, что на карте отображается то самое место.

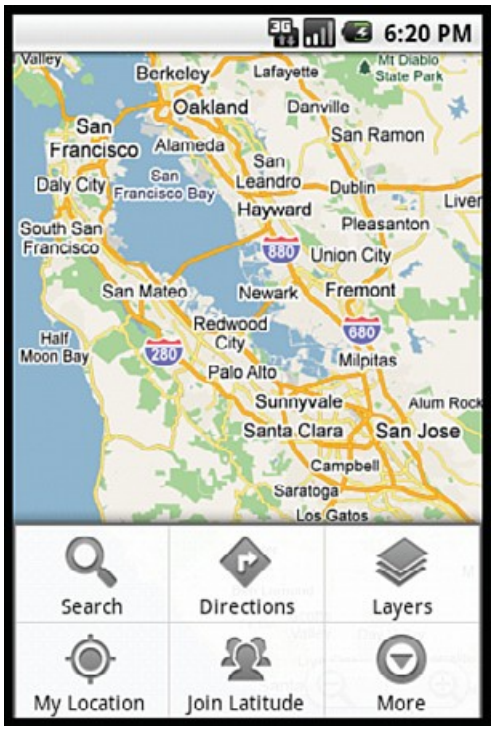

Рис. 14.7. Приложение Maps в эмуляторе Android

Чтобы настроить широту и долготу в эмуляторе, выполните следующие шаги:

1. Запустите эмулятор. Если в эмуляторе запущено какое-либо приложение, нажмите на кнопку **Home**.
2. Запустите приложение Maps.
3. Нажмите на кнопку **Menu.**
4. Выберите команду меню **My Location** (Мое местоположение) (рис. 14.7).
5. Откройте среду разработки Eclipse и выберите перспективу DDMS.
6. Выберите экземпляр эмулятора, для которого вы хотите указать координаты местоположения.

7. Прокрутите содержимое панели **Emulator Control** (Управление эмулятором) вниз, чтобы увидеть группу элементов управления **Location Controls** (Элементы управления местоположением).
8. Введите долготу и широту выбранного вами местоположения. Попробуйте указать координаты для Йеллоустонского национального парка: широта 44.427896 и долгота —110.585632 (рис. 14.8).
9. Нажмите на кнопку **Send** (Отправить).

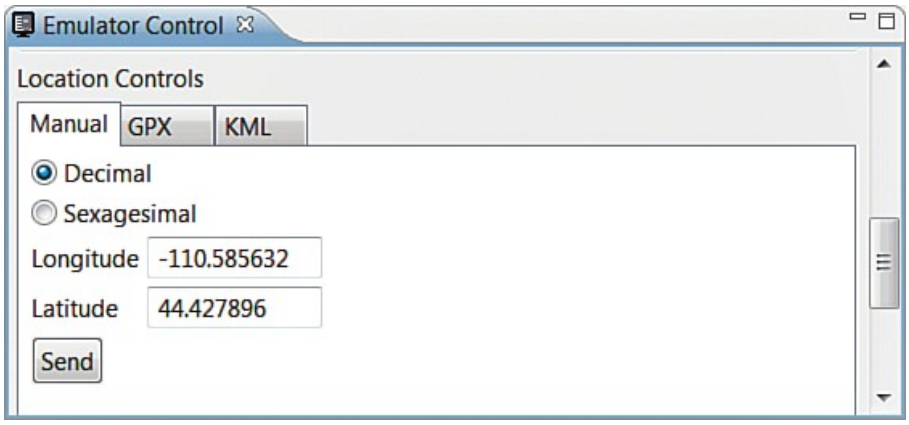

Рис. 14.8. Указание местоположения Йеллоустонского национального парка на эмуляторе при помощи перспективы DDMS

Открыв окно эмулятора, вы увидите, что на карте Google Maps теперь отображается указанное местоположение. На экране в качестве вашего местоположения должен отображаться Йеллоустонский национальный парк, как показано на рис. 14.9.

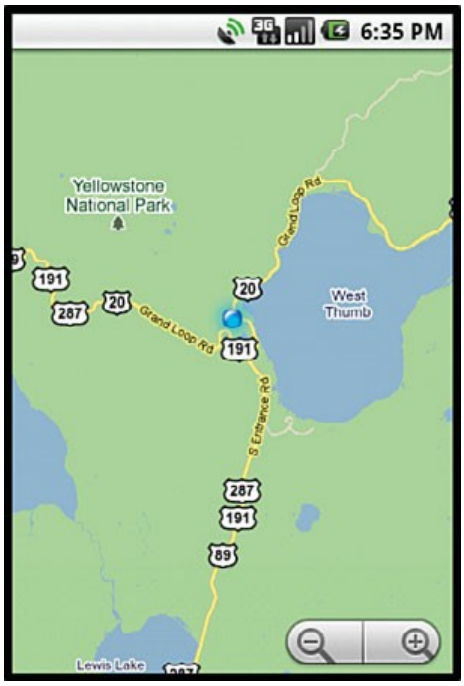

Рис. 14.9. Отображение в качестве местоположения Йеллоустонского национального парка

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Чтобы установить определение местоположения на эмуляторе, вы можете также использовать командную консоль эмулятора и выполнить команду `geo fix`.

Обращение к геолокационным сервисам

Для обращения к LBS-сервису на устройстве Android необходимы соответствующие разрешении. Геолокапионные сервисы не могут быть использованы Android-приложением до тех пор, пока для этого приложения не будут указаны соответствующие настройки `<uses-permission>` в файле манифеста Android.

Наиболее распространенные разрешения, которые работают с LBS-сервисами — android.permission.ACCESS_FINE_LOCATION и android.permission. ACCESS_COARSE_LOCATION. Для использования GPS-провайдера требуется разрешение android.permission.ACCESS_FINE_LOCATION.

Указав подходящее разрешение, вы можете получить экземпляр класса `LocationManager`, используя метод `getSystemService()`, как показано в следующем коде:

```
LocationManager locMgr =
        (LocationManager) getSystemService(LOCATION_SERVICE);
```

Класс `LocationManager` позволяет обращаться к функциям LBS-сервиса, доступного на устройстве.

РАБОТА С ПРОВАЙДЕРАМИ

На устройстве может существовать неограниченное количество LBS-провайдеров. Чтобы получить список всех доступных, вызовите метод `getProviders()` класса `LocationManager`. Вы можете выбрать только активных провайдеров ми указать критерий отбора провайдеров по определенному признаку (например, по точности определения координат). Вы также можете использовать метод `getBestProvider()`, чтобы получить провайдера, максимально соответствующего заданному набору критериев.

Любой из этих способов получения провайдеров позволяет получить список названий LBS-провайдеров. Наилучший LBS-провайдер для заданного набора критериев может быть возвращен по имени при помощи метода `getProvider()`. Вы можете использовать класс `LocationProvider`, чтобы изучить свойства заданного провайдера и определить возможности, которыми он обладает, например, предоставляет ли он информацию о высоте, азимуте и скорости, и должен ли пользователь платить деньги за его использование.

ПОЛУЧЕНИЕ ПОСЛЕДНЕГО ИЗВЕСТНОГО МЕСТОПОЛОЖЕНИЯ

Вы можете получить последнее известное местоположение устройства (вычисленного конкретным провайдером) при помощи метода `getLastKnownLocation()` класса `LocationManager`. Местоположение может быть устаревшим, но зачастую оно – хорошая отправная точка и данные об этом местоположении возвращаются очень быстро, в то время как попытка определить текущее местоположение на основании данных со спутников может занять некоторое время.

Чтобы получить последнее известное местоположение, нам не нужно спускать провайдер; вам достаточно запросить последний известный результат. Метод `getLastKnownLocation()` возвращает объект типа `Location`:

```
Location recentLoc =
    locMgr.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```

Объект типа Location может содержать ряд интересных данных о местоположении устройства. Доступная информация зависит от возможностей LBS-провайдера. Например, большинство провайдеров возвращают широту и долготу, однако не все провайдеры могут определить высоту над уровнем моря. Для получения координат из объекта типа `Location` используются методы `getLatitude()` и `getLongitude()`.

Получение обновлений информации о местоположении

Когда необходимо получать более актуальную информацию о текущем местоположении устройства и необходимо знать, когда происходит изменение местоположения, вы можете зарегистрироваться для получения периодических событий изменения местоположения, используя метод `requestLocationUpdates()` класса `LocationManager`. Этот метод позволяет деятельности прослушивать события, генерируемые конкретным провайдером (например, провайдером, который максимально соответствует вашим критериям). Изменять частоту получения уведомлений можно путем настройки минимального времени (в миллисекундах) и минимальной дистанции (в метрах) между обновлениями.

Чтобы получать уведомления об изменении местоположения, деятельность, заинтересованная в получении этих уведомлений, должна реализовать интерфейс LocationListener (android.location.LocationListener). Этот интерфейс имеет ряд полезных методов обработки событий, которые позволяют деятельности реагировать на ситуации включения и отключения провайдера, изменения его статуса и изменения местоположения.

ВНИМАНИЕ!

Для получения информации о текущем местоположении может потребоваться некоторое время. Поэтому желательно вынести большинство вызовов LBS-функций из основного потока, управляющего пользовательским интерфейсом, в отдельный рабочий поток (или написать для вашего приложения вспомогательную службу, работающую в фоновом режиме). Работа с потоками будет рассмотрена далее в этой книге.

ИСПОЛЬЗОВАНИЕ СЕРВИСОВ, ОСНОВАННЫХ НА ГЕОКОДИРОВАНИИ

Геокодирование - это процесс перевода описания местоположения (адреса) в GPS-координаты (широту, долготу и в некоторых случаях высоту над уровнем моря). Геокодирование позволяет нам укачать название места или достопримечательности в сервисе Google Maps (**maps.google.com**), например Eiffel Tower, и получить соответствующую точку на карте. Многие сервисы геокодиронания также предоставляют возможности обратного геокодирования, благодаря чему вы можете преобразовать координаты местоположения в адрес (обычно неполный).

Очевидно, что для выполнения задач геокодирования требуется специальное серверное приложение и устройство должно быть подключено к сети, чтобы взаимодействовать с ним. Различные сервисы, основанные на геокодировании, поддерживают различные типы описаний, ниже перечислены наиболее распространенные:

- названия городов, штатов и стран;
- разнообразные виды почтовых адресов (полные и неполные);
- почтовые индексы;
- коды аэропортов (например, LAX, LHR, JFK, DME);
- известные достопримечательности.

Безусловно, большинство сервисов, основанных на геокодировании, также позволяют указывать и обычные координаты (широту и долготу). Эти сервисы зачастую локализованы.

Адреса, полученные в результате геокодирования, часто оказываются неоднозначными, поэтому сервис может вернуть несколько записей. Например, если вы попытаетесь определить координаты адреса Springfield, вероятнее всего, вы получите довольно много результатов, поскольку города с названием Спрингфилд существуют в 35 штатах США, а еще больше городов с таким названием за пределами Соединенных Штатов. Также вы можете получить результаты для адресов East Springfield или Springfield by the Sea, например. Чтобы получить наилучшие результаты, указывайте как можно больше специфической информации (например, почтовый индекс города Спрингфилд вместо его названия).

ВНИМАНИЕ!
Как и в случае с любыми другими сетевыми операциями, сервисы, основанные на геокодировании, функционируют с использованием блокирующих операций. Это значит, что вы захотите вынести все обращения к сервисам, основанным на геокодировании, из основного потока, управляющего пользовательским интерфейсом, в сдельный поток.

Использование сервисов, основанных на геокодироваии, в Android-приложении

Android SDK содержит класс `Geocoder` (android.1ocation.Geocoder), позвоваляющий упростить взаимодействие с сервисами мобильного телефона, основанными на геокодировании и обратном геокодировании (если таковые присутствуют). В создании экземпляра класса `Geocoder` нет ничего сложного:

```
Geocoder coder = new Geocoder(getApplicationContext());
```

Имея экземпляр класса `Geocoder`, вы можете приступать к использованию любых доступных на устройстве сервисов, основанных на геокодировании и обратном геокодировании.

ГЕОКОДИРОВАНИЕ: ПРЕОБРАЗОВАНИЕ АДРЕСОВ В КООРДИНАТЫ

Вы можете использовать метод `getFromLocationName()` класса `Geocoder`, чтобы преобразовать адрес местоположения в его координаты. Этот метод принимает два параметра: строку, содержащую информацию о местоположении, и желаемое количество возвращаемых результатов. Например, следующий код выполняет поиск адреса Springfield с максимальным количеством возвращаемых результатов, равным трем:

```
String strLocation = "Springfield";
List<Address> geocodeResults =
    coder.getFromLocationName(strLocation, 1);
```

Для обработки результатов, возвращенных экземпляром класса `Geocoder`, можно использовать итератор:

```
Iterator<Address> locations = geocodeResults.iterator();
while (locations.hasNext()) {
   Address loc = locations.next();
   double lat = loc.getLatitude();
   double lon = loc.getLongitude();
   // TODO: Выполняем некие действия с полученными координатами
}
```

Каждый возвращенный объект типа `Address` содержит информацию о местоположении. Вы можете использовать методы `getLatitude()` и `getLongitude()` класса `Address` для получения координат местоположения.

Вы можете также использовать метод `getFromLocationName()`, чтобы ограничить количество возвращаемых адресов определенным значением.

ОБРАТНОЕ ГЕОКОДИРОВАНИЕ: ПРЕОБРАЗОВАНИЕ КООРДИНАТ В АДРЕСА

Вы можете использовать метод `get.FromLocation()` класса `Geocoder` для преобразования координат широты и долготы в информацию об адресе. Как и в предыдущем случае, в метод передаются координаты местоположения и значение, определяющее максимальное количество возвращаемых результатов.

РАБОТА С КАРТАМИ

Большинство функций для работы с картами на платформе Android реализуется специальным дополнением Google API. Например, вы можете использовать элемент `MapView` в ваших файлах макетов для тесной интеграции функций сервиса Google Maps в приложения. Вы можете интегрировать существующие на мобильном телефоне приложения для работы с картами в ваше приложение с использованием механизма интентов.

Запуск приложения для работы с картами при помощи интента

Приложения, работающие с местоположениями, например приложение Maps, обрабатывают интент `ACTION_VIEW` вместе с адресом URI, содержащим географические координаты. Этот адрес URI имеет специальный формат.

Когда вы определили координаты широты и долготы местоположения, вы можете запустить приложение Maps (или любое другое приложение, позволяющее обрабатывать этот тип данных), используя следующий формат строки адреса URI:

```
geo:широта, долгота
```

Вот пример кода, формирующего данную строку:

```
String geoURI = String.format("geo:%f,%f?z=10", lat, lon);
```

Этот специальный адрес URI также может содержать уровень масштабирования, выражаемый значением в диапазоне от 1 до 23, при этом уровень масштабирования, равный значению 1, позволяет отобразить всю планету, а 23-й уровень применяется для максимального приближения (обычно этот уровень масштабирования значительно превышает существующее разрешение карт). Чтобы вместе с интентом передать уровень масштабирования, используйте для адреса UR1 строку следующего формата:

```
geo:широта, долгота?z=уровень
```

Вот пример кода, позволяющего получить строку данного формата:

```
String geoURI = String.format("geo:%f,%f?z=10", lat, lon);
```

Подготовив правильно отформатированную строку URI, вы можете использовать метод `parse()`, чтобы создать объект типа `Uri` и использовать этот объект вместе с интентом ACTION_VIEW, как показано в следующем коде:

```
Uri geo = Uri.parse(geoURI);
Intent geoMap = new Intent(Intent.ACTION_VIEW, geo);
startActivity(geoMap);
```

Если на устройстве установлены приложения, которые позволяют обрабатывать адреса URI в geo-формате, в качестве активной деятельности будет запушено соответствующее приложение (например, приложение Maps) и отображено указанное местоположение. Когда пользователь изучит карту, он сможет вернуться к вызывающему приложению, нажав кнопку **Back** на мобильном телефоне.

Используя класс `Geocoder` и соответствующий интент для запуска приложения Maps, вы можете завершить реализацию диалогового окна для выбора избранного места пользователя, включая элемент Button с надписью **Map it!** (Показать на карте!), как показано на рис. 14.10.

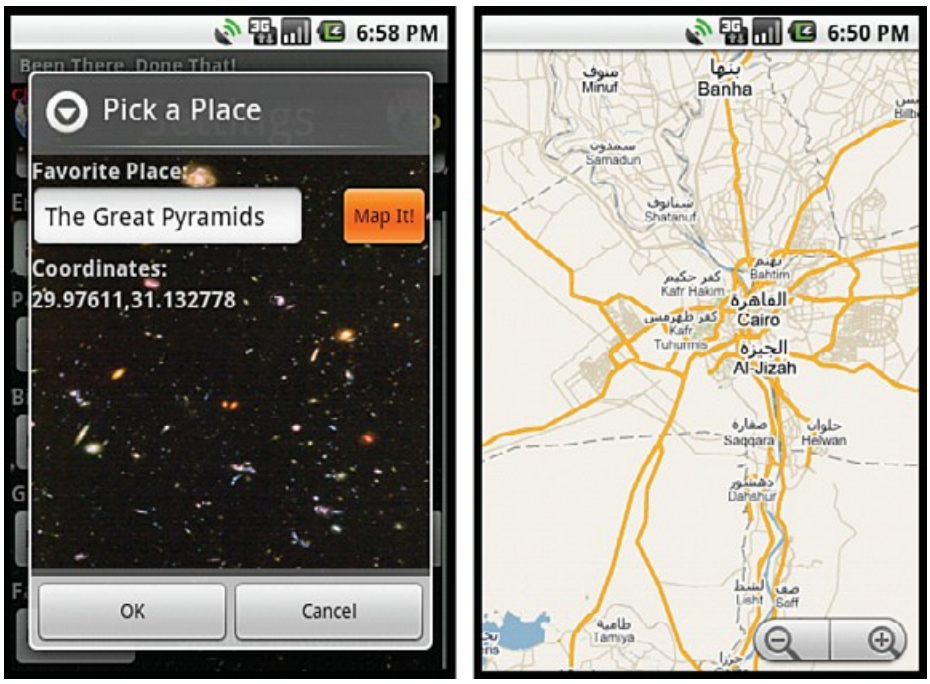

Рис. 14.10. Диалоговое окно для выбора избранного места пользователя с запущенным приложением Maps

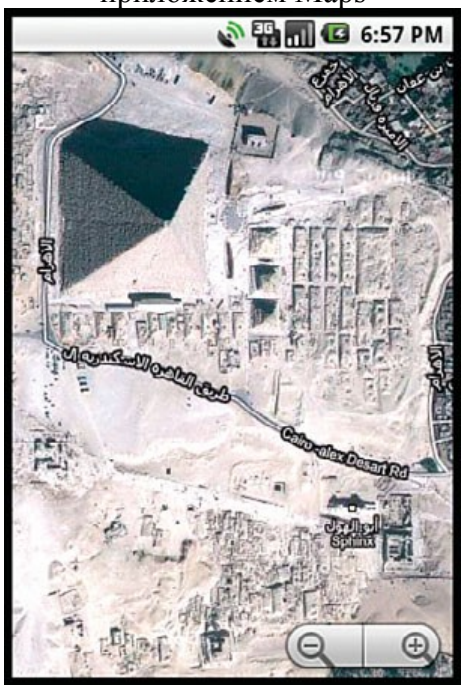

Рис. 14.11. Использование приложения Maps для увеличения масштаба отображения в режиме просмотра снимков со спутника

Если в приложении для работы с картами нажать кнопку **Menu**, вы сможете изменить режим отображения карты, переключившись на режим просмотра снимков со спутника, и увеличить масштаб отображения карты, чтобы увидеть великие пирамиды, а также туристические автобусы и Сфинкса, как показано на рис. 14.11.

Работа с сервисами и приложениями сторонних разработчиков

Поставляемая вместе с инструментарием Android SDK LBS-функциональность доступна в пакете `android.location`. Базовая LBS-функциональность, например получение местоположения на основе спутниковой триангуляции, встроена в инструментарий Android SDK. Тем не менее многие наиболее интересные и мощные функции по работе с картами и LBS-сервисами на телефонах Android «строены не и базовый инструментарии Android SDK, а входят в дополнение Google API, поставляемое вместе с Android SDK.

РАБОТА С ДОПОЛНЕНИЕМ GOOGLE API И РАСШИРЕННОЙ КАРТОГРАФИЧЕСКОЙ ФУНКЦИОНАЛЬНОСТЬЮ

Картографическая функциональность может быть встроена в приложения при помощи дополнения Google API. Вот некоторые из функциональных возможностей, доступных в пакете `com.google.android.map`:

- элемент управления MapView, применяемый для отображения интерактивной карты непосредственно в макете экрана приложения;
- класс MapActivity, упрощающий работу с элементами MapView на экране;
- класс GeoPoint, инкапсулирующий информацию о местоположении:
- классы, поддерживающие возможность наложения информации на карту (рисование поверх карты);
- классы для работы с проекциями местоположений и выполнения других распространенных геолокационных задач.

Для использования дополнения Google API и некоторых функций вы должны создать специальный аккаунт, принять лицензионное соглашение и получить API-ключ. Рассмотрение этой потрясающей и необычно мошной функциональности, к сожалению, выходит за рамки данной книги. Освоив основы реализации поддержки LBS-сервисов на платформе Android, вы можете обратиться к более детальному руководству по разработке на платформе Android, например к нашей книге Android Wireless Application Development (издательство Addison-Wesley), которая содержит подробные примеры использования дополнения Google API. Дополнительную информацию по этим классам можно найти на веб-сайте с описанием дополнения Google API, доступному по адресу **code.google.com/android/add-ons/googleapis/reference/index.html**.

ИТОГИ

В этом часе вы реализовали новую функциональность, связанную с выбором избранного места пользователя, на экране с настройками приложения «Been There, Done That!». Вы узнали, как использовать встроенные геолокационные сервисы для получения текущего местоположения устройства, а также как преобразовывать адреса в географические координаты. Вы также узнали, как можно запускать приложение для работы с картами и отображать указанное местоположение. Наконец, вы познакомились с некоторыми расширенными возможностями использования Функциональности LBS-сервисов, предоставляемыми инструментарием Android SDK.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Я хочу использовать элемент MapView. Где я могу получить ключ для дополнении Google API?

Ответ: На веб-сайте, посвященном дополнению Google API. где описаны нее шаги для получения ключа: **http://code.google.com/android/add-ons/googleapis/mapkey.html**. Этот процесс подразумевает создание аккаунта Google (если у вас его еще нет).

Вопрос: Как мне спроектировать приложение, для которого требуется более интенсивная обработка информация о местоположении, например, выполнение обновлений при изменении местоположения?

Ответ: Существует ряд подходов. Новички могут использовать объект типа `LocationManager`, который позволяет регистрировать деятельность для обработки периодических обновлений информации о местоположении, включая возможность запускать некий интент при возникновении определенного события. Все задачи, связанные с использованием LBS-сервисов, должны быть вынесены из основного потока, управляющего пользовательским интерфейсом, поскольку для их выполнения требуется некоторое время; вместо этого вы можете использовать рабочий поток, класс `AsyncTask` или процесс, выполняющийся в фоновом режиме. Кроме того, прослушивайте только те события, связанные с определением местоположения, которые вам действительно необходимы, чтобы избежать возникновения проблем с производительностью на устройстве.

ПРАКТИКУМ
Контрольные вопросы

1. Какие моменты должны учитывать разработчики при работе с LBS-сервисами?

A. Конфиденциальность пользователя.
B. Затраты пользователя на оплату услуг мобильной связи.
C. Продолжительность жизни батареи устройства.
D. Точность и достоверность информации, предоставляемой LBS-сер- висами и сервисами, основанными на геокодировании.
E. Время получения информации о местоположении.
F. Все вышеперечисленное.

2. Верно ли это? Помимо элементов Button, доступных в диалоговом окне AlertDialog, в пользовательском макете могут использоваться другие элементы Button.

3. Какие сервисы входят в Android SDK?

A. Геолокационные сервисы.
B. Сервисы, основанные на геоколировании и обратном геокодировании.
C. Сервисы для работы с картами.

4. Верно ли это? Поскольку эмулятор не настоящее устройство, на нем нет возможности использовать LBS-сервисы.

Ответы

1. F. При разработке приложений, использующих функциональность LBS-сервисов, разработчики должны принимать во внимание все перечисленные моменты.

2. Верно. Допускается использовать элементы `Button`, являющиеся частью пользовательского макета. При создании диалогового окна вы должны определить подходящие обработчики событий нажатий `View.OnClickListener`. Обратите внимание, что эти обработчики немного отличаются от обработчиков событий нажатий `DialogInterface.OnClickListener`, необходимых для управления тремя основными кнопками в диалоговом окне **AlertDialog**.

3. A и B. Инструментарий Android SDK включает поддержку LBS-сервисов. а также сервисов, основанных на геокодировании и обратном геокодировании. Сервисы, предоставляемые конкретными устройствами, могут отличаться. Картографические сервисы, поставляемые в виде дополнения Google API, не входят в базовый инструментарий Android SDK.

4. Неверно. Эмулятор обеспечивает поддержку LBS-сервисов (для некоторых сервисов требуется дополнение Google API), а перспектива DDMS может быть использована для изменения местоположения устройства.

Упражнения

1. Измените диалоговое окно для выбора избранного места пользователя таким образом, чтобы пользователь мог вводить GPS-координаты широты и долготы.

2. Измените диалоговое окно для выбора избранного места пользователя таким образом, чтобы пользователь мог указывать уровень масштабирования для отображаемой карты.

3. Измените приложение «Been There, Done That!» таким образом, чтобы вместе со значениями широты и долготы в настройках приложения сохранялась информация о высоте над уровнем моря.

Час 15. ДОБАВЛЕНИЕ СЕТЕВОЙ ПОДДЕРЖКИ

Вопросы, рассматриваемые в этом часе:

- **проектирование приложений с сетевой поддержкой;**
- **асинхронное выполнение задач;**
- **работа с индикаторами хода выполнения процесса;**
- **загрузка данных с сервера приложения.**

В этом и следующем часе вы реализуете поддержку двухстороннего сетевого взаимодействия в приложении «Been There, Done That!». В этом часе вы сосредоточитесь на загрузке данных из Интернета. В частности, вы измените приложение таким образом, чтобы наборы вопросов викторины и актуальные данные о результатах загружались с удаленного сервера приложения.

ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЙ С СЕТЕВОЙ ПОДДЕРЖКОЙ

Несмотря на то, что мобильные устройства прошли долгий путь в плане повышения скорости обработки и доступа к данным, связь с серверами по-прежнему важна. Она требуется для обеспечения переноса данных между устройствами, для создания резервных копий, для доступа к базам данных, которые не могут храниться на мобильном устройстве. К счастью, мобильные устройства прошли долгий путь и в плане развития возможностей подключения к вычислительным сетям и Интернету. Многие Android-фоны могут подключаться к Интернету различными способами, включая сети 3G (и более современные версии) или подключения с использованием технологии Wi-Fi. Android-приложения могут использовать многие из наиболее популярных интернет-протоколов, включая протоколы HTTP, HTTPS, TCP/IP и сокеты прямого доступа.

До сих пор в приложении «Been There, Done That!» вы использовали только тестовые XML-данные. Теперь пришло время изменить программу таким образом, чтобы она подключалась к удаленному серверу для получения актуальных данных. Для этого вам нужно познакомиться с сетевыми возможностями, доступными на платформе Android, а также узнать, как можно выносить выполнение задач за пределы основного потока, управляющего пользовательским интерфейсом, и выполнять их асинхронно.

В приложении «Been There, Done That!» загрузка информации с сервера требуется двум классам:

`QuizScoresActivity` — этому классу требуется загрузка информации о результатах;
`QuizGameActivity` — этому классу требуется загрузка каждого набора с вопросами викторины.

Чтобы приложение «Been There, Done That!» могло обрабатывать актуальные данные, вам потребуется доступ к серверу приложения, а также необходимо добавить сетевую поддержку в клиентское Android-приложение.

КСТАТИ

Полная версия кода, реализующего сетевую поддержку в приложении и рассматриваемого в этом часе, доступна на прилагаемом к книге диске.

Работа с сервером приложения

Приложения с сетевой поддержкой чаше всего связаны с сервером. Сервер обеспечивает централизованное хранение информации (в базе данных) и большую вычислительную мощность. Использование сервера позволяет разработчику использовать клиент-серверную архитектуру, при которой ядро приложения исполняется на сервере, а у клиента только интерфейс для связи с ним. Таким образом мы можем легко реализовать приложение «Been There, Done That!» для устройств iPhone и BlackBerry, а также веб-версии этой программы. В этом случае информация о результатах игры и друзьях пользователей могла бы без проблем использоваться в разнообразных клиентских средах.

Существует множество вариантов разработки сервера приложения. В этом примере используется очень простой масштабируемый сервер на базе платформы Google App Engine (дополнительную информацию можно получить по адресу **code.google.com/appengine/)** с использованием языка Java и сервлетов. Платформа Google App Engine помещает всю информацию в хранилище, где структура документов не имеет жесткой регламентации, и поддерживающем механизм запросов и атомарные транзакции. И хотя рассмотрение деталей реализации сервера приложения выходит за рамки данной книги, возможно, вам будет полезно узнать общие характеристики этого сервера.

Представьте, что сервер приложения для данного примера — это черный ящик со следующими свойствами:

- Сервер приложения всегда находится в работоспособном состоянии и доступен пользователям.
- Удаленный доступ к серверу приложения организуется по протоколу HTTP.
- Сервер приложения хранит данные, например настройки игрока, результаты и вопросы викторины.
- Сервер приложения может обрабатывать запросы на получение некой информации, например лучших результатов игры или наборов вопросов.
- Сервер приложения использует технологию Java Server Pages (JSP) для обработки HTTP-запросов и возвращает соответствующие результаты в XML-формате, который умеет обрабатывать Android-приложение.

КСТАТИ

Код сервера приложения, используемого в этой книге, доступен в папке **/Книга/ Сервер** на прилагаемом к книге диске.

Вы могли бы создать сервер приложения с другими характеристиками. Например, сервер под управлением базы данных SQL-типа с использованием СУБД MySQL и языка PHP или ряда других технологий.

Информирование пользователя о сетевой активности

В этом часе мы в основном будем знакомиться с механизмом отправки запросов серверу приложения и получения XML-данных, возвращаемых в результате выполнения этого запроса. Реализация сетевой поддержки не требует внесения каких-либо изменений в пользовательский интерфейс или в макеты экранов приложения «Been There, Done That!». Тем не менее каждый раз, когда пользователь должен дожидаться окончания операции, выполнение которой занимает определенное время, — например, операция загрузки с сервера и разбора XML-данных — важно информировать пользователя о том, что в приложении происходят какие-то действия, используя визуальный механизм, например элемент `ProgressBar` (рис. 15.1). В противном случае пользователь может завершить приложение из-за отсутствия обратной реакции.

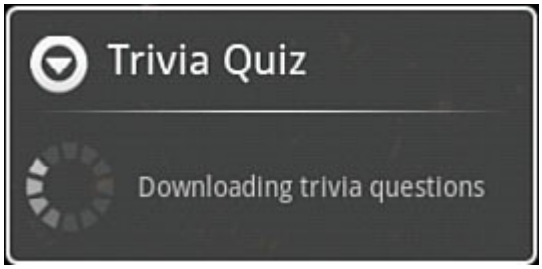

Рис. 15.1. Использование индикатора непрерывного хода выполнения процесса для информирования пользователя о продолжительной операции

РАЗРАБОТКА ПРИЛОЖЕНИЙ С СЕТЕВОЙ ПОДДЕРЖКОЙ

Программисты, реализующие сетевую поддержку в своих приложениях должны учитывать ряд моментов, в чем-то схожих с проблемами, возникающими при использовании геокодирования. Ключевые вопросы здесь – обеспечение конфиденциальности информации пользователя, снижение производительности устройства и расходы, связанные с передачей по сети нежелательных данных. Поскольку гарантировать постоянное подключение к сети невозможно (доступность сети, уровень сигнала и его качество), реализация сетевой поддержки в вашем приложении создает множество потенциальных источников для возникновения сбоя.

Система Android позволяет частично сгладить эти проблемы при помощи разрешений, однако основной труд по управлению сетевыми возможностями и производительностью устройства ложится на разработчика. Вот несколько советов тем, кто собирается использовать сетевую функциональность в своих приложениях:

- Используйте сетевые сервисы только по необходимости и выполняйте локальное кэширование данных всегда, когда это возможно.

- Поставьте пользователя в известность о том, что вы собираете и используете конфиденциальные данные.

- Предоставьте пользователю возможность настраивать и отключать функциональность, которая может вызвать негативное впечатление от использования вашего приложения. Например, реализуйте «режим полета», чтобы позволить пользователю наслаждаться вашим приложением без обращения к удаленному серверу.

- Корректно обрабатывайте такие ситуации, как, например, отсутствие покрытия сети. Ваше приложение будет более ценным для пользователя, если им можно будет пользоваться даже без подключения к Интернету.

- Подумайте над добавлением раздела, посвященного конфиденциальности, в соглашение об использовании приложения. Используйте эту возможность, чтобы предупредить пользователя о том, какие данные собирает приложение, каким образом они будут (или не будут) использоваться, и будут ли они храниться где-либо (например, на удаленном сервере приложения).

ВНИМАНИЕ!

На сегодняшний день большинство Android-устройств — телефоны и планшеты с возможностью подключения к Интернету, но не только. Нет никакой гарантии, что каждое устройство под управлением ОС Android имеет сетевую поддержку, однако можно с большой уверенностью утверждать, что в том или ином виде подключение к Интернету будет присутствовать. Учитывайте этот факт при принятии решения об использовании сетевых возможностей вашим приложением.

Включение возможности тестирования сетевой поддержки на эмуляторе

Для написания приложений с сетевой поддержкой не нужно вносить никаких изменений в эмулятор. Эмулятор будет использовать интернет-соединение, доступное на нашем компьютере, и имитировать реальное покрытие сети. Кроме того. эмулятор имеет ряд настроек, позволяющих имитировать задержки в сети и скорости передачи данных, благодаря чему вы сможете получить лучшее представление о реальной работе вашего приложения, когда оно будет использоваться конечным пользователем. Дополнительную информацию по настройкам, связанным с отладкой сетевой поддержки приложения, можно найти в документации по эмулятору Android.

ВНИМАНИЕ!

Поскольку эмулятор использует подключение к Интернету, доступное на вашем компьютере, вероятнее всего, оно будет иметь более высокую скорость, чем подключение к Интернету на реальном устройстве.

Тестирование приложений с сетевой поддержкой на реальных устройствах

Как всегда, лучший способ протестировать приложения с сетевой поддержкой — использовать реальное устройство Android. На устройстве Android доступен целый ряд настроек, связанных с сетевой функциональностью. Их изменение осуществляется при помощи приложения **Settings** (Настройки) на устройстве.

- **Airplane Mode** (Режим полета) — этот режим позволяет блокировать все сетевые соединения в соответствии с правилами большинства авиакомпаний.

- **Wi-Fi** — настройки Wi-Fi для тех случаев, когда доступны беспроводные сети.

- **Mobile Networks** {Мобильная сеть) — настройки для передачи данных в роуминге.

Вы можете увидеть полезную служебную информацию, выбрав приложение **Settings** (Настройки) в списке приложений, а затем команду меню **About phone** (О телефоне) (или **About device** (Об устройстве)), и затем команду **Status** (Состояние). Здесь вы сможете найти следующую служебную информацию:

- номер телефона (например, 888-555-1212);

- беспроводная сеть (например, Verizon, T-Mobile);

- тип сети (например, CDMA EVDO rev. A или EDGE);

- уровень сигнала (например, -81 dBm 0 asu);

- состояние подключения к сотовой сети (например, **In Service** (Обслуживается));

- состояние роуминга (например, **Roaming** (Роуминг включен) или **Mot roaming** (Роуминг отключен));

- состояние сотовой сети (например, **Connected** (Подключено));

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Чтобы заставить мобильное устройство потерять сигнал, вы можете поместить это устройство в жестяную коробку из-под печенья, холодильник, микроволновую печь или в любой другой экранированный объект. Это позволит протестировать ситуацию потери сигнала и сети. Только не оставляйте мобильное устройство на холоде в течение длительного времени, поскольку это приведет к быстром разряду батареи. И не включайте микроволновую печь, когда в ней находится телефон.

ОБРАЩЕНИЕ К СЕТЕВЫМ СЕРВИСАМ

Платформа Android имеет большое количество библиотек для работы с сетью. Те из вас, кто занимался реализацией сетевого взаимодействия на языке Java, должны быть знакомы с пакетом java.net. Кроме того, в Android SDK существует ряд полезных вспомогательных классов для сетевых операций и протоколов. Чтобы сделать передачу данных по сети более защищенной, разработчики могут использовать распространенные технологии, например протоколы SSL и HTTPS.

Для работы с сетью Android-приложение должно иметь соответствующие разрешения, которые устанавливаются в файле манифеста Android. Сетевые задачи сами по себе — блокирующие операции, а скорость передачи данных по мобильным сетям может быть очень медленной, поэтому просто необходимо, чтобы сетевые операции выполнялись асинхронно.

ВНИМАНИЕ!
Выполнение сетевых операций может занимать достаточно много времени. Поэтому вызовы всех методов, подразумевающих работу с сетью, должны выполняться асинхронно, отдельно от основного потока, управляющего пользовательским интерфейсом. Это можно сделать при помощи класса `Thread` платформы Java или при помощи класса `AsyncTask` платформы Android, который будет рассматриваться далее в этом часе.

Настройка разрешений для работы с сетью

Для работы с сетью на устройстве под управлением операционной системы Android приложение должно иметь соответствующие разрешения. Android-приложение может использовать большинство сетевых сервисов только в том случае, если в элементе `<uses-permission>` файла манифеста Android будут указаны необходимые значения.

Ниже перечислены два наиболее часто используемых значения разрешений для приложений, работающих с сетью:

- android.permission.INTERNET;
- android.permission.ACCESS_NETWORK_STATE.

Существует ряд других разрешений, имеющих отношение к работе с сетью, включая разрешения, которые позволяют обращаться и изменять состояние сети Wi-Fi и общее состояние сети. Возможно, вас также заинтересует разрешение android.permission.WAKE_LOCK, которое используется для предотвращения перехода устройства в спящий режим.

Проверка состояния сети

Инструментарий Android SDK предоставляет механизмы для сбора информации о текущем состоянии сети. Их полезно применять для определения доступности сетевого соединения перед использованием необходимого сетевого ресурса. Проверив доступность сети, вы сможете избежать многих источников ошибок и обеспечить более приятное взаимодействие с приложением для ваших конечных пользователей.

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О СОСТОЯНИИ СЕТИ ПРОГРАММНЫМ ПУТЕМ

Чтобы иметь возможность получать информацию о состоянии сети на мобильном устройстве, для приложений в файле манифеста Android должно быть указано разрешение

android.permission.ACCESS_NETWORK STATE. Чтобы иметь возможность изменять состояние сети на мобильном устройстве, приложение также должно иметь разрешение android.permission. CHANG E_NETWORK_STATE.

Разработчики могут использовать класс `ConnectivityManager(android.net .ConnectivityManager)` для получения информации о состоянии сети на устройстве программным путем. Получить экземпляр класса `ConnectivityManager` можно при помощи знакомого вам метода `getSystemService()` объекта типа `Context` приложения:

```
ConnectivityManager conMgr = (ConnectivityManager)
    getSystemService(Context.CONNECTIVITY_SERVICE);
```

Получив экземпляр класса `ConnectivityManager`, вы можете запросить информацию о мобильной (сотовой) сети при помощи метода `getNetworkInfo()`:

Класс `Networklnfo(android.net.Networklnfo)` предоставляет ряд методов для получения важной информации о состоянии сети, включая доступность сети, состояние подключения и состояние роуминга:

```
boolean isMobileAvail = netInfo.isAvailable();
boolean isMobileConn = netInfo.isConnected();
boolean isRoamingConn = netInfo.isRoaming();
```

Класс `NetworkInfo` также имеет множество других методов для получения подробной информации о состоянии сети. Дополнительную информацию по этим методам можно найти в документации.

ПРОВЕРКА ДОСТУПНОСТИ СЕРВЕРА ПРОГРАММНЫМ ПУТЕМ

Даже если сеть доступна и устройство подключено к этой сети, нет никакой гарантии, что доступен удаленный сервер, с которым будет взаимодействовать ваше приложение. Однако класс `ConnectivityManager` имеет удобный метод `requestRouteToHost()`, который позволяет убедиться в возможности передачи трафика на определенный IP-адрес с использованием указанного типа сети (например, сотовой сети или сети Wi-Fi).

Использование соединений по протоколу HTTP

Самый распространенный протокол передачи данных по сети - Hypertext Transfer Protocol (HTTP). Большинство портов для соединения по протоколу HTTP открыто и доступно для использования в мобильных сетях.

Чтобы быстро получить сетевой ресурс, достаточно загрузить содержимое этого ресурса в виде потока. Многие интерфейсы платформы Android, предназначенные для чтения данных, позволяют работать с потоками. Один из таких примеров — класс `XmlPullParser`. Метод `setInput()` класса `XmlPullParser` принимает параметр типа `InputStream`. Ранее вы получали этот поток из ресурсов. Теперь, однако, вы можете получить этот поток из сети, используя простой класс URL, как показано ниже:

```
URL xmlUrl = new URL(xmlSource);
XmlPullParser questionBatch =
    XmlPullParserFactory.newInstance().newPullParser();
questionBatch.setInput(xmlUrl.openStream(), null);
```

Остальная часть кода, связанная с разбором XML-данных, не меняется, поскольку XML-формат остался прежним, а используемый класс `XmlResourceParser` был унаследован от класса `XmlPullParser`. Поскольку разбор данных теперь может занимать более продолжительное время, вы вынесете эту функциональность за пределы основного потока далее в этом часе.

Теперь, когда вопросы и данные о результатах игры находятся на удаленном сервере, вы можете удалить из проекта тестовые XML-ресурсы и код, который их загружает.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Если вы хотите, чтобы ваше приложение получало и отображало веб-контент, вы можете использовать элемент WebView, в котором применяется движок WebKit для визуализации HTML-контента на экране. Элемент WebView может применяться для отображения как локального контента, так и контента с сайтов в Интернете.

ИНФОРМИРОВАНИЕ ПОЛЬЗОВАТЕЛЯ О СЕТЕВОЙ АКТИВНОСТИ ПРИ ПОМОЩИ ИНДИКАТОРОВ ХОДА ВЫПОЛНЕНИЯ ПРОЦЕССА

Приложения е сетевой поддержкой часто выполняют такие задачи, как уста- ноа1ение соединений с удаленными серверами, загрузка и разбор полученных данных. Их выполнение занимает определенное время, и пользователь должен понимать, что приложение работает в текущий момент. Отличный способ проинформировать пользователя — отобразить на экране индикатор хода выполнения процесса. Инструментарий Android SDK предоставляет два основных варианта представления для элемента управления `ProgressBar` — индикатор непрерывного хода выполнения процесса и индикатор хода выполнения процесса на основе значений.

Отображение индикатора непрерывного хода выполнения процесса

Простейшее представление элемента `ProgressBar` выглядит как анимированный круговой индикатор (рис. 15.2). По существу индикатор данного вида не отражает ход выполнения процесса, однако он информирует пользователя о некоторой активности, происходящей в приложении. Этот тип индикатора используется в тех случаях, когда длительность выполнения фонового процесса заранее не известна.

Рис. 15.2. Анимированный индикатор непрерывного хода выполнения процесса

Отображение индикатора хода выполнения процесса на основе значений

Когда вы хотите информировать пользователя о достижении определенных ключевых этапов в ходе выполнения процесса, вы можете использовать индикатор хода выполнения процесса на основе значений. Этот элемент пользовательского интерфейса представляет собой горизонтальную полосу, которая может постепенно заполняться другим цветом по мере выполнения процесса (рис. 15.3). Вместе с этим типом индикатора хода выполнения процесса применяется метод `setProgress()` элемента `ProgressBar`.

Рис. 15.3. Индикатор хода выполнения процесса на основе значений

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Как будет показано ниже, вы можете поместить индикаторы хода процесса в строку заголовка приложения, Это позволяет сэкономить пространство экрана.

Отображение диалоговых окон, демонстрирующих ход выполнения процесса

Вы можете продемонстрировать ход выполнения процесса в отдельном диалоговом окне, вместо добавления элемента `ProgressBar` в макет существующего экрана. Для этой цели вы можете использовать специальный класс `ProgressDialog`, унаследованный от класса **Dialog**. Например, вы можете использовать диалоговые окна **ProgressDialog** (рис. 15.4) в приложении «Been There, Done That!», чтобы проинформировать пользователя о том, что в текущий момент происходит загрузка или разбор данных, перед тем. как отобразить соответствующий экран приложения.

Рис. 15.4. Информирование пользователя о том, что в текущий момент происходит загрузка вопросов викторины

Ниже представлен код, который используется для создания и отображения диалогового окна ProgressDialog программным путем:

```
ProgressDialog pleaseWaitDialog = ProgressDialog.show(
    QuizGameActivity.this,
    "Trivia Quiz",
    "Downloading trivia questions…",
    true);
```

Вы можете использовать метод `dismiss()`, чтобы прекратить отображение объекта `pleaseWaitDialog`, когда фоновая обработка данных будет завершена:

```
pleaseWaitDialog.dismiss();
```

КСТАТИ

Отображение объекта `pleaseWaitDialog` на экране может быть отменено пользователем, если в метод `show()` будет передан пятый параметр, установленный в значение true. В предыдущем коде мы не позволяем пользователю закрывать диалоговое окно раньше времени, поскольку хотим, чтобы оно отображалось в течение всего процесса загрузки данных. В коде примера, демонстрирующем готовый функционал для данного часа, вы увидите, что пользователю предоставлена возможность принудительного закрытия диалогового окна.

Теперь вы знаете, как создавать индикаторы хода выполнения процесса и отображать их в диалоговых окнах **ProgressDialog**. Поскольку выполнение процесса, ход которого вы хотите представлять при помощи индикаторов, должно происходить асинхронно, пришло время познакомиться с фоновыми процессами.

АСИНХРОННОЕ ВЫПОЛНЕНИЕ ЗАДАЧ

Несмотря на стремительное развитие технологий, скорость интернет-соединений в мобильных беспроводных сетях по-прежнему остается относительно невысокой. Ваши Android-приложения должны быстро реагировать на действия пользователей, поэтому все сетевые операции следует выносить во вспомогательный «рабочий» поток. Платформа Android позволяет сделать это двумя способами.

- Класс `AsyncTask` — этот абстрактный класс может быть использован для эффективного вынесения фоновых операций за пределы потока, управляющего пользовательским интерфейсом. Данный класс — оптимальный вариант для выполнения операций, занимающих определенное время и могущих оказать влияние на производительность и время реакции вашего приложения на действия пользователя.

- Классы `Thread` и `Handler` — эти классы могут использоваться совместно для выполнения параллельных вычислений и взаимодействия с очередью сообщений поюка, управляющего пользовательским интерфейсом. Этот расширенный подход предоставляет большую гибкость в плане реализации, однако на разработчика ложится ответственность за правильное управление потоковыми операциями

Для приложения «Been There, Done That!» лучше всего подходит использование класса `AsyncTask`, поскольку эта реализация максимально проста и понятна.

Использование класса AsyncTask

В состав инструментария Android SDK входит класс `AsyncTask` (`ancroid.os.AsyncTask`), помогающий организовать выполнение фоновых операций, результаты которых со временем будут возвращены в поток, управляющий пользовательским интерфейсом.

Вместо использования обработчиков и создания потоков вы можете просто создать класс, производный от класса `AsyncTask`, и реализовать необходимые методы обработки событий.

- onPreExecute() — этот метод выполняется в потоке, упраатяюшем пользовательским интерфейсом, до того, как будет запушен фоновый процесс.

- `dolnBackground()` — этот метод выполняется в фоновом режиме, и именно в нем происходит вся основная работа.

- `publishProgress()` — этот метод, вызываемый из метода `dolnBackground()`, периодически информирует поток, управляющий пользовательским интерфейсом, о ходе выполнения фонового процесса. Этот метод отправляет информацию в основной процесс приложения.

- `onProgressUpdate()` — этот метод выполняется в потоке, управляющем пользовательским интерфейсом, каждый раз, когда в методе `dolnBackground()` происходит вызов метода `publishProgress()`. Этот метод получает информацию от фонового процесса.

- `onPostExecute()` — этот метод выполняется в потоке, управляющем пользовательским интерфейсом, сразу после завершения фоновой обработки.

После вызова метода `execute()` вся обработка, выполняемая классом `AsyncTask`, происходит в фоновом потоке, позволяя избежать блокировки потока, управляющего пользовательским интерфейсом.

Использование потоков и обработчиков

Если вы хотите самостоятельно управлять потоком, вы можете использовать класс `Thread` совместно с объектом типа `Handler`.

Представленный ранее способ реализации сетевых операций имеет существенный недостаток: поток заблокирован до тех пор, пока сетевая операция не завершена. Для небольших задач такой подход может оказаться приемлемым. Тем не менее, когда в процессе выполнения задач возникают задержки или требуется дополнительное время на обработку данных, например, для разбора XML-данных, вы должны вынести все эти продолжительные операции за пределы основного потока.

Следующий кол демонстрирует создание и запуск потока дли анонимного класса `Thread`, который устанавливает соединение с удаленным сервером, получает и разбирает XML-данные, и отправляет сообщения обратно в поток, управляющий пользовательским интерфейсом, используя объект типа `Handler` в основном потоке, чтобы изменить состояние элемента `TextView` с именем `parsingStatus`, представляющего статус процесса разбора данных:

```
import android.os.Handler;
Handler mHandler = new Handler();
// ...
new Thread() {
    public void run() {
        // Instantiate XML parser

        mHandler.post(new Runnable() {
           public void run() {
           parsingStatus.setText("Began Parsing...");
           }
        });
        // XML Parsing loop here
        // Update parsingStatus has needed…
        mHandler.post(new Runnable() {
           public void run() {
           parsingStatus.setText("Finished parsing...");
           }
        });
    }
}.start();
```

Класс `Thread` использует объект типа `Handler` с именем `mHandler` для отправки информации обратно в основной поток, управляющий пользовательским интерфейсом.

ЗАГРУЗКА И ОТОБРАЖЕНИЕ РЕЗУЛЬТАТОВ ИГРЫ

Теперь давайте разберем простой пример использования класса `AsyncTask` в классе `QuizSettingsActivity` для выполнения нагрузки и разбора информации о результатах игры в формате XML. Сначала создается производный класс с именем `ScoreDownloaderTask`, который расширяет класс `AsyncTask` внутри класса `QuizSettingsActivity`:

```
private class ScoreDownloaderTask extends AsyncTask<Object, String,
Boolean> {
    // TODO: Implement AsyncTask callback methods
    TableLayout table;
}
```

Поскольку вы будете заполнять данными элемент-контейнер `TableLayout` в процессе выполнения этой фоновой задачи, также имеет смысл добавить в класс `ScoreDownloaderTask` удобную переменную-член.

Запуск задачи, реализуемой классом ScoreDownloaderTask

Реализовав класс `ScoreDownloaderTask`, вы должны будете обновить метод `onCreate()` класса `QuizScoresActivity`, чтобы вызвать метод `execute()` класса `ScoreDownloaderTask` при первой загрузке соответствующего экрана. Метод `execute()` принимает два параметра: адрес URL сервера и таблицу, в которую будут помещаться результаты игры (элемент- контейнер `TableLayout`):

```
public static final String TRIVIA_SERVER_BASE =
    "http://tqs.mamlambo.com/";
public static final String TRIVIA_SERVER_SCORES =
    TRIVIA_SERVER_BASE + "scores.jsp";
// ...
allScoresDownloader =
    new ScoreDownloaderTask();
allScoresDownloader.execute(TRIVIA_SERVER_SCORES, allScoresTable);
SharedPreferences prefs =
    getSharedPreferences(GAME_PREFERENCES, Context.MODE_PRIVATE);
Integer playerId = prefs.getInt(GAME_PREFERENCES_PLAYER_ID, -1);
if (playerId != -1) {
    friendScoresDownloader = new ScoreDownloaderTask();
    friendScoresDownloader.execute(
        TRIVIA_SERVER_SCORES + "?playerId="
        + playerId, friendScoresTable);
}
```

Теперь, когда все готово к запуску задачи, реализуемой классом `ScoreDownloaderTask`, вы можете сконцентрироваться на надлежащей реализации методов обработки событий класса `AsyncTask`.

КСТАТИ

Не беспокойтесь по поводу значения переменной `playerId`. Мы рассмотрим соответствующую функциональность в следующем часе.

Запуск индикатора хода выполнения процесса

Теперь вы должны реализовать метод `onPreExecute()`, который выполняется в потоке, управляющем пользовательским интерфейсом, перед началом выполнения фонового процесса. Это отличное место для демонстрации добавления индикатора непрерывного хода выполнения процесса в строку заголовка приложения:

```
@Override
protected void onPreExecute() {
   mProgressCounter++;
   QuizScoresActivity.this.setProgressBarIndeterminateVisibility(true);
}
```

Для отображения результатов игры имеются две вкладки. Результаты для каждой вкладки загружаются по отдельности, и нужно сделать так, чтобы индикатор хода выполнения процесса отображался до тех пор, пока все данные не будут загружены. Таким образом, вы можете создать счетчик — `mProgressCounter` — для отслеживания каждой загрузки. В этом случае, даже если бы вы добавили еще одну вкладку, индикатор по-прежнему исчезал бы с экрана в нужный момент.

В метод `onCreate()` деятельности должна быть добавлена следующая строка, чтобы индикатор непрерывного хода выполнения процесса правильно отображался в строке заголовка приложения:

```
requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
```

Этот метод должен быть вызван перед вызовом метода `setContentView()`.

Выполнение фоновой обработки

Теперь настало время определить, какие операции должны выполнятся асинхронно. В данном примере к этим операциям относится разбор XML-данных. Вы переопределяете метод `doInBackground()`, в кокором будет происходить выполнение соответствующих функций. Методы, вызываемые внутри метода `doInBackground()`, не будут блокировать основной поток, управляющий пользовательским интерфейсом. Ниже представлен пример реализации метода `doInBackground()`, из которого для простоты был удален код обработки исключений:

```
@Override
protected Boolean doInBackground(Object... params) {
    boolean result = false;
    String pathToScores = (String) params[0];
    table = (TableLayout) params[1];
    XmlPullParser scores = null;
    URL xmlUrl = new URL(pathToScores);
    scores = XmlPullParserFactory.newInstance().newPullParser();
    scores.setInput(xmlUrl.openStream(), null);
    if (scores != null) {
        processScores(scores);
    }
    return result;
}
```

Здесь вы просто генерируете подходящий объект типа URL. содержащий адрес URL сервера приложения, и используете метод `openStream()` для обращения к потоку данных с удаленного сервера приложения.

После вызова метода `setInput()` класса `XmlPullParser` вы можете использовать этот экземпляр класса `XmlPullParser` точно так же, как вы использовали экземпляр класса, который представлял данные локального ресурса.

Теперь переместите метод `processScores()` в класс `ScoreDownloaderTask`. Нужно обновить этот метод таким образом, чтобы он просто принимал параметр типа `XmlPullParser` и разбирал XML-данные, публикуя результаты игры после завершения их разбора при помощи метода `publishProgress()`:

```
private void processScores(XmlPullParser scores)
    throws XmlPullParserException, IOException {
    int eventType = -1;
    boolean bFoundScores = false;
```

```
    // Находим элементы score в XML-документе
    while (eventType != XmlResourceParser.END_DOCUMENT) {
        if (eventType == XmlResourceParser.START_TAG) {

            // Получаем название тега (например, scores или score)
            String strName = scores.getName();

            if (strName.equals("score")) {
            bFoundScores = true;
            String scoreValue =
            scores.getAttributeValue(null, "score");
            String scoreRank =
            scores.getAttributeValue(null, "rank");
            String scoreUserName =
                scores.getAttributeValue(null, "username");
            publishProgress(scoreValue, scoreRank, scoreUserName);
            }
        }
    eventType = scores.next();
    }
    // Обрабатываем ситуацию, когда результаты игры недоступны
    if (bFoundScores == false) {
        publishProgress();
    }
}
```

Метод `publishProgress()` может в любой момент времени вызываться из метода `doInBackground()`, в результате чего вызывается метод обработки события `onProgressUpdate()`. Благодаря этому фоновый процесс может взаимодействовать с потоком, управляющим пользовательским интерфейсом.

Обновление индикатора хода выполнения процесса

Чтобы получать информацию из фонового процесса в потоке, управляющем пользовательским интерфейсом, вы можете переопределить метод `onProgressUpdate()`. В этом примере вы получаете новый обработанный результат, переданный в виде параметров метода, и вставляете новую строку в элемент-контейнер `TableLayout`, отвечающий за отображение результатов.

```
@Override
protected void onProgressUpdate(String... values) {
    if (values.length == 3) {
        String scoreValue = values[0];
        String scoreRank = values[1];
        String scoreUserName = values[2];
        insertScoreRow(table, scoreValue, scoreRank, scoreUserName);
    } else {
        final TableRow newRow =
            new TableRow(QuizScoresActivity.this);
        TextView noResults =
            new TextView(QuizScoresActivity.this);
        noResults.setText(
            getResources().getString(R.string.no_scores));
```

```
            newRow.addView(noResults);
            table.addView(newRow);
        }
    }
```

Метод `insertScoreRow()` просто создает новый элемент `TableRow` и добавляет его в элемент-контейнер `TableLayout`. Массив значений должен всегда передаваться в одном и том же порядке. Это ограничение обусловлено реализацией класса `AsyncTask` в платформе Java.

Сброс индикатора хода выполнения процесса

Теперь вы реализуете метод `onPostExecute()`, который выполняется в потоке, управляющим пользовательским интерфейсом, после того, как фоновый процесс будет завершен. В частности, когда разбор и отображение данных о результатах игры будут завершены, на что указывает значение переменной `mProgressCounter`, вы можете скрыть индикатор хода выполнения процесса.

```
@Override
protected void onPostExecute(Boolean result) {
    Log.i(DEBUG_TAG, "onPostExecute");
    mProgressCounter--;
    if (mProgressCounter <= 0) {
        mProgressCounter = 0;
        QuizScoresActivity.this.
            setProgressBarIndeterminateVisibility(false);
    }
}
```

Преждевременное завершение фонового процесса

Вы можете преждевременно завершить выполнение фонового процесса, переопределив метод `onCancelled()`. Метод `onCancelled()` выполняется в потоке. управляющим пользовательским интерфейсом, и, если вызывается этот метол, метод `onPostExecute()` вызываться не будет. Таким образом, любые действия по освобождению ресурсов должны выполняться именно в методе `onCancelled()`. В этом примере мы выполняем следующую операцию:

```
@Override
protected void onCancelled() {
    Log.i(DEBUG_TAG, "onCancelled");
    mProgressCounter--;
    if (mProgressCounter <= 0) {
    mProgressCounter = 0;
    QuizScoresActivity.this.
        setProgressBarIndeterminateVisibility(false);
    }
}
```

Вызов метода `onCancelled()` происходит в том случае, когда вызывается метод `cancel()` класса `AsyncTask`. Метод `cancel()` не вызывается автоматически. Возь-

мите за правило отменять выполнение задач, в которых больше нет необходимости. Для экрана с результатами игры отменить выполнение задач можно в том случае, когда пользователь покинул его по какой-либо причине. Иначе говоря, вы завершаете их выполнение в методе `onPause()` деятельности, как показано ниже:

```
@Override
protected void onPause() {
    if (allScoresDownloader != null &&
        allScoresDownloader.getStatus() !=
        AsyncTask.Status.FINISHED) {
        allScoresDownloader.cancel(true);
    }
    if (friendScoresDownloader != null &&
        friendScoresDownloader.getStatus() !=
        AsyncTask.Status.FINISHED) {
        friendScoresDownloader.cancel(true);
    }
    super.onPause();
}
```

ЗАГРУЗКА И РАЗБОР НАБОРОВ ВОПРОСОВ

Теперь, когда вы понимаете, как происходит асинхронная загрузка данных, можете использовать класс `AsyncTask` для выполнения загрузки и отображения наборов вопросов на игровом экране. Этот процесс во многом напоминает процесс загрузки результатов игры. Однако вы не будете выполнять никаких обновлений экрана до завершения процесса, вместо этого вы будете просто отображать индикатор хода выполнения процесса до тех пор. пока не загрузятся все вопросы из заданного набора.

Сначала внутри класса `QuizGameActivity` вы создадите производным класс `QuizTask`, который расширяет класс `AsyncTask`, как показано в следующем коде:

```
private class QuizTask extends AsyncTask<Object, String, Boolean> {
    // TODO: Implement AsyncTask callback methods
}
```

КСТАТИ

Вы можете добавить отдельную переменную-член `DEBUG_TAG` в класс `QuizTask`. Это позволит вам различать отладочную информацию фонового потока и потока, управляющего пользовательским интерфейсом.

В класс `QuizTask` также нужно добавить переменные-члены для начального номера вопроса и экземпляра класса `ProgressDialog`, который будет представлять ход выполнения фонового процесса пользователю:

```
int startingNumber;
ProgressDialog pleaseWaitDialog;
```

Завершив реализацию класса `QuizTask`, вы можете обновить метод `onCreate()` класса `QuizGameActivity` так, чтобы при первой загрузке экрана происходил вызов метода

execute() класса QuizTask. Метод execute() принимает два параметра: адрес URL сервера, с которого будет происходить загрузка вопросов, и начальный номер вопроса (типа integer) для загружаемого набора:

```
public static final String TRIVIA_SERVER_QUESTIONS =
    TRIVIA_SERVER_BASE + "questions.jsp";
// ...
QuizTask downloader = new QuizTask();
downloader.execute(TRIVIA_SERVER_QUESTIONS, startingQuestionNumber)
```

Открытие диалогового окна, представляющего ход выполнения процесса

Теперь вы должны реализовать метод onPreExecute(). Это подходящее место для открытия диалогового окна, которое будет сообщать пользователю о том, что в настоящее время происходит загрузка вопросов викторины. Пользователь не сможет делать ничего, пока не будут загружены вопросы. И хотя ранее вы добавили индикатор хода выполнения процесса, связанного с загрузкой результатов, в строку заголовка приложения, вы хотите отображать диалоговое окно, представляющее ход выполнения процесса, поверх игрового экрана:

```
@Override
protected void onPreExecute() {
    pleaseWaitDialog = ProgressDialog.show(
        QuizGameActivity.this, "Trivia Quiz",
        "Downloading trivia questions", true, true);
    pleaseWaitDialog.setOnCancelListener(new OnCancelListener() {
        public void onCancel(DialogInterface dialog) {
            QuizTask.this.cancel(true);
        }
    });
}
```

И хотя в этом примере для простоты мы использовали жестко закодированные строки, в хорошо написанном приложении эти строки были бы заменены строковыми ресурсами, чтобы упростить локализацию приложения

Добавляется слушатель события отмены деятельности, Это позволяет пользователю нажать кнопку **Back** на мобильном телефоне, чтобы «крыть диалоговое окно до окончания загрузки вопросов. В этом случае происходит вызов метода cancel() класса AsyncTask. Иначе говоря, если пользователь закроет диалоговое окно раньше времени, произойдет преждевременное завершение задачи, что в свою очередь приведет к прекращению соответствующей сетевой операции.

Выполнение фоновой обработки данных

Теперь необходимо определить, какие операции должны выполняться асинхронно. Как и раньше, это операции, связанные с загрузкой и разбором данных. Следующий код (для простоты понимания обработка исключений была удалена из этого кода) демонстрирует переопределение метода doInBackgroun():

```
@Override
protected Boolean doInBackground(String... params) {
```

```
    boolean result = false;
    startingNumber = (Integer)params[1];
    String pathToQuestions = params[0] +
        "?max=" + QUESTION_BATCH_SIZE + "&start=" + startingNumber;
    result = loadQuestionBatch(startingNumber, pathToQuestions);
    return result;
}
```

В данном случае фоновая обработка данных подразумевает определение соответствующего набора вопросов для загрузки и вызов вспомогательного метода `loadQuestionBatch()`. Этот метод должен быть перемешен в класс `QuizTask` и обновлен соответствующим образом для подключения к серверу приложения. Как и раньше, функциональность данного метода включает генерацию адреса URL в нужном формате, открытие потока к удаленному серверу приложения и использование метода `setInput()` класса `XmlPullParser`.

КСТАТИ

Полную версию кода можно найти на диске, прилагаемом к данной книге.

Закрытие диалогового окна, представляющего ход выполнения процесса

Далее вы реализуете метод `onPostExecute()`. Теперь, когда фоновая обработка данных завершена, вы можете просто добавить в этот метод код, который изначально использовался для отображения экрана. Это также место для закрытия диалогового окна, представляющего ход выполнения процесса:

```
@Override
protected void onPostExecute(Boolean result) {
    Log.d(DEBUG_TAG, "Download task complete.");
    if (result) {
        displayCurrentQuestion(startingNumber);
    } else {
        handleNoQuestions();
    }

    pleaseWaitDialog.dismiss();
}
```

ПРИМЕЧАНИЕ

Полную версию кода можно найти на диске, прилагаемом к данной книге.

ИТОГИ

В этом часе вы изменили приложение «Been There, Done That!» таким образом. чтобы оно загружало данные, включая наборы вопросов викторины и результаты игры пользователей, с удаленного сервера приложения. Вы узнали, как использовать класс

`AsyncTask` для выполнения фоновой обработки данных и обеспечения оперативного реагирования вашего приложения на действия пользователя. Были также рассмотрены многие моменты, которые необходимо учитывать при разработке приложений для мобильных устройств, использующих сетевую функциональность.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Каков оптимальный размер данных для загрузки?

Ответ: Это каверзный вопрос. Если кратко: не настолько много, чтобы пользователь устал дожидаться момента начала работы с приложением, но достаточно для того, чтобы слишком часто подгружать очередные порции данных. В идеальном случае вся загрузка данных должна производиться незаметно для пользователя в то время, когда он занят другим делом, например, когда он отвечает на уже загруженные вопросы.

Вопрос: Где я могу найти дополнительную информацию о том, какие сетевые протоколы поддерживаются платформой Android?

Ответ: В инструментарии Android SDK доступно три пакета, реализующих сетевую функциональность: android.net, java.net и org.apache.

ПРАКТИКУМ
Контрольные вопросы

1. Где можно найти информацию о состоянии сети на мобильном телефоне под управлением операционной системы Android?

   A. В строке состояния.
   B. В приложении **Settings** (Настройки) операционной системы Android.
   C. Вызвав метод `getHandsetNetworkStatus()` класса `NetStatus`.

2. Верно ли это? Эмулятор Android не может имитировать скорость передачи данных и задержки, присущие настоящим мобильным устройствам, работающим под управлением операционной системы Android.

3. Верно ли это? Для создания серверов Android-приложений вы должны использовать платформу Google App Engine.

4. Что из нижеперечисленного не поддерживается платформой Android?

   A. HTTP.
   B. HTTPS.
   C. TCP.
   D. IP.
   E. Сокеты прямого доступа (RS — Raw Sockets).

Ответы

A и B. Некоторая основная информация о состоянии сети на мобильном телефоне действительно отображается в строке состояния, однако получить более детальную информацию о состоянии сети можно при помощи приложения **Settings** (Настройки) операционной системы Android.

Неверно. Эмулятор Android имеет ряд настроек для имитации скорости передачи данных и задержек сети.

Неверно. Вы можете использовать любую серверную технологию для реализации сервера приложения, с которым будет взаимодействовать ваше Android-приложение. Платформа Google App Engine — это всего лишь одна из таких технологий.

Вопрос с подвохом. Все перечисленные протоколы и сетевые технологии могут быть использованы в Android-приложениях. Протоколы HTTP и HTTPS могут использоваться для приложений, в которых применяются веб-технологии. Протоколы TCP и IP - это сетевые протоколы нижнего уровня, используемые платформой Android, и, кроме того, существуют стандартные интерфейсы API платформы Java для работы с сокетами прямого доступа.

Упражнения

1. Протестируйте приложение «Been There, Done That!» с учетом различных ситуаций, связанных с работой сети. Измените настройки эмулятора, чтобы имитировать медленную скорость передачи данных по сети, и затем запустите приложение и понаблюдайте за результатами.

2. Измените приложение так, чтобы вместо подхода с использованием класса `AsyncTask` применялся подход с использованием классов `Thread` и `Handler`.

3. Измените приложение таким образом, чтобы следующий набор вопросов викторины загружался в фоновом режиме до того момента, когда пользователь ответит на все существующие вопросы в текущем наборе.

Час 16. ДОБАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ СЕТЕВОЙ ПОДДЕРЖКИ

Вопросы, рассматриваемые в этом часе:

- **получение доступа к информации о телефонии;**
- **использование клиентских HTTP-сервисов;**
- **отправка GET-запросов по протоколу HTTP;**
- **отправка POST-запросов по протоколу HTTP;**
- **добавление в ваш проект JAR-файлов сторонних разработчиков;**
- **работа с составными MIME-сообщениями.**

В этом часе вы расширите функционал приложения «Been There, Done That!», добавив возможность выгружать данные игрока, включая его настройки, результат игры и аватар, на сервер приложения. Вы также узнаете, как можно получить доступ к информации о статусе телефонии мобильного устройства. Наконец, вы добавите несколько внешних библиотек в Android- проект и будете работать с составными MIME-сообщениями.

ОПРЕДЕЛЕНИЕ ДАННЫХ ДЛЯ ОТПРАВКИ НА СЕРВЕР

До сих пор в приложении «Been There, Done That!» вы только загружали данные с сервера. Теперь настало время выгрузить информацию об игроке на сервер приложения. Для этого вам нужно познакомиться с возможностями HTTP-клиента Apache, доступного на платформе Android, и узнать о том, как добавлять дополнительные библиотеки Apache в проект.

Выгружать данные на сервер приложения должны три части приложения «Been There, Done That!»:

- `QuizSettingsActivity` — этот класс должен создавать запись игрока на сервере приложения и выгружать информацию о настройках данного игрока.

- `QuizGameActivity` — этот класс должен выгружать результаты игры Для данного игрока.

- `QuizSettingsActivity` — этот класс должен выгружать изображение аватара.

Чтобы приложение "Been There, Попе Thai!» могло работать с данными в реальном режиме времени, нам потребуется как доступ к серверу приложения. так и добавление соответствующей сетевой функциональности в клиентское Android -приложение.

КСТАТИ

Полная версия кода, реализующего сетевую поддержку в приложении и рассматриваемого в этом часе, доступна на диске, прилагаемом к книге.

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О СОСТОЯНИИ ТЕЛЕФОНА

Некоторые настройки игрока приложения «Been There, Done That!» будут выгружаться на сервер приложения. Сервер приложения должен иметь возможность определить, с каким игроком (или устройством) он взаимодействует в настоящий момент. По этой причине приложение использует уникальный идентификатор мобильного устройства каждого игрока для отличия игроков друг от друга. Чтобы получить этот уникальный идентификатор, вам потребуется обратиться к информации, предоставляемой классом TelephonyManager.

Помимо разнообразных библиотек для работы с сетевыми возможностями, которые доступны в инструментарии Android SDK и были рассмотрены ранее, вы также можете получить информацию о состоянии телефона и телефонии, используя класс `TelephonyManager`. Класс `TelephonyManager` позволяет Android-приложению получать информацию об используемой сотовой сети, подключении, модуле идентификации абонента (SIM — Subscriber Identity Module), а также о самом мобильном устройстве.

ВНИМАНИЕ!
Под управлением операционной системы Android работают не только мобильные телефоны. Поддержка сервисов телефонии может значительно варьироваться в зависимости от типа телефона и поставщика услуг, или вообще может быть недоступна.

Настройка разрешений для получения информации о состоянии телефона

Для получения информации о состоянии телефона Android-приложение должно иметь соответствующие разрешения, устанавливаемые в файле манифеста Android. Android-приложение может получать или изменять информацию о состоянии телефона только в том случае, если в элементе `<uses-permission>` файла манифеста Android будут указаны необходимые значения.

Ниже перечислены наиболее часто используемые значения разрешений *0*[я] приложений, работающих с информацией о состоянии телефона:

- `android.permission.READ_PHONE_STATE`
- `android.permission.MODIFY_PHONE_STATE`

Получение информации о телефонии

Разработчики могут использовать класс `TelephonyManager(android.telephony.TelephonyManager)` для получения информации о состоянии сети на устройстве программным путем. Получить экземпляр класса `TelephonyManager` можно при помощи знакомого вам метода `getSystemService()` объекта типа `Context` приложения:

```
TelephonyManager telMgr = (TelephonyManager)
    getSystemService(Context.TELEPHONY_SERVICE);
```

Получив экземпляр класса `TelephonyManager`, вы можете запросить подробную информацию об устройстве и его сервисах телефонии.

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О СОСТОЯНИИ ЗВОНКА

Вы можете использовать класс `TelephonyManager` для определения состояния звонка сотового устройства при помощи метода `getCallstate()`. Этот метод сообщает, находится ли телефон в состоянии готовности, ожидает ли ответа абонента на вызов или же в настоящий момент принимает входящий звонок.

Вы также можете зарегистрировать слушателя для получения событий об изменении состояний звонка, используя метод `listen()`, благодаря чему ваше приложение сможет получать уведомления при поступлении входящих звонков. Эту информацию можно использовать для сокрытия определенных телефонных номеров или для подготовки приложения к тому, что пользователь ответит на входящий звонок.

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ПРОТОКОЛЕ СЕТИ

Вы можете воспользоваться классом `TelephonyManager` для запроса информации об используемом протоколе сотовой сети при помощи метода `getNetworkType()`. Этот метод сообщит вам о том, какой протокол используется в текущий момент на устройстве, например, GPRS, EDGE или EVDO. Эти и многие другие протоколы сети определены в виде констант в классе `TelephonyManager` (например, `NETWORK_TYPE_GPRS`).

Вы можете также зарегистрировать слушателя для получения событий об изменении используемого протокола сети при помощи метода `listen()`. Подобная информация нужна для определения момента переключения на высокоскоростное соединение, чтобы загрузить в фоновом режиме наборы вопросов викторины больших размеров и выполнить их кэширование или изменить размер набора для загрузки.

ОПРЕДЕЛЕНИЕ ПОДДЕРЖИВАЕМОГО ТЕЛЕФОНОМ СТАНДАРТА: CDMA ИЛИ GSM

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О SIM-КАРТЕ

Вы можете использовать метод `getPhoneType()` класса `TelephonyManager`, чтобы определить, какой стандарт поддерживается мобильным устройством: CDMA или GSM. Более того, вы можете получить уникальный идентификатор мобильного телефона при помощи метода `getDeviceId()`. Этот метод возвращает IMEI-идентификатор для GSM-телефонов и MEID-идентификатор для CDMA-телефонов. Вы также можете получить уникальный идентификатор абонента (например, IMSI-идентификатор для абонентов GSM-сетей), используя метод `getSubscriberId()`.

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О SIM-КАРТЕ

Класс TelephonyManager предоставляет ряд методов для получения информации о SIM-карте и соответствующем операторе сотовой связи. Вот некоторые из этих методов:

`getSimState()` — этот метод возвращает состояние SIM-карты, например, вставлена ли SIM-карта в мобильное устройство, заблокирована ли она (т. е. необходимо ввести PIN-код) или готова к использованию.

`getSimSerialNumber()` — этот метод возвращает уникальный серийный номер SIM-карты.

`getSimOperatorName()` — этот метод возвращает имя оператора для данной SIM-карты.

`getSimOperator()` — этот метод возвращает мобильный код страны и код сети для оператора, выпустившего данную SIM-карту.

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ГОЛОСОВОЙ ПОЧТЕ УСТРОЙСТВА

Вы можете использовать класс `TelephonyManager` для получения информации о голосовой почте пользователя телефона. Например, вы можете получить телефонный номер для доступа к голосовой почте при помощи метода `getVoiceMailNumber()`.

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О РОУМИНГЕ

В часе 15 было рассмотрено, как можно получить информацию о роуминге, используя сетевые сервисы. Вы также можете определить, находится ли абонент в роуминге, используя метод `isNetworkRoaming()` класса `TelephonyManager`. Этот подход не позволяет с уверенностью сказать, что пользователь будет нести дополнительные расходы за передачу данных или телефонные звонки, но такая вероятность есть. Приложение может использовать эту информацию, чтобы запросить разрешение у пользователя на передачу данных по сети, находясь в роуминге, отобразив сообщение наподобие следующего: «Ваш телефон находится в роуминге. Вы хотите продолжить?»

ПОЛУЧЕНИЕ ДРУГОЙ ИНФОРМАЦИИ, ОТНОСЯЩЕЙСЯ К ТЕЛЕФОНИИ

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Вы можете использовать класс SmsManager (android.telephony.SmsManager) и класс SmsMessage (android.telephony.SmsMessage) для отправки SMS-сообщений.

ВЫГРУЗКА ДАННЫХ НА УДАЛЕННЫЙ СЕРВЕР ПРИЛОЖЕНИЯ

В часе 10 вы создали экран с настройками и использовали для хранения введенных пользователем данных экземпляр класса `SharedPreferences`. Теперь вы измените класс этого экрана таким образом, чтобы копия настроек игрока выгружалась на сервер (помимо хранения этих настроек в экземпляре класса `SharedPreferences`). Благодаря этой реализации клиентское устройство будет всегда иметь последнюю версию данных. Сервер приложения просто сохраняет копию настроек. В этом конкретном приложении отсутствует двухсторонняя синхронизация; рассмотрение этой темы выходит за рамки данной книги и для ее объяснения потребовалось бы больше часа времени.

КСТАТИ

Чтобы упростить понимание кода и продемонстрировать важную сетевую функциональность, мы позволили себе некоторые вольности в отношении архитектуры приложения и не всегда использовали сетевые возможности максимально эффективно. Тем не менее примененные нами подходы позволяют продемонстрировать важный функционал, например фоновое обновление данных, индикаторы хода выполнения процесса и другие возможности класса `AsyncTask`.

Для взаимодействия с сервером приложения вы можете использовать пакет `HttpClient (org.apache.http)`, включенный в состав инструментария Android SDK. Этот пакет предоставляет классы для реализации большого числа сценариев сетевого взаимодействия по протоколу HTTP в вашем приложении.

Вы узнаете, как использовать класс HttpGet для отправки переменных запроса аналогично тому, как происходит отправка веб-формы с применением GET-метода протокола HTTP. Затем вы познакомитесь, как использовать класс HttpPost для отправки переменных запроса и выгрузки изображения аватара аналогично тому, как происходит отправка веб-формы с применением POST-метода протокола HTTP.

Сервер приложения был написан с прицелом на обычные веб-формы языка HTML. Фактически, до того, как был написан Android-клиент, тестирование сервера производилось с использованием стандартной HTML-формы.

Еели вы разработаете веб-клиента раньше, чем будет разработан Android-клиент, вы сможете убедиться, что протоколы взаимодействия между клиентом и сервером стандартные и кроссрпатформенные. Используя эту процедуру, вы будете знать, что любая платформа — включая платформу Android, способная использовать GET- и POST-методы для отправки данных формы, будет совместима с данным сервером приложения. В этом случае приложение может использовать библиотеки Apache HTTP — в основном, пакет `org.apache.http.client`.

КСТАТИ

Код сервера приложения, используемого в этой книге, доступен в папке **/Книга/ Сервер** на прилагаемом к книге диске.

ВНИМАНИЕ!

Сетевые операции, связанные с выгрузкой данных на сервер, могут занимать некоторое время. Поэтому вызовы всех методов, подразумевающих работу с сетью, должны выполняться асинхронно, отдельно от основного потока, управляющего пользовательским интерфейсом. Это можно сделать при помощи класса Thread платформы Java или при помощи класса `AsyncTask` платформы Android. Также обязательно предупреждайте пользователя о длительных операциях, используя, например, элемент `ProgressBar`. Дополнительную информацию можно найти в часе 15.

Выгрузка данных игрока при помощи GET-метода протокола HTTP

Отправка данных игрока на сервер приложения посредством GET-метода протокола HTTP осуществляется с использованием класса `HttpGet`. Поскольку вы отправляете переменные запроса, нужно использовать вспомогательный класс `URLEncodedUtils` вместе с экземпляром класса `List`, содержащим объекты типа `BasicNameValuePair`, что поможет сформировать окончательный адрес URL для данного запроса. Наконец, вся процедура взаимодействия с сетью должна быть представлена в виде объекта типа `AsyncTask`, чтобы пользовательский интерфейс мог продолжать реагировать на действия пользователя в то время, пока будет обрабатываться сетевой запрос.

СОЗДАНИЕ КЛАССА ASYNCTASK, ПРЕДНАЗНАЧЕННОГО ДЛЯ ВЫГРУЗКИ ДАННЫХ

Для выгрузки информации о настройках пользователя на сервер приложения вам потребуется добавить еще один класс `AccountTask`, производный от класса `AsyncTask`, в класс `QuizSettingsActivity`. Класс `AccountTask` может выполняться в результате вызова метода `execute()` при каждом изменении значения на экране с настройками (рис. 16.1). Вы также можете выполнять эту задачу при первом запуске приложения пользователем, чтобы настроить уникальный идентификатор игрока на сервере приложения для данного игрока на данном устройстве, даже если пользователь не ввел свой ник или адрес электронной почты. Вы также можете получить значения настроек с сервера (путем использования двухсторонней синхронизации).

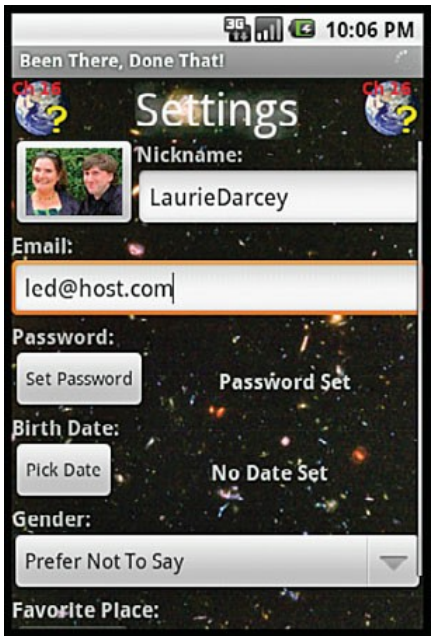

Рис. 16.1 Использование индикатора непрерывного хода выполнения процесса в верхнем правом углу строки меню (анимированный круг)

Реализация произвольного класса `AccountTask` во многом похожа на другие произвольные классы, унаследованные от класса `AsyncTask`, над которым вы работали на протяжении последних двух часов. (В следующем примере для простоты понимания и краткости был удален код, обрабатывающий исключения.) Здесь заслуживает внимания метод `doInBackground()`, в котором происходит передача настроек игрока на сервер приложения:

```
@Override
protected Boolean doInBackground(Object... params) {
Boolean succeeded = false;

Integer playerId =
mGameSettings.getInt(GAME_PREFERENCES_PLAYER_ID, -1);
String nickname =
mGameSettings.getString(GAME_PREFERENCES_NICKNAME, "");
String email =
mGameSettings.getString(GAME_PREFERENCES_EMAIL, "");
String password =
```

```
mGameSettings.getString(GAME_PREFERENCES_PASSWORD, “”);
Integer score =
mGameSettings.getInt(GAME_PREFERENCES_SCORE, -1);
Integer gender =
mGameSettings.getInt(GAME_PREFERENCES_GENDER, -1);
Long birthdate =
mGameSettings.getLong(GAME_PREFERENCES_DOB, 0);
String favePlaceName =
   mGameSettings.getString(GAME_PREFERENCES_FAV_PLACE_NAME, “”);

Vector<NameValuePair> vars = new Vector<NameValuePair>();

if (playerId == -1) {
    TelephonyManager telManager =
    Using an indeterminate
    progress indicator
    in the top
    right of the
    menu bar (faint
    circle).
    (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
    String uniqueId = telManager.getDeviceId();
    vars.add(new BasicNameValuePair(“uniqueId”, uniqueId));

} else {
    vars.add(
     new BasicNameValuePair(“updateId”, playerId.toString()));
    vars.add(
        new BasicNameValuePair(“score”, score.toString()));
}

vars.add(new BasicNameValuePair(“nickname”, nickname));
vars.add(new BasicNameValuePair(“email”, email));
vars.add(new BasicNameValuePair(“password”, password));
vars.add(new BasicNameValuePair(“gender”, gender.toString()));
vars.add(new BasicNameValuePair(“faveplace”, favePlaceName));
vars.add(new BasicNameValuePair(“dob”, birthdate.toString()));

String url =
TRIVIA_SERVER_ACCOUNT_EDIT+ “?” + URLEncodedUtils.format(vars, null);

HttpGet request = new HttpGet(url);
ResponseHandler<String> responseHandler =
    new BasicResponseHandler();
HttpClient client = new DefaultHttpClient();
String responseBody = client.execute(request, responseHandler);

if (responseBody != null && responseBody.length() > 0) {
    Integer resultId = Integer.parseInt(responseBody);
    Editor editor = mGameSettings.edit();
    editor.putInt(GAME_PREFERENCES_PLAYER_ID, resultId);
    editor.commit();
    succeeded = true;
}
return succeeded;
}
```

Пусть этот относительно большой фрагмент кода вас не обескураживает. На самом деле в нем нет ничего сложного: вы создаете вектор, состоящий из пар имя/значение для каждой отдельно взятой настройки игрока (сохраненных в экземпляре класса `SharedPreferences`), которые вы хотите передать на сервер.
Если настройки для данного игрока передаются на сервер впервые, будет создана новая запись. Новая запись представляет собой пустую запись игрока на сервере с одним лишь значением: уникальным идентификатором устройства. Благодаря этому в вашем распоряжении окажется разумный идентификатор для игрока еще до того момента, когда он начнет изменять значения других настроек. Также пользователь сможет изменять другую информацию, например, свой ник, без непредвиденных последствий. Вы должны сохранить результирующий идентификатор, поскольку он будет использоваться во всех последующих запросах.

Установив все необходимые переменные запроса, вы можете приступать непосредственно к генерации запроса. Для преобразования вектора переменных в GET-запрос вы можете использовать удобный метод `URLEncodedUtils.format()`. Наконец, вы инициализируете объект типа HttpGet, передаете в него сформированную строку GET-запроса и осуществляете отправку запроса при помощи экземпляра класса `HttpClient`.

Класс `HttpClient` предоставляет вспомогательные инструменты в виде классов `ResponseHandler` для разбора ответа от сервера. В данном случае класс `BasicResponseHandler` используется для простого получения типа `String`. В этом случае объект типа `String` содержит уникальный идентификатор игрока, который может быть сохранен в экземпляре класса `SharedPreferenees` для дальнейшего использования.

КСТАТИ

В большинстве случаев следует избегать отправки конфиденциальных данных по сети в открытом виде. Например, идентификатор устройства может быть использован злоумышленником в выгодных для него целях. Поскольку все, что нужно для данного приложения, — это некий уникальный, но многократно отправляемый на сервер идентификатор, вы можете использовать однонаправленную хеш-функцию, например, алгоритм `Secure Hash Algorithm (SHA)`, для криптозащиты идентификатора устройства. Класс `MessageDigest`, являющийся частью пакета java. security, предоставляет необходимую функциональность:

```
MessageDigest sha = MessageDigest.getInstance("SHA");
byte[] enc = sha.digest(uniqueId.getBytes());
```

В полной реализации данного примера, которую можно найти на диске, прилагаемом к данной книге, на сервер происходит отправка хешированного значения.

ВЫГРУЗКА ИНФОРМАЦИИ О НАБРАННЫХ ОЧКАХ ИГРОКА

Для выгрузки информации о набранных очках игрока на сервер приложения можно добавить в класс `QuizGameActivity` еще один класс, производный от класса `AsyncTask`. Но почему просто не обновить существующий класс `QuizTask`, чтобы он осуществлял передачу информации о набранных очках игрока на сервер приложения каждый раз, когда в приложении происходит загрузка новых вопросов? Это помогло бы уменьшить задержки в приложении и повысить эффективность использования сети.

Вы можете просто добавить информацию о набранных очках игрока в виде переменной запроса, отправляемого на сервер из метода `doInBackground()` класса `QuizTask`. Для этого используется следующий код:

```
SharedPreferences settings =
    getSharedPreferences(GAME_PREFERENCES, Context.MODE_PRIVATE);
Integer playerId = settings.getInt(GAME_PREFERENCES_PLAYER_ID, -1);
if (playerId != -1) {
    Log.d(DEBUG_TAG, "Updating score");
    Integer score = settings.getInt(GAME_PREFERENCES_SCORE, -1);
    if (score != -1) {
    pathToQuestions +=
       "&updateScore=yes&updateId="+playerId+"&score="+score;
    }
}
```

Этот код добавляется сразу после кода, отвечающего за генерацию значения переменной `pathToQuestions`, преладставляющей результирующий адрес URL, но перед кодом, использующим это значение, позволяя модифицировать данное значение.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Вы также могли бы использовать контейнер типа `List`, содержащий объекты типа `BasicNameValuePair`, вместе с методом `URLEncodeUtils.format()`. Тем не менее для простых значений, например целых чисел, дополнительное кодирование не требуется.

Выгрузка аватаров пользователей при помощи POST-метода протокола HTTP

Для выгрузки двоичных данных, например изображения аватара, вместо GET-запроса вам потребуется воспользоваться POST-запросом протокола HTTP. Это позволяет отправлять изображение аватара наподобие того, как происходит отправка файлов при использовании обычной веб-формы языка HTML. Сервер приложения затем может обработать POST-запрос соответствующим образом и сохранить копию аватара игрока (рис. 16.2).

Рис. 16.2. Выгрузка аватара

ВНИМАНИЕ!

При сохранении данных непримитивных типов принимайте во внимание лучшие практики построения хранилищ данных, используемых вашим сервером приложения. Например, при программировании SQL-ориентированных баз данных принято хранить в базе данных адреса изображений, но не их содержимое (в виде значения типа BLOB).

ДОБАВЛЕНИЕ JAR-ФАЙЛОВ В ВАШ ANDROID-ПРОЕКТ

В идеальном случае для выгрузки изображения аватара и другой важной информации на сервер приложения в виде составного MIME-сообщения удобно использовать класс `HttpClient`. Тем не менее, на момент написания данной книги поддержка класса `Apache HttpClient` в инструментарии Android SDK была неполной. Инструментарий Android SDK пока еще не имеет поддержки составных MIME-сообщений, но высока вероятность, что этот функционал появится в одной из будущих версий инструментария SDK. Пока же, если вы хотите добавить поддержку составных MIME-сообщений, необходимо включить несколько библиотек Apache в ваш проект в виде JAR- файлов. В частности, нужно добавить следующие JAR-файлы в ваш проект:

- Mime4j (**james.apache.org/mime4j/index.html**);
- HttpMime 4.0 (**hc.apache.org/httpcomponents-client-ga/httpmime/index. html**);
- Apache Commons 10 (**commons.apache.org/io/).**

КСТАТИ

Все вышеперечисленные библиотеки можно найти на компакт-диске, прилагаемом к данной книге, в папке **/Книга/Час 15 и 16/libs**.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Не знаете, что такое составные MIME-сообщения? Отличное описание доступно на сайте Wikipedia: **en.wikipedia.org/wiki/MIME#Multipart_messages**. По существу, составное MIME-сообщение — это способ представления нескольких фрагментов данных, включая бинарные данные, в виде одного текстового сообщения. Для генерации составных MIME-сообщений при помощи HTML-формы атрибут enctype тега form должен быть установлен в значение multipart/form-data. Использование составных MIME-сообщений не ограничено только протоколом HTTP. Например, составные MIME-сообщения часто применяются в электронной почте.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО ДОБАВЛЕНИЕ JAR-ФАЙЛА В ANDROID-ПРОЕКТ

Чтобы добавить JAR-файл в Android-проект, выполните следующие шаги:

1. Загрузите JAR-файл(ы), который вы хотите включить в ваш проект.
2. Создайте каталог с именем **/libs** в вашем проекте. Эта папка должна находиться на том же уровне, что и папки **/src** и **/res**.
3. Скопируйте JAR-файлы) в каталог/libs.
4. В меню среды разработки Eclipse выберите команду **Project=>Properties** (Проект => Свойства), чтобы открыть диалоговое окно со гв^исгвами вашего Android-

проекта. Выберите категорию **Java Build Path** (Путь сборки Java) в левой части диалогового окна и откройте вкладку **Libraries** (Библиотеки).

5. Нажмите на кнопку **Add JARs** (Добавить JAR-файлы) и в открывшемся диалоговом окне **JAR Selection** (Выбор JAR-файлов) выберите JAP-файлы, которые вы хотите добавить в проект. Нажмите на кнопку **ОК**.

6. При необходимости обновите проект. Теперь вы можете приступать к программированию!

Чтобы добавить полнофункциональную поддержку составны/ MIME-сообщений в приложение «Been There, Done That!», вы должны добавить все три JAR-фэйла, перечисленных выше. Эти пакеты также содержат другие полезные инструменты, например, класс lOUtils, который предоставляет ряд удобных методов для работы с потоками данных.

СОЗДАНИЕ КЛАССА ASYNCTASK ДЛЯ ВЫПОЛНЕНИЯ ВЫГРУЗКИ ИЗОБРАЖЕНИЯ АВАТАРА

Для выгрузки изображения аватара на сервер приложения вам потребуется добавить класс `QuizSettingsActivity` еще один класс `ImageUploadTask`, производный от класса `AsyncTask`. Выполнение методов класса imageUploadTask может быть запушено с использованием метода `execute()` при каждом изменении игроком изображения аватара на экране с настройками.

Реализация класса `ImageUploadTask` очень похожа на реализацию других классов, производных от класса `AsyncTask`, которые были написаны в течение последних двух часов. Единственный действительно интересный момент в этой новой задаче — код метода `doInBackground()`, осуществляющий выгрузку файла аватара на сервер (из кода для простоты понимания и краткости удалена обработка исключений):

```
@Override
protected Boolean doInBackground(Object... params) {
    String avatar =
        mGameSettings.getString(GAME_PREFERENCES_AVATAR, "");
    Integer playerId =
        mGameSettings.getInt(GAME_PREFERENCES_PLAYER_ID, -1);

    MultipartEntity entity =
        new MultipartEntity(HttpMultipartMode.BROWSER_COMPATIBLE);
    File file = new File(avatar);
    FileBody encFile = new FileBody(file);
    entity.addPart("avatar", encFile);
    entity.addPart("updateId", new StringBody(playerId.toString()));

    HttpPost request = new HttpPost(TRIVIA_SERVER_ACCOUNT_EDIT);
    request.setEntity(entity);

    HttpClient client = new DefaultHttpClient();
    ResponseHandler<String> responseHandler =
        new BasicResponseHandler();
    String responseBody = client.execute(request, responseHandler);

    if (responseBody != null && responseBody.length() > 0) {
        Log.w(DEBUG_TAG,
           "Unexpected response from avatar upload: " + responseBody);
    }
```

```
    return null;
}
```

Вам потребуется создать составной MIME-объект, аналогичный MIME-объектам, создаваемым браузерами, при помоши класса MultipartEntity, и добавить в него две части: одна часть будет представлять содержимое файла аватара, а другая — уникальный идентификатор игрока, которому принадлежит данный аватар. Далее вы генерируете объект типа `HttpPost` с адресом URL сервера приложения. Наконец вы передаете созданный MIME-объект в объект типа `HttpPost`, используя метод `setEntity()`. Чтобы отправить запрос на сервер и получить ответ, вы, как обычно, используете классы `HttpClient` и `ResponseHandler`.

ИТОГИ

В этом часе вы изменили приложение «Been There, Done That!» таким образом. чтобы оно могло выгружать данные игры, включая настройки игрока, изображение аватара и набранные очки, на удаленный сервер приложения. Также вы узнали, как получать информацию, относящуюся к телефонии, например, информацию об используемой сотовой сети и информацию о нахождении абонента в роуминге, при помощи класса TelephonyManager. Кроме того, было рассмотрено использование GET- и POST-методов протокола HTTP для выгрузки данных на сервер при помощи класса `HttpClient`.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Как можно избежать передачи данных пользователя по сети в открытом виде?

Ответ: Существует множество способов защитить данные пользователя в процессе передачи по сети. Например, вы можете зашифровать все данные, передаваемые по протоколу Н П Р, с использованием крипшрафиче- ского протокола SSL (протокол HTTPS представляет собой расширение протокола HTTP, поддерживающее шифрование). Пароли которые уже были отправлены по защищенному каналу, в дальнейшем могут отправляться в хешированном виде с использованием класса `MessageDigest`.

Вопрос: На платформе Android доступна поддержка формата JSON (JavaScript Object Notation)?

Ответ: Да, библиотеки, предназначенные для работы с форматом JSON. доступны в пакете `org. json`, входящем в состав инструментария Android SDK,

ПРАКТИКУМ
Контрольные вопросы

1. Какие из перечисленных ниже сведений может предоставить класс `TelephonyManager`?

   A. Состояние звонка.
   B. Используемая сотовая сеть.
   C. Имя абонента.

2. Верно ли это? Инструментарий Android SDK предоставляет полнофункциональную поддержку составных MIME-сообщений.

3. Верно ли это? Сетевые операции всегда должны выполняться в потоке, управляющем пользовательским интерфейсом, поскольку эти операции очень быстрые.

4. Какие из нижеперечисленных классов или объектов не могут быть использованы для выполнения задач в фоновом режиме?

A. `BackgroundTask`.
B. `AsyncTask`.
C. `Thread`.
D. `AsyncActivity`.

Ответы

A и B. Класс `TelephonyManager` позволяет получать информацию о состоянии звонка и об используемой сотовой сети.

Неверно. Встроенная поддержка составных MIME-сообщений отсутствует. Вместо этого в данном часе было продемонстрировано, как можно добавить в проект внешние библиотеки, обеспечивающие поддержку составных MIME-сообщений.

Неверно. Операции, дли выполнения которых фебуетси некоторое время. например сетевые онераиин. никогда не должны выполнять в потоке. управляющем пользовательским ина'рфейсом, чтобы мобильный телефон мог оперт ивно реагировать на действия пользователя.

A и D. Фактически класс `AsyncTask` — это вспомогательный класс, который упрощает использование класса `Thread`. Использоваться могут оба класса. Два других класса не входят в состав инструментария SDK, а может и не существуют вовсе.

Упражнения

1. Используйте хешированный уникальный идентификатор устройства и адрес URL **http://tqs.mamlambo.com/getplayer**, чтобы отправить GET-запрос для получения информации об игроке и загрузки настроек игрока с сервера. Чтобы получить общедоступные данные, передайте в запросе только переменную `playerId`, а для получения всех данных передайте в запросе переменные `playerId` и `password` (эти данные вам понадобятся для полного восстановления информации).

2. Добавьте в приложение новый функционал, позволяющий игрокам предлагать новые вопросы викторины посредством отправки SMS-со- обшения разработчику.

3. Добавьте в приложение новый функционал, позволяющий игрокам предлагать новые вопросы викторины — вместе с изображениями — путем загрузки составных MIME-сообшений с использованием POST- метода на адрес **http://tqs.mamlambo.com/suggest.** Запрос должен содержать переменные `playerId` (идентификатор игрока), **question** (текст вопроса) и `questionImage` (изображение,

относящееся к вопросу), при этом для обработки данных изображения может использоваться тот же подход, который применялся для обработки данных изображения аватара (как было рассмотрено ранее в этом часе).

Час 17. ДОБАВЛЕНИЕ СОЦИАЛЬНЫХ ВОЗМОЖНОСТЕЙ

Вопросы, рассматриваемые в этом часе:

- **улучшение приложений путем добавления социальных возможностей;**
- **добавление поддержки функции приглашения друзей;**
- **отображение очков, набранных друзьями;**
- **интеграция приложения со сторонними социальными сетевыми сервисами.**

В этом часе вы улучшите приложение «Been There, Done That!», добавив в него некоторую интеграцию с социальными сервисами. В частности, вы измените приложение таким образом, чтобы пользователь мог отслеживать результаты других игроков. Вы также познакомитесь с различными приемами использования возможностей социальных сервисов и сторонних сайтов социальных сетевых сервисов в Android-приложениях, чтобы повысить привлекательность игры для пользователей.

УЛУЧШЕНИЕ ВАШЕГО ПРИЛОЖЕНИЯ ПРИ ПОМОЩИ ВОЗМОЖНОСТЕЙ СОЦИАЛЬНЫХ СЕРВИСОВ

За последние несколько часов приложение «Been There, Done That!» приобрело вполне завершенный вид. Тем не менее постоянно играть в одиночку довольно скучно. В идеальном случае пользователи хотят делиться своими результатами и впечатлениями от игры с другими людьми. Приложения, предоставляющие определенные возможности для взаимодействия между пользователями, с большей долей вероятности приобретут популярность и будут очень быстро распространяться, тем самым гарантируя успех их создателям.

Приложения с поддержкой социальных сервисов можно примерно разделить на две категории: те, которые предоставляют непосредственный доступ к социальным сетям, например MySpace или Facebook, и те, которые используют социальную информацию для улучшения восприятия игры пользователем. Программу «Been There, Done That!» можно с уверенностью отнести ко второй категории.

Добавление социальных возможностей в ваше приложение

Придумать, какие именно социальные и интерактивные возможности будут добавлены в ваше приложение, непросто. Вы, как разработчик приложения, можете задать себе следующие вопросы:

- Какие социальные возможности (если они есть) будут уместны в данном случае? Будут ли эти возможности использоваться для поддержания духа соперничества (сравнение лучших результатов игры, отправка уведомлении, когда один из друзей превзошел результат самого пользователя, и т. д.)? Будет ли приложение использовать возможности социальных сетей для информирования пользователей о последних изменениях в игре (публиковать имена рекордсменов игры в ленте Facebook или Twitter), тем самым обеспечивая бесплатную рекламу приложению?

- Каким образом конечный пользователь сможет пригласить в мое приложение других людей? Потребуется ли вводить адреса электронной почты, телефонные номера или имена друзей для связи с ними? Будут ли приглашения отправляться по электронной почте? Через SMS- сообщения? Должны ли оба игрока подтвердить, что они знают друг друга?

- Какие социальные сети (например, Facebook, Twitter и т. д.) посещают целевые пользователи моего приложения, и имеет ли смысл интегрировать в приложение какие-либо из возможностей этих сайтов?

- Как мое приложение будет защищать конфиденциальные данные пользователей (и их друзей)? Каких рекомендаций я буду придерживаться для определения того, что может и чего не может делать приложение (и моя компания) с конфиденциальными данными пользователей?

Поддержание основных отношений между игроками

Приложения, поддерживающие социальные возможности, основаны на определенных отношениях между пользователями. В разных приложениях для описания этих отношений используется различная терминология. Чаще всего для описания отношений между пользователями используются термины «контакт» и «друг» («френд»), однако на некоторых сайтах используется иная терминология, например «круг доверия» и «читатели». В хорошо продуманных приложениях для приглашения «друзей» и «контактов» может учитываться контекст игры. Например, в игре на военную тематику можно было бы использовать фразу «Призовите бывалых солдат для выполнения миссии» вместо типичной «Пригласите ваших друзей поиграть в эту игру» указав их адреса электронной почты».

ДОБАВЛЕНИЕ ПОДДЕРЖКИ КРУГА ДРУЗЕЙ В ВАШЕ ПРИЛОЖЕНИЕ

Вы добавите простую социальную возможность в приложение There Pone That!», которая позволит игрокам видеть количество очков, набранных другими игроками. Это достаточно простой способ заинтересован игроков распространение только «открытой» информации о набранных очках позволит вам избежать необходимости реализации функциональности, осуществляющей проверку и подтверждение отношений между пользователями.

Простая социальная возможность, которая будет добавлена в этом часе, работает следующим образом:

1. Игрок указывает адрес электронной почты своего друга, чтобы добавить этого человека в свой круг друзей.

2. Если указанный адрес электронной почты принадлежит другому игроку, зарегистрированному на сервере приложения, создается дружественная связь.

3. Каждый из игроков теперь может видеть очки, набранные другим игроком, на вкладке Scores of Friends (Результаты друзей) экрана с результатами игры.

КСТАТИ

Полная версия кода, рассматриваемого в этом часе, доступна на диске, прилагаемом к данной книге.

Добавление возможности приглашения друзей на экран с настройками

Чтобы добавить поддержку круга друзей в приложение «Been There, Done That!», вам потребуется изменить класс `QuizSettingsActivity`, чтобы позволить пользователю вводить адреса электронной почты своих друзей. В частности, вам нужно сделать следующее:

- Добавить кнопку на экран с настройками, при нажатии на которую будет открываться новое диалоговое окно.

- Добавить в класс `QuizSettingsActivity` реализацию нового диалогового окна, позволяющего пользователю вводить адрес электронной почты своего друга.

- Добавить реализацию некоторых сетевых операций, связанных с передачей на сервер приложения запроса на добавление пользователя в круг друзей.

ОБНОВЛЕНИЕ МАКЕТА ЭКРАНА С НАСТРОЙКАМИ

Вам потребуется обновить пользовательский интерфейс приложения «Been There, Done That!», чтобы игрок мог ввести адреса электронной почты своих Друзей. Безусловно, сделать это можно несколькими различными способами. Можно добавить новую деятельность и обновить экран с основным меню, создав тем самым новый экран в приложении, или можно просто обновить экран с настройками, добавив новую область.

Рис. 17.1. Обновленный экран с настройками,
позволяющий создавать запросы
на добавление пользователей в круг друзей

Проще всего добавить новую область в нижнюю часть экрана с настройками (рис. 17.1).

Например, вы могли бы добавить следующий фрагмент кода макета непосредственно под элементами пользовательского интерфейса, предназначенными для выбора избранного места пользователя:

```
<TextView
      android:id="@+id/TextView_Friend_Email"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/settings_friend_email_label"
      android:textSize="@dimen/help_text_size"
      android:textStyle="bold"></TextView>
<LinearLayout
      android:id="@+id/LinearLayout_Friend_Email"
      android:orientation="horizontal"
      android:layout_height="wrap_content"
      android:layout_width="fill_parent">
<Button
      android:id="@+id/Button_Friend_Email"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/settings_button_friend_email"></Button>
<TextView
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:textSize="@dimen/help_text_size"
      android:textStyle="bold"
      android:gravity="center"
      android:id="@+id/TextView_Friend_Email_Tip"
      android:text="@string/settings_friend_email_tip"></TextView>
</LinearLayout>
```

Как и в случае с другими настройками на этом экране, обновление макета подразумевает добавление нескольких элементов `TextView` и элемента `Button` с именем `Button_Friend_Email`. Нажатие на эту кнопку будет приводить к открытию нового диалогового окна. Следовательно, вы должны добавить новый ресурс макета, описывающий макет этого диалогового окна (рис. 17.2).

Этот макет должен быть описан следующим образом в XML-файле макета с именем **/res/layout/friend_entry.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:id="@+id/root"
     android:orientation="vertical"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:background="@drawable/bkgrnd">
```

```
    <TextView
    android:id="@+id/TextView_Friend_Email"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="@dimen/help_text_size"
    android:textStyle="bold"
    android:text="@string/settings_friend_email"></TextView>
    <EditText
          android:id="@+id/EditText_Friend_Email"
          android:layout_height="wrap_content"
          android:maxLength="50"
          android:layout_width="fill_parent"
          android:maxLines="1"
          android:inputType="textEmailAddress"></EditText>
</LinearLayout>
```

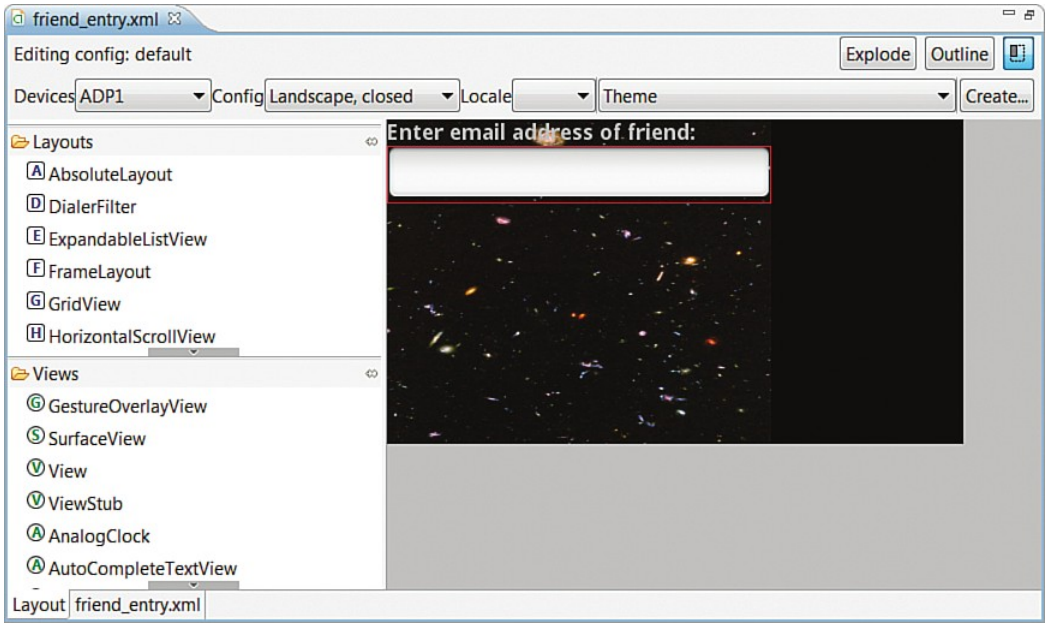

Рис. 17.2. Предварительный просмотр макета диалогового окна, предназначенного для формирования запроса на добавление пользователя в круг друзей

Содержимое пакета довольно простое. Макет представляет собой элемент- контейнер `LinearLayout` с двумя элементами: элементом `TextView`, который содержит текст, предлагающий пользователю ввести адрес электронной почты, и элемент `EditText`, позволяющий пользователю указать адрес электронной почты.

ОТКРЫТИЕ ДИАЛОГОВОГО ОКНА, ПРЕДНАЗНАЧЕННОГО ДЛЯ ДОБАВЛЕНИЯ ПОЛЬЗОВАТЕЛЯ В КРУГ ДРУЗЕЙ

Нажатие на элемент `Button` с именем `Button_Friend_Email` приводит к открытию диалогового окна, позволяющего пользователю ввести адрес электронной почты своего друга. Открытие этого диалогового окна во многом похоже на открытие других диалоговых окон, доступных на экране с настройками:

```
Button addFriend = (Button) findViewById(R.id.Button_Friend_Email);
addFriend.setOnClickListener(new View.OnClickListener() {
   public void onClick(View v) {
       showDialog(FRIEND_EMAIL_DIALOG_ID);
```

```
    }
});
```

Вам нужно обновить метод `onCreateDialog()` класса `QuizSettingsActivity`, чтобы добавить инструкцию case для этого нового диалогового окна:

```
case FRIEND_EMAIL_DIALOG_ID:
    final View friendDialogLayout = layoutInflater.inflate(
        R.layout.friend_entry, (ViewGroup)
        findViewById(R.id.root));

    AlertDialog.Builder friendDialogBuilder =
       new AlertDialog.Builder(this);
    friendDialogBuilder.setView(friendDialogLayout);
    final TextView emailText = (TextView)
       friendDialogLayout.findViewById(R.id.EditText_Friend_Email
       );

    friendDialogBuilder.setPositiveButton(
    android.R.string.ok, new DialogInterface.OnClickListener() {

    public void onClick(DialogInterface dialog, int which) {

        String friendEmail = emailText.getText().toString();
        if (friendEmail != null && friendEmail.length() > 0) {
            doFriendRequest(friendEmail);
        }
    }
});
return friendDialogBuilder.create();
```

Данная реализация диалогового окна должна быть вам уже достаточно знакома. Как и раньше, вы создаете диалоговое окно **AlertDialog**, применяя к нему ресурс макета. Когда пользователь щелкает по кнопке **0К** в этом диалоговом окне (рис. 17.3), указанный адрес электронной почты передается на сервер приложения в асинхронном режиме при помощи класса `FriendRequestTask`. (Мы поговорим об этом классе в следующем разделе.) Сервер приложения отвечает за создание дружественного отношения, если указанный пользователем адрес электронной почты друга присутствует в базе данных.

Рис. 17.3. Диалоговое окно, предназначенное для добавления пользователя в круг друзей

СОЗДАНИЕ КЛАССА ASYNCTASK, ПРЕДНАЗНАЧЕННОГО ДЛЯ ОТПРАВКИ ЗАПРОСОВ НА ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЕЙ В КРУГ ДРУЗЕЙ

Для отправки на сервер приложения запросов на добавление пользователей вкруг друзей вам потребуется добавить еще один класс `FriendRequestTask`, производный от класса `AsyncTask`, в класс `QuizSettingsActivity`. Класс `FriendRequestTask` может выполняться в результате вызова метода `execute()` каждый раз, когда игрок вводит адрес электронной почты своего друга в соответствующем диалоговом окне и нажимает на кнопку **ОК**.

Реализация производного класса `FriendRequestTask` во многом похожа на другие производные классы, унаследованные от класса `AsyncTask`, над которыми вы работали на протяжении последних нескольких часов. (В следующем примере для простоты понимания и краткости был удален код. обрабатывающий исключения.) Единственный действительно интересный момент в этой новой задаче — код метода `doInBackground()`, формирующий запрос на добавление пользователя в круг друзей:

```
@Override
protected Boolean doInBackground(String... params) {
   Boolean succeeded = false;
   String friendEmail = params[0];
   SharedPreferences prefs =
       getSharedPreferences(GAME_PREFERENCES, Context.MODE_PRIVATE);
   Integer playerId = prefs.getInt(GAME_PREFERENCES_PLAYER_ID, -1);

   Vector<NameValuePair> vars = new Vector<NameValuePair>();
```

```
    vars.add(new BasicNameValuePair("command", "add"));
    vars.add(new BasicNameValuePair("playerId", playerId.toString()));
    vars.add(new BasicNameValuePair("friend", friendEmail));
    HttpClient client = new DefaultHttpClient();
    HttpPost request = new HttpPost(TRIVIA_SERVER_FRIEND_EDIT);
    request.setEntity(new UrlEncodedFormEntity(vars));
    ResponseHandler<String> responseHandler = new
    BasicResponseHandler();
    String responseBody = client.execute(request, responseHandler);
    if (responseBody != null) {
        succeeded = true;
    }
    return succeeded;
}
```

В этом примере для передачи POST-запроса протокола HTTP, содержащего переменные формы, на сервер приложении вы используете объект типа `HttpClient`. Вы можете сформировать объект запроса типа `HttpPost` путем объединения соответствующего адреса URL сервера приложения с переменными запрос;», закодированными с использованием удобного класса `UrlEncodedFormEntity`, установив экземпляр этою класса в качестве содержимого POST-запроса. Когда запрос будет полностью сформирован, вы можете отправить его на сервер при помощи метода `execute()` класса `HttpClient`, как вы делали это в предыдущих примерах. После этого вы просто проверяете ответ, полученный от сервера, чтобы определить успешность выполнения запроса на добавление пользователя в круг друзей, путем анализа результирующего объекта типа `ResponseHandler`.

Отображение результатов игры друзей

Теперь, когда игроки могут формировать круг своих друзей, вам нужно обновить класс `QuizScoresActivity` для заполнения вкладки **Scores of Friends** (Результаты друзей) оперативными данными, получаемыми с сервера приложения. К счастью, в реализации этой функциональности нет ничего сложного, поскольку поддержку связей между друзьями в основном осуществляет сервер приложения. Для получения результатов игры друзей требуется просто немного измененный запрос к базе данных сервера приложения

Отображение результатов игры друзей на экране с результатами точно так же, как отображение лучших результатов игры, за одним исключением— адрес URL сервера приложения для запроса данных содержит идентификатор игрока. В этом случае сервер приложения знает, что вы хотите получить результаты только тех игроков, которые имеют отношение к данному игроку.

С точки зрения реализации, вы можете просто создать еще один экземпляр класса `ScoreDownloaderTask` для получения этих результатов и их отображения на вкладке **Scores of Friends** (Результаты друзей) (рис. 17.4). Когда экран с результатами игры будет использовать оперативные данные, получаемые с сервера приложения, вы можете удалить из проекта тестовые XML-файлы ресурсов и весь связанный с ними код.

Рис.17.4. Вкладка **Scores of Friends**
(Результаты друзей)

Расширение отношений между игроками

Возможность установления дружественных отношений может существенно повысить интерес пользователей к вашей игре помимо того функционала, который был реализован вами до сих пор. Добавление поддержки круга друзей может показаться очень примитивной социальной возможностью, но вы только представьте, как можно развить социальные функции вашего приложения на базе этой простой отправной точки. Отношения между игроками оставляют разработчикам широкое поле для развития приложений в разнообразных направлениях, включая следующие:

- Сервер приложения мог бы отправлять приглашение по электронной почте любому другу, который пока отсутствует в базе данных.

- Игроки не обязательно должны быть привязаны к платформе Android. Можно легко добавить другие платформы (веб-сайт, iPhone, BlackBertv и т. д.). Это значит, что друзья могли бы взаимодействовать с одним и тем же сервером приложения и играть друг с другом на различных платформах.

- Дружественные отношения можно было бы сделать как односторонними, так и двухсторонними (это значит, что результаты мот отображаться в списке у одного игрока или у каждого из двух игроков соответственно). Можно было бы устанавливать различные доверительные отношения, позволяя игрокам получать доступ к разнообразной информации о других игроках, включая ответы друзей на вопросы и их избранные места в мире.

- После установления дружественных отношений могут быть активизированы дополнительные функции приложения, включая поединки, обмен сообщениями, уведомления... возможностям нет предела. Используйте ваше воображение.

Полнофункциональная реализация возможности создания круга друзей, описанная в данном часе, может показаться вам незавершенной и это действительно так! Любое приложение, предоставляющее похожие функции по созданию круга друзей, по крайней мере, должно позволять игроку управлять (например, просматривать, удалять) его существующими дружественными отношениями. Тем не менее реализация этих улучшений предоставляется читателю в качестве упражнения.

ИНТЕГРАЦИЯ С СОЦИАЛЬНЫМИ СЕТЯМИ

В последние несколько лет социальные сети обрели неимоверную популярность, позволяя людям находить друг друга, поддерживать связь и делиться информацией (в радости и в горе) о своей жизни. Многие сайты социальных сетей предоставляют сторонним разработчикам интерфейсы API, многие из которых представляют собой веб-сервисы, построенные на базе архитектуры REST (Representational State Transfer). Для социальных сетей, например, сети Facebook, появилось огромное количество приложений.

Android-приложения могут интегрироваться с сайтами социальных сетей в соответствии с планами развития этих сайтов и с использованием интерфейсов API, предоставляемых конкретными сайтами или сервисами. Уровень интеграции может быть как поверхностным, так и полным. Вот несколько примеров интеграции с социальными сетями, которые могут оказаться актуальными в Android-приложении:

- предоставление пользователю возможности автоматически опубликовать твит на сайте Twitter, когда он победит в игре;

- разработка приложения, которое позволит пользователю просматривать и обновлять свой персональный блог, ленту на сайте Twitter и статус на сайте Facebook;

- разработка полнофункционального Twitter-клиента.

В каждом из перечисленных случаев интеграция возможностей сайта Twitter в Android-приложение осуществляется различными способами. Теперь давайте рассмотрим реализацию поддержки для некоторых социальных сетей, популярных в наши дни.

Добавление поддержки сети Facebook

Сеть Facebook — это популярный социальный веб-сервис, где люди могут находить друг друга, обмениваться изображениями и видеоклипами, и общаться. По адресу **developers.facebook.com** доступен портал для разработчиков, желающих интегрировать функциональность сети Facebook в сторонние приложения. Дополнительную информацию но платформе Facebook Platform for Mobile (Facebook Connect, Facebook SMS и т.д.) можно найти по адресу **wiki.developers.facebook.com/index.php/Mobile**.

Добавление поддержки сети Twitter

Сеть Twitter — это популярная социальная сеть, где люди обмениваются короткими текстовыми сообщениями, называемыми твитами. Каждый твит может содержать не более, чем 140 символов, что делает сеть Twitter идеальной платформой для мобильных

устройств. Сеть Twitter имеет портал для разработчиков, доступный по адресу **apiwiki.twitter.com**, со справочной информацией по интерфейсу Twitter API.

Работа с платформой OpenSocial

Если вы хотите реализовать поддержку более чем одного сайта социальной сети или охватить максимально возможное число конечных пользователей, вам следует взглянуть на интерфейсы API платформы OpenSocial, дополнительная информация по которой доступна по адресу wiki.opensocial.org. В платформе OpenSocial используются общепринятые интерфейсы API (вместо интерфейсов API, разработанных для конкретных сайтов), которые позволяют интегрировать в приложения возможности многих популярных социальных приложений и сетей, включая следующие (перечисленные в алфавитном порядке):

- friendster (по-прежнему популярный сервис в Юго-Восточной Азии);
- hi5 (популярный сервис в Европе и Центральной и Южной Америке);
- Hyves (популярный сервис в Нидерландах);
- Linkedln (социальная сеть для поиска и установления деловых контактов);
- Mail.ru (популярный сервис в России);
- mixi (популярный сервис в Японии);
- MySpace (популярный сервис в Соединенных Штатах и по всему миру);
- NetLog (популярный сервис в Европе и на Ближнем Востоке);
- orkut (популярный сервис в Южной Америке и Индии);
- RenRen (прежнее название — Xiaonei, популярный сервис среди студентов Китая);
- Yahoo! (популярный сервис в Соединенных Штатах и по всему миру);
- XING (социальная сеть для поиска и установления деловых контактов, популярная в Европе и Китае).

Ежедневное и ежемесячное количество активных пользователей в каждой ш них социальных сетей исчисляется миллионами.

ИТОГИ

В этом часе вы у знати, как можно использовать социальные возможности для улучшения восприятия мобильного приложения пользователем. Вы рассмотрели небольшой пример, демонстрирующий добавление социальных возможностей в приложение «Been There, Done That!», которые позволяют пользователю указывать своих друзей (используя адреса

электронной почты) и просматривать набранные ими очки. Наконец, вы узнали о многих социальных сетях, с которыми можно интегрировать ваше приложение.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Как выбрать наилучший уникальный идентификатор для проведения различий между пользователями?

Ответ: Несмотря на ряд предложений по реализации сервисов, поддерживающих технологию единого входа, до сих пор нет четкого ответа на данный вопрос. Некоторые из кандидатов на эту роль — уникальные пары «логин/ пароль», адреса электронной почты или номера телефонов. В примере, рассмотренном в этом часе, в качестве уникального идентификатора мы использовали адрес электронной почты игрока и позволяли пользователю указывать пароль. На многих сайтах социальных сетей применяется похожий механизм, однако этот подход не лишен недостатков — например, адреса электронной почты могут измениться, пользователи могут иметь более одной учетной записи, и они вынуждены запоминать еще одну комбинацию «логин/пароль». Когда вы интегрируете ваше приложение с вебсайтом одной из социальных сетей, вы должны использовать тот механизм аутентификации и те учетные данные, которые определены интерфейсом API данного сайта.

Вопрос: Какие моменты, связанные с конфиденциальными данными пользователей, я должен учитывать при разработке социальных приложений?

Ответ: Когда дело касается социальных приложений, вы должны всегда предупреждать пользователя о том, как вы собираетесь использовать любую информацию, получаемую от пользователя. Чтобы максимально обезопасить себя, вы должны следовать следующим принципам: не получать, не использовать и не сохранять никакую информацию, которая не требуется вашему приложению, а также всегда предполагать, что любая информация, предоставляемая пользователем, конфиденциальна. Теперь, в соответствии с данными принципами, даже простейшая поддержка круга, которую вы добавили в приложение «Been There, Done That!», подразумевал передачу конфиденциальных данных: ника пользователя, его результаты игры и аватар. (Среди упражнений, представленных в конце этого часа, есть упражнение, предлагающее реализовать функционал для загрузки изображений аватар друзей с сервера.) С технической точки зрения, если вы публикуете данное приложение, вы должны ясно дать понять игроку, что эта информация будет выгружена на сервер приложения и станет доступна другим игрокам

Вопрос: Как определить, могу ли я интегрировать мое приложение с сервисом социальной сети, не описанной в этом часе?
Ответ: Если вы желаете интегрировать ваше приложение с сервисом какой- либо социальной сети или любым другим веб-сервисом (например, с сервисами сайтов Google, Amazon, eBay), простейший способ выяснить, имеет ли сервис интерфейс API для интеграции, — посетить веб-сайт компании. Зачастую на сайте рядом с информацией о поддержке пользователей, контактной информацией и информацией о компании или в разделе часто задаваемых вопросов, связанных с поддержкой пользователей, присутствует ссылка для разработчиков. Большинство компаний требуют, чтобы разработчики приняли условия использования их сервисов, а некоторые компании требуют зарегистрироваться на сайте, чтобы получить специальный API-ключ для использования сервисов.

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Все Android-приложения могут и должны быть улучшены с использованием социальных возможностей.

2. Каким образом приложение «Been There, Done That!» создает дружественные отношения?

A. Предоставляя игроку возможность поиска друзей в списке на сервере приложения.
B. Предоставляя игроку возможность вводить адрес электронной почты своего друга.
C. Запуская приложение для работы с контактами и позволяя игроку выбрать желаемый контакт.
D. Предоставляя игроку возможность ввести номер телефона своего друга.

3. Верно ли это? Инструментарий Android SDK имеет встроенную поддержку сервисов таких социальных сетей, как Facebook, Twitter и MySpace.

Ответы

1. Неверно. Добавление социальных возможностей и приложение позволяет улучшить восприятие этого приложения пользователями, однако данное архитектурное решение требует взвешивания и планирования. Некоторые типы приложения получают очевидную пользу от реализации этих возможностей, при этом для других типов приложений подобные возможности могут не иметь смысла. Добавляйте социальные возможности в приложение только тогда, когда в результате их внедрения очевидные преимущества получат как пользователи, так и разработчик.

2. B. В приложении «Been There, Done That!» игроки могут добавлять друзей путем указания их адресов электронной почты.

3. Неверно. Вы можете использовать сетевые возможности, предоставляемые инструментарием Android SDK, для доступа к интерфейсам API, нредостааняемым сайгами таких социальных сетей, как Facebook, Twitter и MySpace.

Упражнения

1. Измените вкладку **Scores of Friends** (Результаты друзей) на экране с результатами игры таким образом, чтобы вместе с результатами игры друзей отображались их аватары. (Подсказка: адрес URL изображения аватара каждого друга игрока включен в XML-данные с результатами игры, загружаемые с сервера приложения.)

2. Измените экран с результатами игры, добавив на него еще одну вкладку, на которой будут отображаться набранные очки игроков, включивших этою игрока в круг своих друзей (другими словами, игроков, которые наблюдают за результатами данного игрока). На сервере приложения соответствующий функционал уже реализован.

Используйте тот же самый адрес URL, лишь добавив в строку запроса переменную followers со значением true (например, «**http://tqs.nnamlambo.com/scores.jsp7playerIds##&followers=true**»).

3. Изучите документацию по интерфейсу API сервиса любой понравившейся вам социальной сети. Попробуйте реализовать простую функциональность в приложении «Been There, Done That!», которая бы позволяла обращаться к данному сетевому сервису оригинальным образом. Например, вы могли бы публиковать твит в ленте игрока на сайте Twitter каждый раз, когда этот игрок утвердительно отвечает на очередной вопрос викторины (например, «Игрок X совершил восхождение на гору Эверест!»).

4. Добавьте функционал, позволяющий отправлять текстовое сообщение другу ифока с предложением побить установленный им рекорд, т. е. бросить вызов другому игроку.

Час 18. СОЗДАНИЕ ВИДЖЕТА ДЛЯ ДОМАШНЕГО ЭКРАНА

Вопросы, рассматриваемые в этом часе:

- **подготовка дизайна и разработка виджета;**
- **обработка событий, генерируемых виджетом в результате действий пользователя;**
- **работа со службами.**

В этом часе вы создадите виджет для приложения «Been There, Done That!». В частности, вы создадите простой виджет в виде элемента управления, который можно добавлять на домашний экран пользователя и который будет отображать аватар пользователя, его ник и информацию о набранных очках, а также предоставлять пользователю возможность продолжить игру.

ПОДГОТОВКА ДИЗАЙНА ВИДЖЕТА

Инструментарий Android SDK предоставляет разработчикам интересную возможность реализации функциональности, выходящей за рамки традиционных приложений для мобильных устройств, — с использованием виджетов. Разработчики могут использовать интерфейс API виджетов для создания маленьких элементов управления или представлений, которые могут быть добавлены на домашний экран пользователя. Эти простые элементы управления могут предоставлять пользователю информацию о приложении и напоминать ему в нужный момент о необходимости запустить приложение.

Виджеты могут оказаться полезными для определенных типов приложений, например, для тех приложений, которые должны сообщать пользователю о некотором статусе или обновлении информации. Приложение прогноза погоды может иметь виджет, который отображает текущие погодные условия для заданного местоположения на домашнем экране. Приложение, управляющее задачами, может иметь виджет, который информирует пользователя о следующей задаче в его списке задач или о том, какое количество задач осталось выполнить на сегодня. Приложение для работы с галереей изображений может иметь виджет, который используется для отображения слайд- шоу, включающего все изображения из галереи.

В этом часе вы создадите простой виджет для приложения «Been There, Done That!». Этот виджет будет выполнять следующее:

- отображать аватар пользователя;
- отображать ник пользователя;
- отображать текущий результат игры пользователя;
- запускать приложение «Been There, Done That!», когда пользователь нажмет на виджет.

Описание свойств виджета

Описание виджета и свойства его конфигурации задаются в отдельном XML-файле, ссылка на который добавляется в файл манифеста Android. (Подробнее об этом чуть позже.)

Вот некоторые распространенные свойства, используемые для описания виджета:

- **Размер.** Ширина и высота виджета, выражаемая в независимых от разрешения пикселах (dp или dip), которые применяются для определения количества ячеек таблицы на домашнем экране, необходимых для правильного отображения виджета.

ЗНАЕТЕ ЛИ ВЫ, ЧТО

Домашний экран операционной системы Android организован в виде ячеек таблицы, которые обычна представляют собой квадрат с размерами 74x74 пикселов. В каждой ячейке может размещаться только один элемент, например виджет или ярлык приложения. Таким образом, элементы на домашнем экране не перекрывют друг друга.

- **Частота обновлений.** Интервал (в миллисекундах) между системными вызовами провайдера виджета для обновления содержимого виджета.

- **Исходный макет.** Файл макета, используемый при первоначальном добавлении виджета. В дальнейшем этот макет может быть изменен программным путем.

- **Деятельность конфигурации.** Описание запускаемой деятельности для настройки различных аспектов виджета перед его первоначальным отображением.

Чтобы добавить описание виджета для данного примера, начните с добавления нового XML-файла с именем **widget_info.xml** в папку **/res/xml**. Поместите в этот файл следующее описание виджета:

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:minWidth="146dp"
      android:minHeight="146dp"
      android:updatePeriodMillis="10800000"
      android:initialLayout="@layout/widget">
</appwidget-provider>
```

В этом файле определяется виджет, который будет обновляться каждые три часа и занимать две ячейки таблицы по вертикали и две ячейки таблицы по горизонтали. Если провести несложные вычисления, вы, возможно, заметите, что значение 146dp, указанное для каждого измерения виджета, нельзя получить путем умножения размера ячейки таблицы (74dp) на два. И хотя размер ячейки таблицы обычно равен значению 74dp, при вычислении размеров вы должны вычитать значение 2dp из итогового результата. В данном примере мы умножили 74 на 2 и получили 148. Затем мы вычли 2 из произведения и получили число 146, которое и указали в файле. Если бы мы не сделали этого, для отображения виджета могло использоваться большее число ячеек таблицы.

Обновление этого виджета будет происходить каждые 10.800.000 миллисекунд, что соответствует трем часам. Кроме всего прочего, этот виджет изначально будет использовать предопределенный макет, который указан в строке `android: initialLayout="@layout/widget"`. Содержимое этого файла макета будет рассмотрено чуть позже.

Обновление файла манифеста Android

Вы должны обновить файл манифеста Android, чтобы сообщить системе, где находится описание виджета. Виджет представляет собой особую реализацию элемента `BroadcastReceiver`.

Следовательно, в файл **AndroidManifest.xml** должен быть добавлен раздел `<receiver>`, определяющий получаемые интенты и некоторую другую информацию, относящуюся непосредственно к виджету.
Чтобы решить данную задачу, добавьте следующий раздел `<receiver>` в раздел `<application>` файла **AndroidManlfastxml**:

```
<receiver
    android:name=”QuizWidgetProvider”>
    <intent-filter>
        <action
           android:name=”android.appwidget.action.APPWIDGET_UPDATE” />
    </intent-filter>
    <meta-data
        android:name=”android.appwidget.provider”
        android:resource=”@xml/widget_info” />
</receiver>
```

Представленный раздел <receiver> файла манифеста Android определяет фильтр для интентов, используемых для обновления виджета. Кроме того, он также связывает виджет и его файл описания с основным приложением.

Разработка дизайна макета виджета

К макетам виджетов предъявляются специфические требования. Начнем с того, что отображение виджета осуществляется через интерфейс `RemoteViews`, накладывающий ограничения на элементы пользовательского интерфейса, которые могут отображаться на экране. Далее, размеры содержимого виджета должны соответствовать размерам, указанным в свойствах данного виджета.

Объект типа `RemoteViews` используется в тех случаях, когда отображение элемента-представления будет осуществляться из другого процесса. Это именно то, что происходит с виджетом: он отображается из хост-процесса виджетов, а не из основного процесса приложения. В объектах типа `RemoteViews` допускается использование элементов-контейнеров и элементов-представлений определенных типов. Ниже перечислены некоторые элементы-контейнеры и элементы-представления, поддерживаемые виджетами:

- `LinearLayout`;
- `FrameLayout`;

- RelativeLayout;
- TextView;
- ImageView;
- Button;
- ImageButton;
- ProgressBar;
- AnalogClock;
- Chronometer.

Использование классов, расширяющих эти элементы макт, не допускается. Это значит, что дизайнер в своих действиях крайне ограничен Но, несмотря на то, что виджеты не предназначены для отображения большого количества информации, когда требуются более мощная функциональность или сложные макеты экранов, вы можете расширить возможности виджета путем запуска с его помощью полноценной деятельности. Деятельность может быть использована для отображения целого экрана или просто всплывающего экрана. В любом случае, на деятельности, запускаемые при помощи виджета, больше не распространяются ограничения, налагаемые на сам виджет и обусловленные использованием объекта типа RemoteViews.

Чтобы подготовить макет виджета, создайте новый файл макета с именем **widget.xml** и поместите в него следующий код:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/widget_view">
<ImageView
    android:layout_centerInParent="true"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:id="@+id/widget_image"></ImageView>
<TextView
    android:text="@+id/TextView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:id="@+id/widget_nickname"></TextView>
<TextView
    android:text="@+id/TextView02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_alignParentBottom="true"
    android:id="@+id/widget_score"></TextView>
</RelativeLayout>
```

Реализация виджет-провайдера

Теперь, когда у вас есть конфигурация виджета, вам нужно реализовать сам виджет. Для этого необходимо расширить класс AppWidgetProvider, содержащий пять методов, которые могут быть переопределены:

- `onUpdate()` — этот метод вызывается по истечении интервала времени, указанного в свойствах виджета.

- `onDeleted()` — этот метод вызывается каждый раз, когда происходит удаление виджета.

- `onEnabied()` — этот метод вызывается при создании первого экземпляра виджета, но не вызывается при создании последующих экземпляров.

- `onDisabied()` — этот метод вызывается в том случае, когда удаляется последний экземпляр данного виджета.

- `onReceive()` — этот метод вызывается для всех получаемых широковещательных событий; в стандартной реализации данного метода вызываются все вышеперечисленные методы обработки событий (методы `onUpdate()`, `onDeleted()`, `onEnabled()` и `onDisabied()`). Этот метод может быть переопределен, когда требуется реализовать нестандартное поведение.

В этом примере необходимо переопределить только метод `onUpdate()` для обновления содержимого объекта типа `RemoteViews`. Поскольку поддержка автономной установки виджета не требуется, переопределять другие методы нет никакой необходимости. Итак, ниже представлена реализация класса, расширяющего класс `AppWidgetProvider`:

```
public class QuizWidgetProvider2 extends AppWidgetProvider {
   @Override
   public void onUpdate(Context context,
          AppWidgetManager appWidgetManager,
          int[] appWidgetIds) {
      WidgetData widgetData = new WidgetData("Unknown", "NA", "");
      getWidgetData(widgetData);
      String packageName = context.getPackageName();
      RemoteViews remoteView =
            new RemoteViews(context.getPackageName(), R.layout.widget);
      remoteView.setTextViewText(
            R.id.widget_nickname, widgetData.nickname);
      remoteView.setTextViewText(
            R.id.widget_score, "Score: " + widgetData.score);
      if (widgetData.avatarUrl.length() > 0) {
         URL image;
         try {
         image = new URL(widgetData.avatarUrl);

         Bitmap bitmap =
                BitmapFactory.decodeStream(image.openStream());
         if (bitmap == null) {
            Log.w(DEBUG_TAG, "Failed to decode image");
            remoteView.setImageViewResource(
                R.id.widget_image, R.drawable.avatar);
         } else {
```

```
            remoteView.setImageViewBitmap(
                    R.id.widget_image, bitmap);
            }
            } catch (MalformedURLException e) {
                Log.e(DEBUG_TAG, "Bad url in image", e);
            } catch (IOException e) {
                Log.e(DEBUG_TAG, "IO failure for image", e);
            }
} else {
    remoteView.setImageViewResource(
        R.id.widget_image, R.drawable.avatar);
}

try {
    ComponentName quizWidget =
            new ComponentName(context, QuizWidgetProvider.class);
    AppWidgetManager appWidgetManager =
            AppWidgetManager.getInstance(context);
    appWidgetManager.updateAppWidget(quizWidget, remoteView);
} catch (Exception e) {
    Log.e(DEBUG_TAG, "Failed to update widget", e);
}
}

private void getWidgetData(WidgetData widgetData) {
}
}
```

Сначала вам нужно получить данные, которые будут отображаться в виджете. Эту задачу выполняет метод `getWidgetData()`, получая идентификатор игрока из экземпляра класса SharedPreferences и затем загружая данные с сервера. Загрузка данных осуществляется именно с сервера, а не непосредственно из экземпляра класса `SharedPreferences`, поскольку в дальнейшем такой подход позволит легко изменить метод `getWidgetData()` для загрузки данных друзей или других игроков.

После этого вы создаете объект типа `RemoteViews`. При инициализации данного объекта требуется указать не только используемый макет, но и пакет, к которому принадлежит этот макет. Напомним, что объект типа `RemoteViews` не будет отображаться из того же процесса, в котором выполняется приложение, поэтому для доступа к ресурсам этому объекту требуется дополнительная информация.

У вас есть доступ к информации о нике пользователя и набранных им очках (или указанным значениям по умолчанию), поэтому вы можете поместить эту информацию в объект типа `RemoteViews` при помощи метода `setTextViewText()`, передав в него соответствующий идентификатор элемента макета и значение, которое должно отображаться в этом элементе. Код для передачи изображения аватара в объект типа `RemoteViews` отдаленно напоминает предыдущий код, однако сначала происходит проверка доступности указанного адреса URL для изображения аватара и что данное изображение может быть декодировано. Наконец, изображение аватара передается в объект типа `RemoteViews` путем вызова метода `setImageViewBitmap()` или метода `setImageViewResource()`.

После этого вы должны непосредственно обновить виджет. Для этого используется метод `updateAppWidget()` класса `AppWidgetManager`. В этот метод передается объект

типа `RemoteViews` и объект типа `ComponentName` для класса `QuizWidgetProvider`, получаемый путем непосредственного создания экземпляра класса `ComponentName`.

КСТАТИ

Если вы внимательно изучили представленный код, возможно, вы заметили, что два параметра метода `onUpdate()` — `appWidgetManager` и `appWidgetIds` - не используются. Параметр `appWidgetManager` не используется потому, что вы скоро переместите код в другой метод, где экземпляр класса `AppWidgetManager` будет получаться отдельно. Параметр `appWidgetIds` используется в тех случаях, когда необходимо поддерживать несколько уникальных экземпляров виджета, которые отображают разные данные. В данном случае приложение должно отслеживать отдельные значения параметра `appWidgetIds`, которые генерируются системой, и сопоставлять их с данными, которые должны отображаться в каждом экземпляре виджета. Обычно это осуществляется путем использования отдельной конфигурационной деятельности для каждого экземпляра виджета, благодаря чему пользователь может управлять теми данными, которые должны отображаться в каждом отдельном экземпляре этого виджета.

Теперь созданный виджет должен появиться в списке виджетов домашнего экрана, как показано на рис. 18.1.

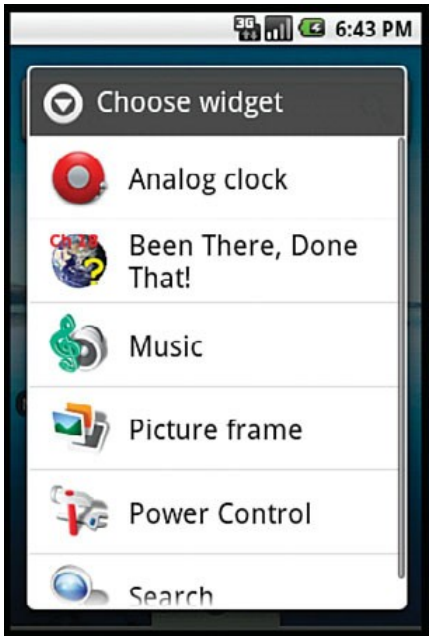

Рис. 18.1. Добавление виджета на домашний экран

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО

Чтобы добавить виджет на домашний экран телефона работающего под управлением операционной системы Android или эмуляторе, выполните следующие простые шаги:

1. Откройте домашний экран.
2. Найдите подходящую область на экране. (Помните что для виджета приложения «Been There, Done That!» требуются две ячейки по горизонтали и две - по вертикали).
3. Нажмите и удерживайте ваш палец (или щелкните и удерживайте кнопку мыши при использовании эмулятора).
4. Когда на экране появится меню **Add to Home Screen** (Добавить на главный экран), выберите пункт **Widgets** (Виджеты).
5. В открывшемся меню **Choose widget** (Выбор виджета) выберите виджет, который вы только что создали (или любой другой виджет), чтобы добавить его на ваш домашний экран.

Виджет будет выглядеть аналогично тому, который изображен на рис 18.2.

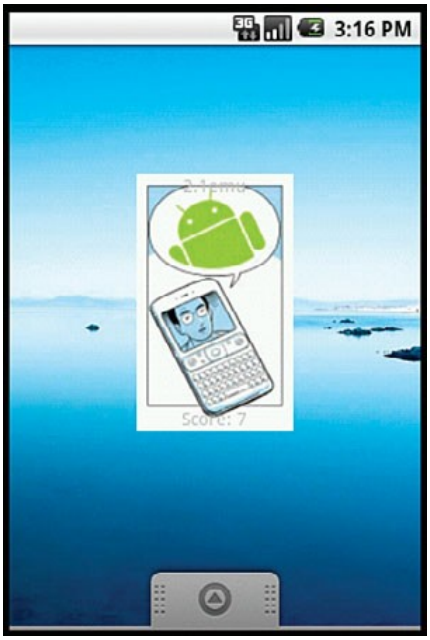

Рис. 18.2. Виджет приложения «Been There, Done That!»

ОБРАБОТКА СОБЫТИЙ, ГЕНЕРИРУЕМЫХ ВИДЖЕТОМ В РЕЗУЛЬТАТЕ ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЯ

На данном этапе созданный вами виджет работает, однако он представляет весьма скромные интерактивные возможности. Вы хотите, чтобы он не просто отображал информацию для пользователя, - вы хотите чтобы он также возвращал пользователя в приложение. Еще одна возможная область для улучшения виджета – изящная обработка ситуаций, связанных с медленной загрузкой изображений аватаров.

Виджеты отображаются через объекты типа `RemoteView`, а не и приложения, в котором они создаются. Отображение виджетов осуществляется из виджет-хоста. Это обстоятельство влияет на способ обработки событий, связанных с действиями

пользователя. Если вернуться к списку элементов представлений, поддерживаемых виджетом, вы увидите, что в этом списке отсутствуют поля ввода. По существу, единственное событие, поддерживаемое виджетом, - это нажатие.

КСТАТИ

Возможно, вы встречали виджеты, например, виджет сети Facebook, которые выглядят как поля ввода `EditText`. Тем не менее, если вы нажмете на них, чтобы ввести текст вы увидите, что во всех случаях для ввода используется другой пользовательский интерфейс. Это отличный способ предоставить пользователю элементы с расширенной функциональностью, тем самым обойдя ограничение платформы разработчика виджетов.

Поскольку отображение виджета осуществляется не из этого процесса, в котором выполняется приложение, для получения событий нажатий требуется новый подход. Инструментарий Android SDK предоставляет для этих целей специальный тип интента под названием `PendingIntent`. Это интент, который еще только будет отправлен когда-то впоследствии и, возможно, из другого процесса. Чтобы создать интент типа `PendingIntent`, сначала необходимо создать экземпляр класса `Intent`. После этого создается экземпляр класса `PendingIntent` с некоторой дополнительной информацией, например с указанием того, что делать при последующем использовании того же интента. То есть может быть использован тот же экземпляр или создан новый экземпляр. Как только объект типа `PendingIntent` будет создан, его можно связать с объектом типа `RemoteView`, вызвав метод `setonClickPendingIntent()`. Вам нужно добавить следующий код перед вызовом метода `updateAppWidget()`.

```
Intent launchAppIntent =
    new Intent(context, QuizMenuActivity.class);
PendingIntent launchAppPendingIntent =
    PendingIntent.getActivity(context, 0, launchAppIntent,
    PendingIntent.FLAG_UPDATE_CURRENT);
remoteView.setOnClickPendingIntent(
    R.id.widget_view, launchAppPendingIntent);
```

Идентификатор элемента-представления, с которым связывается объект типа `PendingIntent` путем вызова метода `setClickpendingIntent()`, указывает на элемент-контейнер `RealativeLayout` из файла макета widget.xml. Теперь, когда пользователь нажмет на виджет, запустится деятельность `QuizmenuActivity`; именно эта деятельность запускается при открытии приложения «Been There, Done That!». (Обратите внимание, что по нажатию можно запустить любую деятельность, например, деятельность, отображающую результаты игры, однако в данном случае деятельность `QuizMenuActivity` имеет наибольший смысл.)

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

При создании динамического виджета с отдельными элементами-представления могут быть связаны свои собственные объекты типа `PendingIntent`. В этом случае отправляемые запросы приводят к вызову метода `updateAppWidget()`, позволяя изменить по нажатию внешний вид виджета.

ВЫПОЛНЕНИЕ ФОНОВЫХ ОПЕРАЦИЙ В ВИДЖЕТЕ

Возможно, вы считаете, что, поскольку виджет не выполняется в процессе приложения, вам не нужно беспокоиться об операциях, которые могут занимать достаточно длительное время. Также, возможно, вы предполагаете, что. поскольку виджет представляет собой объект типа `BroadcastReceiver`, он будет автоматически выполнять все операции в фоновом режиме. Оба предположения неверны.

Для операций, которые могут занимать продолжительное время, обычно запускается отдельный поток, в котором выполняются необходимые задачи. Тем не менее в случае с виджетами запустить отдельный поток не представляется возможным. Вместо этого, вы должны создать и запустить экземпляр класса `Service`, представляющий службу в операционной системе Android, и уже из этой службы можно запустить фоновый поток. Служба в операционной системе Android предоставляет две полезных возможности: она позволяет осуществлять выполнение задач в фоновом режиме, и она может предоставлять интерфейс к удаленному объекту, например доступ к расширениям или к программным библиотекам.

КСТАТИ

Почему нельзя использовать класс Thread непосредственно в виджете? И, хотя подробное рассмотрение этой особенности выходит за рамки данной книги, попробуем ответить коротко: когда завершает выполнение метода `onUpdate()`, процесс виджета может также завершится, попутно уничтожив все выполняющиеся потоки. В отличии от этого, экземпляр класса `Service`, представляющий службу в операционной системе Android, удален не будет, и поэтому вы можете использовать его для выполнения фоновых операций.

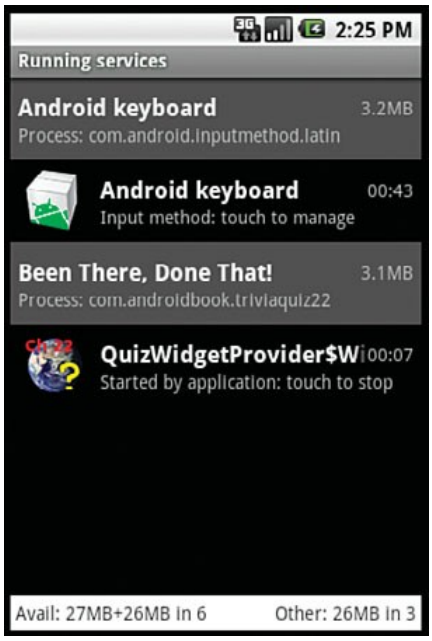

Рис. 18.3. Экран **Running Services**
(Работающие программы)

Вы можете увидеть все службы, выполняющиеся в данный момент на мобильном устройстве Android или на эмуляторе, выбрав приложение **Settings** (Настройки) в списке приложений, а затем выбрав в меню команд) **Applications** (Приложения) и затем команду **Running Services** (Работающие программы). Здесь вы также можете завершить выполнение выбранной службы. На рис. 18.3 изображен экран **Running Services** (Работающие программы).

Создание службы

Объект типа Service создается путем расширения класса `Service`, переопределением трех основных методов и описанием соответствующей службы в файле **AndroidManifest.xml**. Для приложения «Been There, Done That!-» вы должны определить объект типа `Service` в классе `QuizWidgetProvider`. Затем вы можете перенести большую часть кода из метода `onUpdate()` в метод `onStartCommand()` объекта типа Service и создать объект типа `Thread`, который будет выполнять всю работу, чтобы виджет и служба могли продолжать реагировать на действия пользователя.

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Жизненный цикл экземпляра класса `Service` отличается от жизненного цикла экземпляра класса `Activity`. Поскольку предоставленный пример был упрощен, вы не увидели полный жизненный цикл службы. Тем не менее, в сущности, после вызова метода `onCreate()` происходит вызов либо метода `onStartCommand()`, либо метода `onBind()`, в зависимости от типа службы и способа ее запуска. Когда служба завершает свою работу в нормальном режиме или когда не остается процессов, связанных со службой, вызывается метод `onDestroy()`.

Добавьте следующий код в класс `QuizWidgetProvider`:

```
public static class WidgetUpdateService extends Service {
    Thread widgetUpdateThread = null;
    private static final String DEBUG_TAG = "WidgetUpdateService";

    @Override
    public int onStartCommand(Intent intent, int flags, final int
    startId) {
        widgetUpdateThread = new Thread() {
        public void run() {
            // code moved from onUpdate() method
        }
    };
        widgetUpdateThread.start();
        return START_REDELIVER_INTENT;
    }

    @Override
    public void onDestroy() {
        widgetUpdateThread.interrupt();
```

```
        super.onDestroy();
    }

    @Override
    public IBinder onBind(Intent intent) {
        // no binding; can't from an App Widget
        return null;
    }
}
```

Так выглядела бы реализация службы, если бы код из метода `onUpdate()` был перемещен в метод `run()` потока, созданного в методе `onStartCommand()`.

КСТАТИ

Полная реализация виджета доступна на диске, прилагаемом к данной книге.

Теперь необходимо обновить файл манифеста Android, чтобы система знала о данной службе. Для этого добавьте следующий блок `<service>` раздел `<application>` файла манифеста:

```
<service
    android:name="QuizWidgetProvider$WidgetUpdateService" />
```

Этот код сообщает системе о существовании службы и о том, где она находится.

Управление службой

Теперь вы должны запустить службу из метода `onUpdate()` виджета. Вы можете запустить службу одним из двух способов: либо путем вызова метода `Context.startService()`, либо путем вызова метода `Context.bindService()`. В данном примере используется метод `startService()` после замены метода `onUpdate()` следующим кодом:

```
@Override
public void onUpdate(Context context,
    AppWidgetManager appWidgetManager, int[] appWidgetIds) {
    Intent serviceIntent = new Intent(context,
    WidgetUpdateService.class);
    context.startService(serviceIntent);
}
```

Когда служба завершит выполнение поставленной задачи, желательно остановить эту службу, чтобы освободить ценные системные ресурсы. Виджет будет обновляться каждые три часа. И, хотя со службой можно ничего не делать — она будет бездействовать, — возможно, вы захотите остановить эту службу до тех пор, пока она не понадобится снова. Этот результат достигается путем добавления следующего кода в конец метода `run()` класса `Thread`, который вы используете для выполнения фоновых задач:

```
if (!WidgetUpdateService.this.stopSelfResult(startId)) {
```

```
    Log.e(DEBUG_TAG, "Failed to stop service");
}
```

Вызов метода `stopSelfResult()` отправляет службе сигнал об остановке и возвращает информацию о том, был ли вызов успешным или нет. Если результат вызова данного метода вас не интересует, вы можете просто вызвать метод `stopSelf()`. Вызов этого метода происходит в тот момент, когда обработка данных в потоке уже завершена, и больше нет смысла продолжать выполнение службы, поскольку в данном случае новых задач больше нет. Служба может быть запушена снова при последующем обновлении виджета, что для данного виджета произойдет по истечении трех часов.

Ест и виджет будет удален из своего хоста, например домашнего экрана, во время выполнения операции обновления, службу придется завершать другим способом. Для этого поместите следующий код метода `onDeleted()` в реализацию класса `AppWidgetProvider`:

```
@Override
public void onDeleted(Context context, int[] appWidgetIds) {
    Intent serviceIntent = new Intent(context,
    WidgetUpdateService.class);
    context.stopService(serviceIntent);
    super.onDeleted(context, appWidgetIds);
}
```

Вызов метода `StopService` о приводит к вызову метода `onDestroy()` класса `Service`, который, в свою очередь, пытается прервать выполнение
потока.

КСТАТИ

Пример, представленный в данном часе, достаточно прост. Все будет работать прекрасно для одного экземпляра виджета. Тем не менее, если вы захотите добавить поддержку нескольких экземпляров виджета, выполняющихся одновременно, вам понадобится реализовать код, который позволит проводить различия между экземплярами виджета.

ИТОГИ

В этом часе вы создали простой виджет для приложения «Been There, Done That!», который отображает аватар пользователя, его ник и набранные очки. В этом часе были рассмотрены все детали реализации виджета, включая разработку дизайна макета и описание свойств виджета. Вы также добавили простейшую обработку событий, позволяя пользователю нажать на виджет, чтобы запустить приложение «Been There, Done That!». Наконец, вы использовали фоновую службу для выполнения обработки событий виджета и обновления его содержимого.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Домашний экран — это единственное место, где можно размещать виджеты?

Ответ: Нет. Виджеты могут размещаться в любом виджет-хосте. Домашний экран — это просто то место, где виджеты используются чаще всего. Дополнительную информацию можно найти в документами по классам `AppWidgetHost` и `AppWidgetHostView`.

Вопрос: Как можно добавить в виджет дополнительные интерактивные возможности, например, элементы `Button`?

Ответ: Если вы хотите добавить элементы управления пользовательского интерфейса в виджет и предоставить пользователю возможность обновлять содержимое виджета, вам потребуется описать каждое событие по отдельности и реализовать соответствующие обработчики событий нажатий для отправки определенных команд, связанных с событиями, посредством объектов типа `PendingIntent` в зарегистрированный получатель объектов типа `Intent`. Затем приложение виджета должно получить эти команды и обработать их соответствующим образом, при необходимости обновив содержимое виджета. Вы можете найти полнофункциональный пример интерактивного виджета в статье «Handling User Interaction with Android App Widgets», которая доступна по адресу **www.developer.com/ws/article.php/3837531/Handling-User-Interaction-with-Android-App-Widgets.htm**.

Вопрос: Могу ли я создать несколько экземпляров виджета?

Ответ: Для виджета, который вы реализовали для приложения «Been There, Done That!», создание нескольких экземпляров не имеет смысла. Тем не менее в определенных ситуациях бывает полезно предоставить пользователю возможность создания нескольких экземпляров виджета с разными конфигурациями. Один из способов решить эту задачу — предоставить пользователю возможность настройки каждого экземпляра виджета при помощи конфигурационной деятельности, указанной для данного вцджета. После этого приложение должно отслеживать различия между созданными экземплярами, запоминая настройки, указанные пользователем, для каждого идентификатора экземпляра виджета. Мы также рассмотрели эту интересную тему в статье «Handling User Interaction with Android App Widgets» (дополнительную информацию можно найти в ответе на предыдущий вопрос).

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Виджеты могут размещаться только на домашнем экране.

2. Укажите пример элемента-представления, который не может быть использован в виджете:

A. Button.
B. WebView.
C. ProgressBar.

3. Верно ли это? Несмотря на то, что размеры виджетов определяются в пикселах, их размер непосредственно должен соответствовать определенному количеству ячеек.

4. С какой целью в виджетах применяются службы?

A. Для выполнения продолжительных фоновых операций.
B. Б. Дня выполнения рисования непосредственно на экране.
C. Для доступа к конфиденциальным данным.

Ответы

3. Неверно. Виджеты могут размешаться в любом приложении, в котором используется объект типа `AppWidgetHost`.

4. В. Элементы `Button` и `ProgressBar` могут использоваться, а элемент- представление `WebView` — нет.

5. Верно. Размеры каждой ячейки составляют 74 пикселов по горизонтали и по вертикали, но если вы планируете размещать виджет в нескольких ячейках, от полученной суммы необходимо отнять 2 пиксела. Таким образом, ширина двух соседних ячеек будет равна (72 x 2) — 2, или 146 пикселам.

а.Виджет выполняется в другом процессе, поэтому он должен всегда реагировать на запросы. Класс `Thread` нельзя использовать для этих целей, поскольку его экземпляр может быть уничтожен, когда завершится выполнение кода, связанного с обновлением содержимого виджета. Таким образом, служба запускается для выполнения фоновых операций.

Упражнения

3. Измените виджет в приложении «Been There, Done That!» таким образом, чтобы он отображал информацию о друге пользователя. (XML- данные, возвращаемые в результате запроса информации о набранных очках друга, содержат его ник, результат, позицию и адрес URL изображения аватара.)

4. Добавьте конфигурационную деятельность, которое позволило бы пользователю выбирать, информация о каком друге будет отображаться в виджете.

5. Продолжая работу над двумя предыдущими упражнениями, измените виджет таким образом, чтобы можно было создавать несколько его экземпляров, при этом в одном виджете должны отображаться данные пользователя, а в другом — данные его друга.

ЧАСТЬ IV. СОВЕРШЕНСТВОВАНИЕ ВАШЕГО ANDROID-ПРИЛОЖЕНИЯ

Час 19. ИНТЕРНАЦИОНАЛИЗАЦИЯ ВАШЕГО ПРИЛОЖЕНИЯ

Вопросы, рассматриваемые в этом часе:

- **языки, поддерживаемые платформой Android;**
- **управление строками и другими ресурсами;**
- **инструменты для форматирования значений с учетом локалей;**
- **другие моменты, связанные с интернационализацией.**

Рынок мобильных устройств поистине глобален — он обслуживает множество пользователей во многих странах и регионах. Разработчики должны принимать во внимание этот факт при проектировании и разработке приложений для платформы Android; с большой долей вероятности приложения будут использоваться пользователями, разговаривающими на разных языках. В этом часе вы узнаете о возможностях локализации, предоставляемых платформой Android, а также о сервисе Android Market.

КСТАТИ

В часе 24 вы узнаете, как подготовить ваше приложение для распространения через сервис Android Market.

ОБЩИЕ ПРИНЦИПЫ ИНТЕРНАЦИОНАЛИЗАЦИИ

Имея доступ к глобальному рынку, разработчики могут максимизировать свои доходы и расширить базовый контингент пользователей своего приложения, обеспечив поддержку различных языков и локалей. Давайте рассмотрим некоторые понятия. И хотя, скорее всего, вы знаете, что мы подразумеваем под языком, возможно, вам известно, что каждый язык может иметь несколько различных локалей. Например, испанский язык, на котором разговаривают в Испании, достаточно сильно отличается от испанского языка, на котором разговаривают в Америке; французский язык, на котором разговаривают в Канаде, отличается от французского языка, на котором разговаривают в Европе и в Африке; а английский язык, на котором разговаривают в Соединенных Штатах, отличается от английского языка, на котором разговаривают в Великобритании. Итак, «Английский» — это язык, а «Английский (США)», «Английский (Великобритания)» и «Английский (Австралия)» — это локали (рис. 19.1).

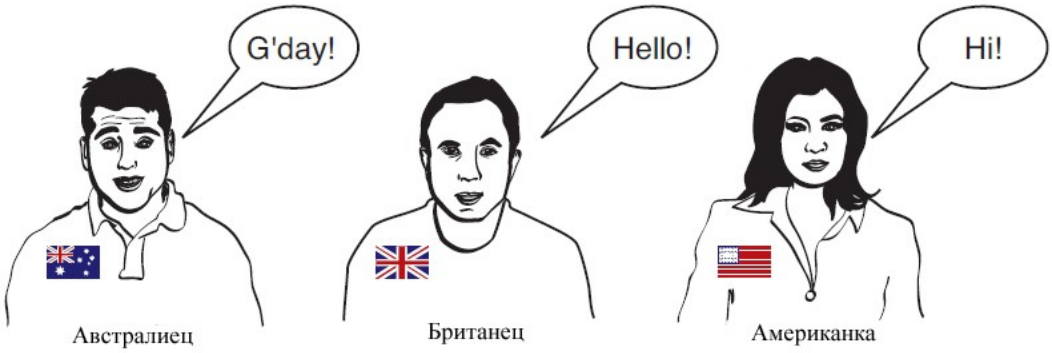

Рис. 19.1. Люди, разговаривающие на одном языке,
зачастую используют локальные диалекты

Приложения состоят из данных и функций (поведения). Для большинства приложений поведение не зависит от локали и остается одним и тем же. Однако данные должны быть локализованы. Это одна из основных причин, обуславливающих существование файлов ресурсов вынести наружу данные приложения.

Самый распространенный тип данных приложения, для которого требуется локализация, — тексты. Например, строка данных может представлять имя пользователя, однако название соответствующего параметра на экране приложения следовало бы отображать на подходящем языке (например, Name, Имя, Nom, Nombre).

Платформы разработки, поддерживающие возможность интернационализации приложений, обычно позволяют создавать таблицы строк, переключение между которыми обеспечивает поддержку различных языков в одном и том же приложении. Платформа Android — не исключение.

ВНИМАНИЕ!
Старайтесь избегать добавления строковой информации непосредственно в код приложения (т. е. в исходные файлы Java). В противном случае это станет помехой для интернационализации приложения.

КАК РАБОТАЕТ ЛОКАЛИЗАЦИЯ НА ПЛАТФОРМЕ ANDROID

В сравнении с другими платформами для мобильных устройств, инструментарий Android SDK обеспечивает достаточно широкую поддержку возможностей интернационализации приложений. Тем не менее /то довольно сложная тема с множеством подводных камней — и документации платформы Android по этой области фактически не существует.

Вопросы, связанные с локализацией Android-приложений, можно разбить на три основные категории:

- языки и локали, поддерживаемые платформой Android (обширный список — супермножество всех доступных языков);
- языки и локали, поддерживаемые конкретным мобильным телефоном Android (этот список может меняться — подмножество языков, выбранное производителем мобильного телефона или оператором сотовой связи);

- страны, языки и локали, поддерживаемые сервисом Android Market (страны и регионы, где компания Google может законно продавать программное обеспечение; этот список постоянно расширяется).

Конкретные локали, поддерживаемые платформой Android (на момент написания этой книги), представлены в табл. 19.1.

Таблица 19.1

Язык	Регионы
Китайский (zh)	КНР (zh_CN)
	Тайвань (zn_TW)
Чешский (cs)	Чехия (cs_CZ)
Голландский (nl)	Нидерланды (nl_NL)
	Бельгия (nl_BE)
Английский (en)	США (en_US)
	Великобритания (en_GB)
	Канада (en_CA)
	Австралия (en_AU)
	Новая Зеландия (en_SG)
	Сигнатур (en_SG)
Французский (fr)	Франция (fr_FR)
	Бельгия (fr_BE)
	Канада (fr_CA)
	Швейцария (fr_CH)
Немецкий (de)	Германия (de_DE)
	Австрия (de_AT)
	Швейцария (de_CH)
	Лихтенштейн (de_LI)
Итальянский (it)	Италия (it-IT)
	Швейцария (it_CH)
Японский (jp)	Япония (jp_JP)
Корейский (ko)	Южная Корея (ko_KR)
Польский (pl)	Польша (pl_PL)
Русский (ru)	Россия (ru_RU)
Испанский (es)	Испания (es_ES)

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Список локалей с большой вероятностью будет расширяться. Список всех локалей,

поддерживаемых платформой Android в настоящий момент, можно найти по адресу **developer.android.com/sdk/sndroid-2.1.html#locs**.

Как операционная система Android работает с локалями

Как и многие другие операционные системы, платформа Android имеет системную настройку для локали. Эта настройка имеет значение по умолчанию, которое может быть изменено мобильным оператором. Например, немецкий мобильный оператор может установить в качестве локали по умолчанию для распространяемых им мобильных телефонов значение Deutsch (Deutschland) (Немецкий (Германия)). Американский мобильный оператор, вероятнее всего, установит в качестве локали по умолчанию значение English (American) (Английский (США)), а также предоставит пользователю возможность выбора локали Espacol (Estados Unidos) (Испанский (США)) — тем самым, он обеспечивает поддержку американских диалектов английского и испанского языков.

Пользователь может изменить системную настройку локали в приложении **Settings** (Настройки). Настройка локали влияет на поведение приложений, установленных на мобильном устройстве.

ВЫПОЛНИТЕ САМОСТОЯТЕЛЬНО

Чтобы изменить настройку локали на мобильном телефоне, выполнить следующие шаги:

1. Открыв домашний экран, нажмите на мобильном телефоне кнопку **Menu** и вы берите пункт **Settings** (Настройки) в появившемся меню
2. В меню приложения **Settings** (Настройки) выберите пункт **Language & Keyboard** (Язык и клавиатура).
3. Нажмите на кнопку **Select Locale** (Выбрать регион) и выберите в появившемся меню нужную локаль. Платформа Android незамедлительно изменит настройки системы в соответствии с выбранной локалью. Например, если выбрать значение **Espanol** (Испанский), многие меню в операционной системе Android будут отображаться на испанском языке.

Постарайтесь запомнить эти шаги, поскольку для возврата к меню выбора локали вам придется использовать выбранный иностранный язык.

Как приложения работают с локалями

Давайте теперь рассмотрим, как системная настройка локали влияет на каждое Android-приложение. Когда в Android-приложении используются ресурсы проекта, операционная система Android пытается найти наиболее подходящий ресурс для данной задачи на этапе выполнения приложения. Во многих случаях это подразумевает поиск ресурса для определенного языка или локали. Если ресурсы для указанной локали отсутствуют, система применяет ресурс, используемый по умолчанию.

Разработчики могут определять ресурсы для конкретных языков и локалей, размещая эти ресурсы в каталогах проекта со специальными именами. Любой ресурс приложения может быть локализован, независимо от того, содержит ли файл строковые ресурсы, изображения, анимационное последовательности или ресурсы другого типа.

ОПРЕДЕЛЕНИЕ РЕСУРСОВ, ИСПОЛЬЗУЕМЫХ ПО УМОЛЧАНИЮ

До сих пор все ресурсы приложения «Been There, Done That!» были ресурсами, используемыми по умолчанию. Такой ресурс не имеет специальных тегов, управляющих его загрузкой в различных ситуациях.

Ресурсы, используемые по умолчанию, наиболее важные, поскольку они применяются в ситуации, когда отсутствует конкретный специализированный ресурс (что случается довольно часто). В случае с приложением «Been There, Done That!» все ресурсы, используемые по умолчанию, переведены на английский язык.

ОПРЕДЕЛЕНИЕ РЕСУРСОВ ДЛЯ КОНКРЕТНОГО ЯЗЫКА

Чтобы определить строки для конкретного языка, вы должны поместить соответствующий ресурс в каталог со специальным именем, содержащим двухбуквенный код языка в соответствии со стандартом ISO 639-1 (список кодов языков можно найти по адресу **www.loc.gov/standards/iso639-2/php/code_list.php**). Например, английский язык обозначается кодом en, французский язык — кодом fr, а немецкий — кодом de. Давайте рассмотрим на примере, как эго работает.

Предположим, что вы хотите, чтобы приложение «Been There, Done That!» поддерживало строки на английском, немецком и французском языках. Для этот вам понадобилось бы выполнить следующие шаги:

1. Создать файл ресурсов **strings.xml** для каждого языка. Каждая локализуемая строка во всех файлах ресурсов должна иметь одинаковое имя, чтобы обеспечить ее правильную загрузку программным путем. Строки, которые вы не хотите локализовывать, можно оставить в файле **/res/values/strings.xml**, используемом по умолчанию (на английском языке).

2. Сохранить файл ресурсов **strings.xml** для французского языка в каталоге **/res/values-fr/**.

3. Сохранить файл ресурсов **strings.xml** для немецкого языка в каталоге **/res/values-de/**.

Операционная система Android теперь сможет выбрать нужную строку в соответствии с системной настройкой локали. Однако, если локализованная строка не существует, система будет использовать строку, определенную в каталоге **/res/values/**. Это значит, что если выбрана локаль для английского языка (или арабского, или китайского, или японского, или совершенно другого языка), будут использованы строки для английского языка (запасной вариант), указанные по умолчанию.

Подобным образом вы могли бы предоставить графический ресурс для немецкого языка, переопределив соответствующий ресурс, который находится в каталоге **/res/drawable/**, для чего достаточно поместить файл изображения с таким же именем в каталог **/res/drawable-de/**.

ОПРЕДЕЛЕНИЕ РЕСУРСОВ ДЛЯ КОНКРЕТНОГО РЕГИОНА

Вы могли обратить внимание, что в предыдущем примере ресурсы определялись для языка верхнего уровня (для английского языка в целом, но не для американского, британского и австралийского диалектов английского языка). Не беспокойтесь! Вы можете включить регион или локаль в название каталога с ресурсами.

Чтобы определить строки для конкретного языка и локали, вы должны поместить соответствующий ресурс в каталог со специальным именем, которое включает двухбуквенный код языка в соответствии со стандартом ISO 639-1 (список кодов языков можно получить по адресу **www.loc.gov/standanls/iso639-2/php/code_list.php**), затем следует дефис и строчная буква «r», и, наконец, код региона ISO 3166-1-alpha-2 (коды регионов можно найти по адресу **www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm**). Например, для американского диалекта английского языка соответствующая часть названия каталога будет выглядеть как en-rUS, для британского диалекта — en-rGB, а для австралийского диалекта — en-rAU. Давайте рассмотрим на примере, как это работает.

Если бы вы хотели, чтобы приложение «Been There, Done That!* поддерживало все эти три версии английского языка, вы могли бы сделать следующее:

1. Создать файл ресурсов **strings.xml** для каждого языка. Строки, которые вы не хотите локализовывать, можно оставить в файле **/res/values/strings.xml**, используемом по умолчанию (на английском языке).

2. Сохранить файл ресурсов **strings.xml** для британского диалекта английского языка в каталоге **/res/values-en-rGB/**.

3. Сохранить файл ресурсов **strings.xml** для австралийского диалекта английского языка в каталоге **/res/values-en-rAU/**.

Подведем итоги. Сначала вы создаете набор ресурсов, используемый по умолчанию, — этот набор ресурсов должен быть создан для языка, который будет чаще всего использоваться в вашем приложении. Затем по мере необходимости вы добавляете исключения — например, строковые значения для конкретного языка и региона. Таким образом, вы можете оптимизировать ваше приложение, чтобы оно выполнялось на различных платформах.

ВНИМАНИЕ!
Более подробное объяснение механизма поиска подходящих ресурсов операционной системой Android можно найти на веб-сайте, посвященном разработке на платформе Android, по адресу **developer.android.com/guide/topics/resources/resources-i18n.html#best-match**.

Как сервис Android Market работает с локалями

Сервис Android Market поддерживает подмножество локалей, доступных на платформе Android. Поскольку для осуществления платежей в сервисе Android Market применяется сервис Google Checkout, для платных приложений могут поддерживаться только те страны, в которых данная торговая интернет-площадка разрешена законом.

Полный список стран и языков, поддерживаемых сервисом Android Market, можно найти по адресу **market.android.com/support/bin/answer.py?hl=en&answer= 138294**. Стоит отметить, что сервис Android Market проводит различия между странами, в которых разрешено распространение только бесплатных приложений, и странами, где разработчики могут распространять свои приложения за определенную плату.

ВНИМАНИЕ!
Для продажи приложений через сервис Android Merket разработчики должны зарегистрироваться Полный список стран, где могут находиться разработчики бесплатных Android-приложений (который отличается от списка стран, где разработчики могут распространяться свои приложения), можно найти по адресу **market. android.com/support/bin/answer.py?hl=en&answer=136758**.

СТРАТЕГИИ ИНТЕРНАЦИОНАЛИЗАЦИИ ANDROID-ПРИЛОЖЕНИЙ

Пусть вас не ошеломляет количество возможностей, доступных для разработчиков. когда дело касается интернационализации приложения. Вместо этого подумайте о том, насколько важна интернационализация для вашего приложения на этапе проектирования. Разработайте стратегию, удовлетворяющую вашим требованиям, и придерживайтесь ее.

Вот некоторые из основных стратегий интернационализации Android- приложений:

- полный отказ от интернационализации приложения;
- ограниченная интернационализация приложения;
- реализация полной интернационализации для целевых аудиторий.

Теперь давайте поговорим о каждой из этих стратегий более подробно.

Отказ от интернационализации приложения

При любой возможности избавляйте ваши команды разработчиков и тестировщиков от большого объема работы — отказывайтесь от интернационализации вашего приложения. Можно сказать, что это подход «один размер подходит для большинства» в области разработки приложений для мобильных устройств, и его применение зачастую возможно в играх и других простых приложениях с большим количеством графики. Если в вашем приложении вместо текстовых надписей можно легко использовать графические обозначения, принятые во всем мире (например, «воспроизведение», «пауза», «стоп» и т. д.), или язык «симов» (неразборчивое бормотание, смысл которого будет понятен всем слушателям, независимо от используемого ими языка), у вас есть возможность полностью отказаться от интернационализации вашего приложения. Тем не менее такой подход работает лишь для определенного подмножества приложений. Например, если в вашем приложении должен присутствовать экран помощи, вам, скорее всего, понадобится локализовать, по крайней мере, часть вашего приложения, чтобы это приложение могли использовать люди по всему миру.

Вот некоторые преимущества использования данной стратегии:

- Упрощается процесс разработки и тестирования.
- Минимально возможный размер приложения (поскольку используется только один набор ресурсов).

Среди недостатков данной стратегии можно отметить следующее:

- Для приложений, которые зависят от культуры или в пользовательском интерфейсе которых используется большое количество текста, этот подход значительно уменьшает ценность приложения. Приложение попросту становится слишком обобщенным.

- Эта стратегия автоматически отстраняет определенные аудитории от вашего приложения и ограничивает число потенциальных рынков для его распространения.

Ограниченная интернационализация приложения

Для большинства приложений требуется лишь «поверхностная» интернационализация. Зачастую это подразумевает только интернационализацию строковых ресурсов, а другие ресурсы, например макеты и графика, остаются одинаковыми для всех языков и локалей.

К достоинствам данной стратегии можно отнести следующее:

- умеренные затраты на разработку и тестирование приложения;
- оптимальный размер приложения (количество специализированных ресурсов остается минимальным).

Среди недостатков данной стратегии можно отметить следующее:

- Приложения некоторых типов могут по-прежнему оставаться слишком обобщенными. От необходимости поддержки нескольких целевых языков может страдать общий дизайн приложения (особенно, дизайн экранов). Например, чтобы поддержать такие «многословные» языки, как немецкий, может понадобиться увеличить размеры полей ввода, которые будут выглядеть уродливо и занимать ценное пространство экрана при использовании менее «многословных» языков.

- Поскольку вы ступили на путь перевода ресурсов для конкретных языков, пользователи вашего приложения, вероятнее всего, будут ожидать добавления поддержки других языков, которые пока не поддерживаются приложением. Другими словами, существует большая вероятность того, что, добавив в приложение поддержку некоторых языков, в скором времени вы начнете получать запросы от пользователей на добавление поддержки других языков. С другой стороны, вы уже разработали и протестировали ваше приложения для ряда языков, поэтому добавление новых языков не должно составить никакого труда.

Полная интернационализация приложения

Для некоторых типов приложений требуется их полная интернационализация. Перевод всех ресурсов для каждого поддерживаемого языка и локали — довольно трудоемкое занятие, и вы не должны делать это, не имея веских оснований, поскольку по мере включения дополнительных ресурсов увеличивается размер приложения. Использование данного подхода зачастую влечет за собой разбиение ресурсов для конкретных языков на отдельные APK-файлы для публикации, что приводит к более сложному управлению конфигурацией. Однако благодаря этому подходу разработчик имеет возможность практически полностью адаптировать свое приложение для каждого конкретного рынка.

К основным достоинствам данной стратегии можно отнести следующее:

- Приложение полностью адаптировано и настроено для конкретных целевых аудиторий; эта стратегия позволяет подстраивать приложение под конкретные локали.

- Эта стратегия позволяет завоевать лояльность пользователей, предоставляя им возможности максимально удобной работы с приложением. (Этот подход также используется компанией Google.)

К недостаткам данной стратегии можно отнести следующее:

- Это наиболее трудоемкая и сложная стратегия в плане реализации.
- Каждая интернационализированная версия приложения должна быть полностью протестирована, как если бы вы разрабатывали совершенно новое приложение (что отчасти справедливо, если речь идет о разбиении ресурсов на различные APK-файлы с целью минимизации размера приложения).

ВНИМАНИЕ!

Избегайте излишней интернационализации вашего приложения. Размер пакета приложения увеличивается по мере добавления ресурсов для конкретных языков и локалей. Нет никакой необходимости двигаться в этом направлении, если для этого отсутствует объективная причина — и если у вас нет специалистов по разработке, тестированию и выпуску продукта для решения данной задачи. Плохо локализованная версия приложения может нанести гораздо больший вред вашему имиджу, чем приложение, не имеющее никакой локализации вообще.

ИСПОЛЬЗОВАНИЕ ИНСТРУМЕНТОВ, ПРЕДНАЗНАЧЕННЫХ ДЛЯ ЛОКАЛИЗАЦИИ ПРИЛОЖЕНИЙ

Инструментарий Android SDK содержит инструменты для работы с информацией о локали. Например, класс `Locale` (`java.util.Locale`) представляет информацию, связанную с локалью.

Определение системной локали

Если вам необходимо изменить поведение приложения в соответствии с информацией о локали, вы должны иметь возможность получать информацию об операционной системе Android. Это можно сделать при помощи метода `getConfiguration()` объекта типа `Context`, как показано в следующем коде:

```
Configuration sysConfig = getResources().getConfiguration();
```

Одна из системных настроек, доступных через объект типа `Configuration`, это локаль:

```
Locale curLocale = sysConfig.locale;
```

Вы можете при необходимости использовать эту информацию о локали, чтобы изменять поведение приложения программным путем.

Форматирование строкового представления даты и времени

Еще один аспект интернационализации приложения — правильное отображение данных. Например, для отображения даты в США используются форматы «ММ/ДД/ГГ» и «December 8, 1975», а во многих других странах мира используются форматы «ДД/ММ/ГГ» и «8 December, 1975». Инструментарий Android SDK предоставляет ряд вспомогательных инструментов для форматирования строк в соответствии с локалью. Например, вы можете использовать класс DateFormat (`android.text.format.DateFormat`) для генерации строк, представляющих дату и время в текущей локали, а также вы можете настроить формат отображения информации о дате и времени в соответствии с требованиями, предъявляемыми к вашему приложению.

ЗНАЕТЕ ЛИ ВЫ ЧТО...

Вы можете использовать класс `TimeUtils`(`android.util.TimeUtilsl`), чтобы определить временную зону конкретной страны по ее названию.

Работа с валютами

Подобно датам и времени, используемые валюты и их форматирование во многом зависят от локали. Вы можете использовать стандартный Java-класс `Currency` (`java.util.Currency`) для хранения информации о валюте. Вы можете использовать класс `NumberFormat` (`java.text.NumberFormat`) для форматирования и разбора чисел в соответствии с информацией о локали.

ИТОГИ

В этом часе были рассмотрены основные принципы интернационализации приложений, например вынесение ресурсов проекта за пределы кода и определение целевых рынков. Вы узнали, как платформа Android позволяет работать с разными странами, языками и локалями. Наконец, вы познакомились с организацией ресурсов Android-приложений для поддержки ряда различных стран и регионов с целью получения максимальной прибыли, используя различные стратегии интернационализации приложений.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Какие языки и локали должны поддерживаться моими Android- иркложениями?

Ответ: Ответ на этот вопрос зависит от множества факторов и напоминает игру в числа. Если коротко; минимально возможное количество языков и локалей, которое позволит вам приступить к распространению приложения. Количество мобильных пользователей, использующих определенный язык, не должно являться решающим фактором при определении того, какие языки поддерживать. Например, мобильных пользователей, разговаривающих на испанском или китайском языках гораздо больше, чем пользователей, разговаривающих на английском языке, однако пользователи англоязычного рынка готовы платить гораздо более высокую цену за приложения. Таким образом, ответ зависит от знания вашей целевой аудитории(й) пользователей, которая должна быть отражена в вашем бизнес-плане.

Вопрос: Почему на моем мобильном телефоне Android присутствует только часть языков и локалей, перечисленных в этом часе?

Ответ: Несмотря на то, что платформа Android поддерживает множество языков и локалей, производители мобильных телефонов и операторы сотовой связи могут ограничивать список поддерживаемых локалей на конкретных устройствах. Это может делаться с целью экономии ресурсов. Например, телефон, продаваемый оператором сотовой связи в США, может поддерживать только американский диалект английского и испанского языков.

Вопрос: Какой язык должен применяться для ресурсов, используемых по умолчанию, таких как строки?

Ответ: Ваши ресурсы, используемые по умолчанию, должны представлять язык/локаль, используемый самой большой целевой аудиторией вашего приложения, — это наиболее общие/применимые значения, которые будут импонировать большинству пользователей. Если планируется, что вашим приложением будут пользоваться люди по всему миру, зачастую выбор падает на английский язык, однако этим языком не обязательно должен быть английский язык. Например, если ваше приложение позволяет получать пошаговые инструкции по маршруту в какой-либо части Китая, вероятно, вы захотите, чтобы языком/локалью, используемым по умолчанию, был один из диалектов китайского языка (и даже на территории Китая одни настройки локали используются гораздо чаш, чем другие) – за исключением тех случаев, когда ваша целевая аудитория – бизнесмены, посещающие Китай, и в этом случае вы остановитесь на английском языке, который по-прежнему остается языком международного бизнеса (по крайней мере, пока).

Вопрос: Я выбрал локаль с испанским языком. Почему в неюлорш приложениях по-прежнему используется английский язык?

Ответ: Если строковые ресурсы приложения, используемые по умолчанию, представлены на английском языке и для испанского языка нет доступных ресурсов, будут использоваться ресурсы по умолчанию, независимо от выбранного языка.

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Android-приложение может поддерживать несколько языков в одном APK-файле.

2. Верно ли это? Количество языков, поддерживаемых платформой Android и сервисом Android Market, неизменно.

3. Какой язык должен применяться для ресурсов вашего приложения, используемых по умолчанию?

A. Английский.
B. Китайский.
C. Ниболее удобный для вашей целевой аудитории язык.
D. Другой язык.

Ответы

1. Верно. Приложение может быть скомпилировано вместе с ресурсами для нескольких различных языков. Платформа Android позволяет переключаться между этими ресурсами в процессе выполнения приложения, в зависимости от настроек локали на мобильном телефоне.

2. Неверно. Список языков, поддерживаемых платформой Android, постоянно обновляется. Новые языки и локали добавляются непрерывно.

3. С. Ресурсы вашего приложения, используемые по умолчанию, должны быть на том языке, с которым, вероятнее всего, будут работать пользователи после загрузки приложения. Следовательно, имеет смысл разрабатывать эти ресурсы на том языке и с использованием той локали, которые наиболее удобны для большинства пользователей.

Упражнения

1. Добавьте новый набор значений строковых «Been There Done That!» для некоторого языка или локали по вашему выбору. Проверьте результаты изменений на эмуляторе Android или на реальном мобильном телефоне (если на нем поддерживается выбранная вами язык/локаль).

2. Измените приложение «Been There, Done That!» таким образом, чтобы оно загружало графическое изображение или ресурс цвета для определенного языка или локали. Например, измените изображение планеты на экране с главным меню на изображение, имеющее отношение к выбранному языку/локали. Проверьте результаты изменений на эмуляторе Android или на реальном мобильном телефоне (если на нем поддерживается выбранный вами язык/локаль).

Час 20. РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ РАЗЛИЧНЫХ УСТРОЙСТВ

Вопросы, рассматриваемые в этом часе:

- **разработка дизайна для различных конфигураций мобильных телефонов;**
- **реагирование на изменения ориентации экрана;**
- **работа с различными версиями инструментария Android SDK.**

Платформа Android развивается ускоренными темпами. Каждые несколько месяцев появляются новые версии инструментария Android SDK вместе с появлением все новых и новых мобильных телефонов. В этом часе вы узнаете, как разрабатывать Android-приложения для различных целевых устройств. Устройства под управлением операционной системы Android отличаются как в плане аппаратных и программных возможностей, так и в плане версии установленного на них инструментария Android SDK.

УПРАВЛЕНИЕ КОНФИГУРАЦИЕЙ НА ПЛАТФОРМЕ ANDROID

Разработчики должны стараться поддержать максимально возможный диапазон мобильных устройств и при этом не переоценить свои возможности с точки зрения сопровождения приложения и управления конфигурацией. При выборе целевых платформ должны учитываться следующие факторы:

- Какие аппаратные возможности требуются приложению? Нужен ли приложению сенсорный экран? Аппаратная клавиатура? Многопозиционный джойстик? Экран определенных размеров?

- Какие программные возможности требуются приложению? Поддерживает ли приложение различные ориентации экрана?

- Какая версия инструментария Android SDK требуется приложению?

И хотя некоторые решения могут повлечь за собой изменения в библиотеках проекта и в файле манифеста Android, многие требования можно реализовать путем использования той же стратегии использования квалификаторов в названиях каталогов ресурсов, применявшейся для интернационализации приложения.

Использование квалификаторов в названиях каталогов ресурсов позволяет предоставить ресурсы для ряда различных конфигураций приложения (табл. 20 1). Вы можете применять эти квалификаторы названий каталогов к вложенным подкаталогам, например **/res/values/**. Квалификаторы присоединяются, к названию существующего подкаталога в строгом порядке, который определен в табл. 20.1. В названии каталога могут использоваться сразу несколько квалификаторов, разделенных дефисами. Квалификаторы всегда записываются строчными буквами, и название каталога может содержать только один квалификатор каждого типа. Пользовательские квалификаторы использовать нельзя.

Таблица 20.1

Тип квалификатора каталога	Значения	Комментарии
Язык	**ru**, **en**, **fr**, **es**, **zh**, **ja**, **ko**, **de** и т.д.	Двухбуквенные коды языков в соответствии со стандартом ISO 639-1
Регион/локаль	**rRu**, **rUS**, **rGB**, **rFR**, **rJP**, **rDE** и т.д.	Код региона ISO 3166-1alpha-2, записанный ЗАГЛАВНЫМИ БУКВАМИ, перед которым указывается строчная буква «r»
Размеры экрана	**small**, **normal**, **large**	Размер экрана и коэффициент плотности размещения точек на экране
Ориентация экрана	**port**, **land**	Портретный режим, альбомный режим
Плотность размещения точек на экране	**ldpi**, **mdpi**, **hdpi**, **nodpi**	Плотность размещения точек на экране, для которой предназначен ресурс
Тип сенсорного экрана	**noutouch**, **stylus**, **finger**	Сенсорный экран отсутствует, сенсорный экран, предназначенный исключительно для работы со стилусом, обычный сенсорный экран
Доступность клавиатуры	**keysexposed**, **keyshidden**, **keyssoft**	Клавиатура доступна, клавиатура недоступна пользователю, ресурсы используются только вместе с программной клавиатурой
Доступность навигационных клавиш	**navexposed**, **navhidden**	Навигационные клавиши доступны или скрыты (клавиатура телефона закрыта)
Основной метод навигации для	**nonav**, **dpad**, **trackball**, **whell**	Четырехпозиционный джойстик, трекбол, колесо прокрутки

несенсорных экранов		
Версия инструментария SDK	**v1**, **v2**, **v3**, **v4**, **v5**, **v6**, **v7** и т.д.	Идентификатор API Level версии инструментария SDK (например, значение **v1** соответствует инструментарию Android SDK 1.0, а также значение **v7** – инструментарию Android SDK 2.1)

Вы можете объединить каталоги ресурсов для определенных конфигураций используя дефисы для разделения квалификаторов. Вот несколько хороших примеров названий каталогов с правильным использованием квалификаторов :

```
/res/values-en-rUS-port-finger
/res/drawables-en-rUS-land
/res/values-en-qwerty
```

И несколько примеров названий каталогов с неправильным использованием квалификаторов:

```
/res/values-en-rUS-rGB
/res/values-en-rUS-port-FINGER
/res/values-en-rUS-port-finger-custom
```

Исчерпывающий список квалификаторов, доступных для использования при настройке ресурсов (мобильный код страны, код оператора, размер экрана и так далее), можно найти на веб-сайте, посвященном разработке на платформе Android, по адресу **developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources**.

Поддержка различных ориентаций экрана

Android-приложения могут выполняться в альбомном или портретном режиме, в зависимости от того, как пользователь располагает экран мобильного телефона. Помимо интернационализации приложений, одна из наиболее распространенных ситуаций, когда может понадобиться адаптация ресурсов, — это поддержка изменений ориентации экрана.

ДОБАВЛЕНИЕ АДАПТИРОВАННОГО МАКЕТА ДЛЯ АЛЬБОМНОГО РЕЖИМА

До этого момента вы разрабатывали и тестировали приложение «Been There, Done That!» в портретном режиме. Запустите приложение сейчас, переведя перед этим мобильный телефон или эмулятор в альбомный режим. Просмотрите каждый экран приложения. Вы увидите, что некоторые экраны отображаются отлично (в этих экранах применяются макеты с динамически изменяемой шириной, которые прекрасно работают как в альбомном, так и в портретном режиме), а для некоторых экранов, например игрового экрана (рис. 20.1), может потребоваться доработка.

Изменить внешний вид экрана в зависимости от ориентации устройства не сложнее, чем добавить новый набор файлов ресурсов с определениями макетов. Для этого вам нужно сделать следующее:

- Использовать все существующие ресурсы макетов в качестве ресурсов по умолчанию.

- Разработать новую версию каждого макета, которая выглядела бы привлекательно в альбомном режиме.

- Добавить все новые макеты для альбомного режима в каталог **/res/ layout-land/**.

Давайте попробуем выполнить необходимые шаги, создав две различные версии макета игрового экрана для приложения «Been There, Done That!» — одну версию для портретного режима (по умолчанию) и другую версию для альбомного режима.

Рис. 20.1. Игровой экран в альбомном режиме (используется стандартный макет)

КСТАТИ

Полная реализация изменений, связанных с поддержкой альбомного режима и различных мобильных телефонов, доступна на диске, прилагаемом к данной книге, в папке **/Книга/Час 20**.

Сначала давайте рассмотрим дизайн, реализованный в существующем файле макета **game.xml** и изображенный на рис. 20.2.

Рис. 20.2. Дизайн игрового экрана
(версия макета, используемая по умолчанию)

В альбомном режиме для размещения элементов `ImageView`, `TextView` и `Button` по вертикали просто не хватает пространства. Чтобы оптимизировать этот макет для альбомного режима, можно рассмотреть возможность размещения элемента `TextView` справа от элемента `ImageView`. Также при необходимости можно было бы изменить размеры изображения или кнопок. В обеих версиях файла макета используются одни и те же элементы (с теми же названиями); в версии макета для альбомного режима изменено только расположение этих элементов (рис. 20.3).

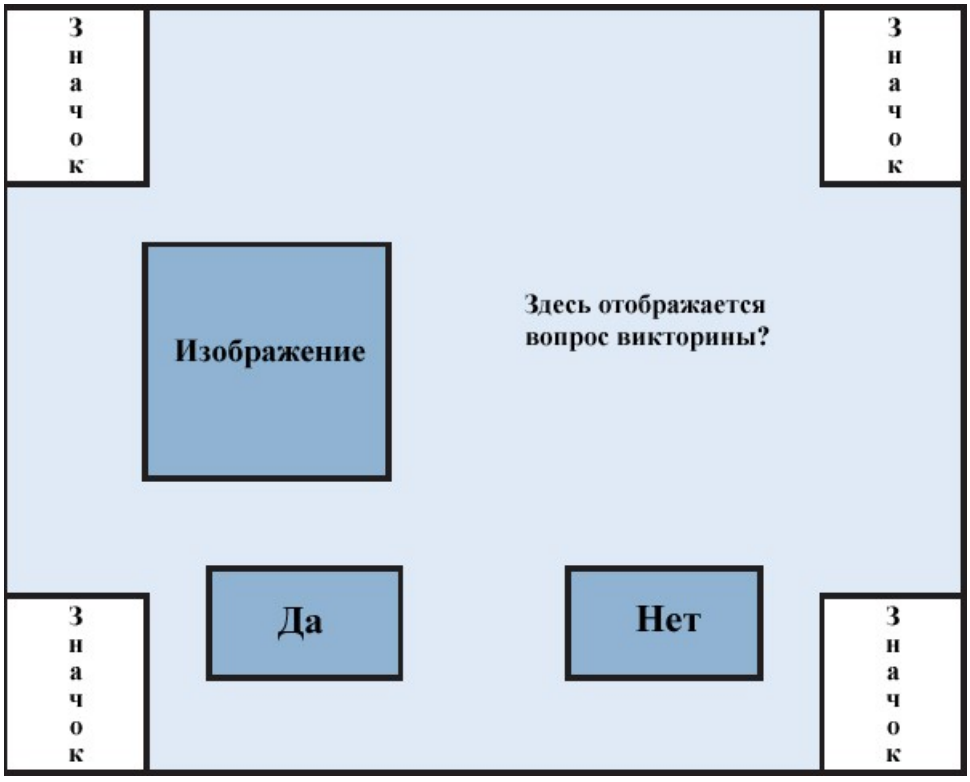

Рис. 20.3. Дизайн игрового экрана
(измененная версия макета для альбомного режима)

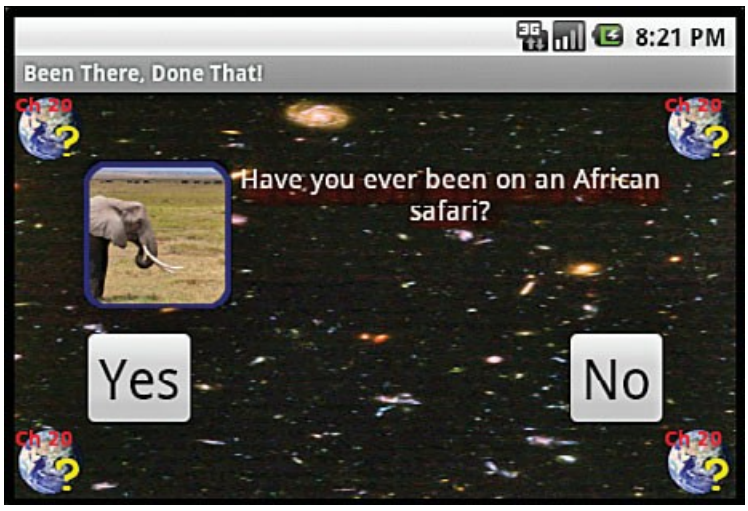

Рис. 20.4. Игровой экран в альбомном режиме
(используется макет, адаптированный для альбомного режиме)

При переключении устройства в альбомный режим результирующий экран выглядит намного лучше, как показано на рис. 20.4.

ВНИМАНИЕ!
Поскольку изменение ориентации экрана приводит к перезапуску текущей деятельности, любые выполняющиеся задачи, например, операции декодирования изображений или сетевые операции, будут запущены снова, если вы не реализуете метод `onRetainNonConfigurationInstance()` класса `Activity`. Дополнительную информацию по ситуациям подобного рода можно найти в статье на веб-сайте, посвященном разработке на платформе Android, по адресу **developer.androjd.com/resources/articles/faster-screen-orientation-change.html**.

ПРОСЛУШИВАНИЕ СОБЫТИЙ ИЗМЕНЕНИЯ ОРИЕНТАЦИИ ЭКРАНА

Приложения могут регистрировать слушателей дли получения событий изменения ориентации экрана. Для этого вы получаете объект типа `SensorManager` при помощи метода `getSystemService()`. Вы можете получить текущую ориентацию экрана, используя метод `getOrientation()` объекта типа `SensorManager`. В качестве альтернативного варианта, вы можете реализовать класс `OrientationEventListener(android.view.OrientationEventListener)` и переопределить метод `onOrientationChanged()`, чтобы зарегистрировать слушателя для получения событий изменения ориентации экрана.

Тем не менее прослушивание событий изменении ориентации требуется только в том случае, когда приложения должны выполнять специальные внутренние операции при смене ориентации экрана. Приложение, для которого определен каталог **/res/layout-land/** с ресурсами макетов для альбомного режима и каталог **/res/layout/** с ресурсами макетов для портретного режима, используемыми по умолчанию, будет работать, не вызывая каких либо проблем, даже без слушателя событий изменения ориентации экрана.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Вы можете сменить ориентацию экрана эмулятора, нажав комбинацию клавиш **Ctrl+F11** и **Ctrl+F12**.

СТРАТЕГИИ ПОДДЕРЖКИ РАЗЛИЧНЫХ ОРИЕНТАЦИЙ ЭКРАНА

Самый лучший способ поддержать различные ориентации экрана — разработать достаточно простые макеты, которые будут прекрасно работать как в портретном, так и в альбомном режиме без внесения дополнительных изменений. Например, экран с настройками приложения «Been There, Done That!» прекрасно работает в обоих режимах, поскольку все параметры размещены друг над другом в элементе-контейнере `LinearLayout`, который, в свою очередь, находится в области с возможностью прокрутки, что позволяет масштабировать содержимое этого экрана до любых размеров. Тем не менее, для некоторых макетов, например, экрана-заставки или игрового экрана, может потребоваться дополнительная доработка для каждого режима.

Существует множество стратегий поддержки экранов различных размеров и режимов ориентации экранов. Вот несколько советов по разработке макетов, которые будут работать на экранах различных типов:

- Не перегружайте экраны информацией. Они должны быть простыми.
- Используйте элементы-представления с возможностями масштабирования, например элементы `ScroliView` и `ListView`.

- Масштабируйте и увеличивайте экраны только в одном направлении (по вертикали или по горизонтали), но не в обоих направлениях.

- Не вставляйте в код позиции элементов экрана. Вместо этого используйте относительные позиции и макеты с относительным позиционированием элементов, например элементы-контейнеры `RelativeLayout`.

- Избегайте использования элементов-контейнеров `AbsoluteLayout` и других макетов с абсолютным позиционированием элементов.

- Используйте растягиваемые изображения, например, изображения в формате Nine-Patch.

- Минимизируйте размеры ресурсов, чтобы обеспечить их быструю загрузку при изменении ориентации экрана.

ВНИМАНИЕ!

На веб-сайте, посвященном разработке на платформе Android, представлен ряд полезных рекомендаций по реализации поддержки различных экранов, которые доступны по адресу **developer.android.com/guide/practices/screens_support.html**.

Поддержка экранов с различными характеристиками

Устройства Android поставляются с экранами, которые отличаются между собой по широкому спектру характеристик, включая размеры экранов плотности размещения точек на

экране, соотношения сторон экранов и разрешения. Кроме того, на различных устройствах используются различные настройки для отображения информации, включая темы и стили. Перед выпуском приложения вы должны проверить наше приложение на всех целевых платформах. С большой долей вероятности поведение и отображение вашего приложения на каждом устройстве будет немного отличаться. Характеристики экрана — важнейший фактор, который необходимо принимать во внимание при проектировании пользовательских интерфейсов.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Используя приложение Android SDK and AVO Manager, вы можете создавать специальные AVD-профили, чтобы имитировать на эмуляторе поведение экранов с особыми характеристиками.

Как видно на рис. 20.5, один и тот же макет выглядит по-разному на различных экранах. Слева показан типовой HVGA-экран со средней плотностью размещения точек, а справа показан экран устройства Motorola Droid, который представляет собой WVGA854-экран с высокой плотностью размещения точек.

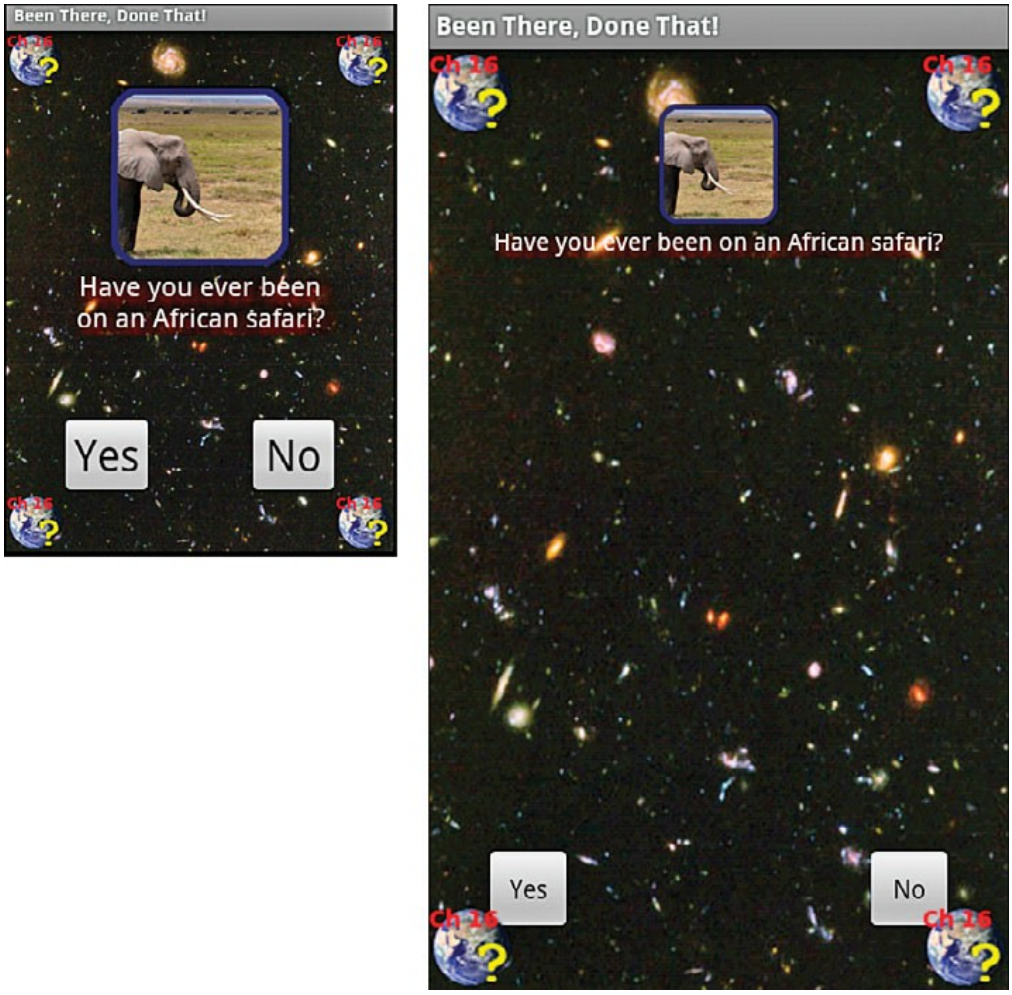

Рис. 20.5. Макет может отображаться по-разному на разных устройствах

Но не стоит падать духом. В исправлении проблем подобного рода нет ничего сложного. Вот несколько подсказок, которые в первую очередь позволят избежать проблем, связанных с плотностью размещения точек на экране, и подобных им:

- Устанавливайте только необходимые значения атрибутов (не нужно поддерживать необязательные настройки).

- Все значения размеров должны храниться в файле ресурсов с размерами, а не в файлах макетов.

- При указании размеров шрифтов используйте единицы измерения **dp** или **sp** вместо единиц измерения **pt**.

- Для указания размеров в пикселах используйте единицы измерения **dp** вместо единиц измерения **px**.

- При необходимости добавьте адаптированные альтернативные ресурсы (но делать это нужно обдуманно).

После внесения перечисленных изменений в проект приложения «Всеп There, Done That!», как мы видим, приложение отображается правильно на широком спектре устройств, включая устройство Motorola Droid.

Поддержка возможностей различных мобильных телефонов

Как было показано в табл. 20.1, разработчики могут создавать файлы ресурсов, адаптированных к различным конфигурациям мобильных телефонов.
В случае с игрой определенные ресурсы могут быть адаптированы для мобильного телефона, у которого отсутствует аппаратная клавиатура или который имеет особый тип сенсорного экрана или многопозиционного джойстика. Графические файлы могут быть увеличены для очень производительных мобильных телефонов, имеющих экраны с высоким разрешением, а для типовых мобильных телефонов эти файлы могут быть уменьшены чтобы сэкономить дисковое пространство. В крайних случаях на одном устройстве может применяться двухмерная графика, а на другом – трехмерная.

Разработка приложений для различных версий инструментарий Android SDK

На момент создания этой книги в руках пользователя было несколько различных версий инструментария Android SDK: Android 1.1, Android 1.5, Android 1.6, Android 2.0, Android 2.0.1, Android 2.2 и Android 2.3. Этот список в скором времени пополнится новыми версиями (с кодовыми именами Froyo и Gingerbread). Время от времени компания Google публикует информацию об использовании различных версий инструментария Android SDK на мобильных телефонах:

- 4,7 % пользователей используют инструментарий Android SDK 1.5.
- 7,9% пользователей используют инструментарий Android SDK 1.6.
- 35,2% пользователей используют инструментарий Android SDK 2.1.
- 51,8% пользователей используют инструментарий Android SDK 2.2.
- 0,4% пользователей используют инструментарий Android SDK 2.3.

Эти данные были собраны и опубликованы на веб-сайте, посвященном разработке на платформе Android, в течении двух недель до 4 января 2011 года. Вы можете увидеть обновленную статистику на этом сайте по адресу **developer.android.com/resources/dashboard/platform-versions.html**. Один крайне интересный показатель некоторые версии инструментария SDK фактически не используются на подавляющем большинстве мобильных устройств, поскольку быстро заменяются новыми (например, версия 2.1 повсеместно вытеснила 2.0). Эта информация просто бесценна для сокращения затрат на тестирование приложения, поскольку вы можете протестировать приложение только на тех платформах, которые актуальны на сегодняшний день. Определенные модели телефонов, особенно старые модели, могут не поддерживать загрузку обновлений системы. Как видно из представленных выше данных, если вы хотите угодить максимально возможному количеству пользователей, вам может потребоваться разработать приложение для нескольких различных версий инструментария SDK. Подробная информация также может оказать неоценимую помощь, позволив при тестировании сосредоточить внимание только на подходящих платформах.
Поиск новой информации, слежение за новостями на рынке и проведение опросов среди целевой аудитории вашего приложения – все это, безусловно, поможет вам определить целевую платформу для вашего приложения.

ВЫБОР ЦЕЛЕВОЙ ПЛАТФОРМЫ ДЛЯ ПРИЛОЖЕНИЯ

Чтобы приложение было интересно большинству пользователей, перед тем как приступить к разработке любого Android-приложения, вы должны подумать о вашей целевой платформе. Будет ли ваше приложение поддерживать более старые и более и известные мобильные устройства или только новые устройства? Проведите исследование рынка, чтобы определить, какие версии инструментария SDK использует долевая аудитория вашего приложения.

ВНИМАНИЕ!
Обратная совместимость версий инструментария SDK на платформе Android не гарантируется. Разработчики сталкивались с ситуациями, когда классы в самой последней версии инструментария SDK менялись или вовсе исключались из инструментария. Например, некоторые приложения, разработанные с использованием инструментария Android SDK 1.5, прекращали свое функционирование после выхода операционной системы Android 1.6, а затем снова начинали функционировать в операционной системе Android 2.0. Этот факт может сильно разочаровать разработчиков и пользователей.

ОПРЕДЕЛЕНИЕ ЦЕЛЕВОГО ИНСТРУМЕНТАРИЯ SDK ДЛЯ ПРОЕКТА

Вы можете указать, какая версия инструментария SDK поддерживаете» вашим приложением, скомпилировав приложение с использованием соответствующей версии инструментария SDK, которая определяется в настройках проекта, а также в файле манифеста Android. Вы также можете использовать определенные ресурсы приложения, которые предназначены для конкретных версий инструментария SDK, применив соответствующие квалификаторы при именовании каталогов с ресурсами, перечисленные в табл. 20.1.

ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЙ ДЛЯ ОБЕСПЕЧЕНИЯ ОБРАТНОЙ СОВМЕСТИМОСТИ

Чтобы ваше приложение могло поддерживать максимально возможное количество мобильных телефонов, вы должны разрабатывать приложение с учетом поддержки нескольких версий инструментария SDK. Однако, если вы укажете требуемые версии инструментария SDK в файле манифеста Android, это сузит диапазон версий инструментария SDK. на которых может быть установлено ваше приложение.

Для этой ситуации существует обходное решение. Поскольку в языке Java используется механизм отражения, вы можете обращаться к классам и методам, не включая их в инструкции `import`. Таким образом, вы могли бы указать в качестве минимальной версии инструментария SDK минимально возможную версию, которая может использоваться вашим приложением. Далее для расширения функциональности или возможностей приложения могла бы использоваться соответствующая бизнес-логика — путем определения на этапе выполнения доступных классов или методов. Этот подход также может быть использован для устройств, которые обладают специальными возможностями или функциями, отсутствующими на других устрой- сгвах, но при этом вы хотели бы использовать эти возможности в вашем приложении в тех случаях, когда они доступны.

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Отличный пример использования механизма отражения для поддержки нескольких версий инструментария Android SDK доступен по адресу **developer.android.com/resources/articles/backward-compatibility.html**.

ОПРЕДЕЛЕНИЕ ВЕРСИИ ИНСТРУМЕНТАРИЯ ANDROID SDK ПРОГРАММНЫМ ПУТЕМ

Вы можете определить версию платформы Android программным путем, используя класс `Build` (`android.os.Build`). В частности, вы можете проверить значение константы `SDK_INT` класса `Build.VERSION` – возможные значения этой константы определены в классе `android.os.Build.VERSION_CODES`.

СОЗДАНИЕ РЕСУРСОВ ПРИЛОЖЕНИЯ ДЛЯ КОНКРЕТНОЙ ВЕРСИИ ИНСТРУМЕНТАРИЯ ANDROID SDK

Во многом подобно тому, как разработчики могут создавать в приложениях ресурсы для конкретного языка, региона и вариантов конфигурации мобильного устройства, они также могут создавать ресурсы для конкретных версий инструментария Android SDK. Если вернуться к табл. 20.1, вы увидите, что названия каталогов с ресурсами могут также содержать идентификатор API Level конкретной версии инструментария Android SDK.

ИТОГИ

В этом часе вы узнали, как можно адаптировать ресурсы приложения к широкому спектру конфигураций мобильных устройств, включая аппаратные и программные возможности. Вы также познакомились с принципами разработки приложений, корректно функционирующих при смене ориентации экрана. Наконец, вы узнали, как разрабатывать приложения для различных версий инструментария Android SDK.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Обновление операционной системы мобильного устройства привело к неработоспособности моего приложения. Что можно сделать, чтобы исправить эту проблему?

Ответ: Во-первых, вытрите слезы и удалите то гневное письмо, которое вы собрались отправить разработчикам операционной системы Android. Ситуации подобного рода вызывают беспокойство и могут даже поставить вас в неловкое положение, но такое случается — и довольно часто. В некоторых случаях вы можете избежать подобных сюрпризов, протестировав ваше приложение с использованием выпускаемой в скором времени версии инструментария Android SDK, исходный код которой есть в свободном доступе (Android Open Source Project), однако нет никакой гарантии, что некий производитель мобильного телефона (или оператор сотовой связи) не изменит эту версию инструментария, добавив или удалив определенные возможности по своему желанию. Иногда вы не будете знать о существовании некоторой проблемы до тех пор, пока вам не сообщит об лом пользователь откуда-нибудь из Сан-Паулу или Пекина или, что, возможно, еще страшнее, ваш босс. Для таких ситуаций у вас должен быть выработан четкий план действий. Заранее назначьте человека (беднягу), который будет следить за выходом новых версий инструментария Android SDK, чтобы оперативно исправлять ошибки и публиковать обновления приложения для пользователей.

Вопрос: Каким образом я могу получать события изменения ориентации экрана и загружать соответствующий макет для портретного или альбомного режима, чтобы экраны моего приложения всегда выглядели красиво?

Ответ: Для этого вашему приложению нет необходимости получать события изменения ориентации экрана. Вместо этого убедитесь, что у вас есть ресурсы макета, которые хранятся в каталоге с правильно определенным названием (с использованием квалификатора **port** или **land**). Операционная система Android автоматически загрузит подходящий макет при изменении ориентации экрана устройства. Как и в случае с любым другим ресурсом, вы должны убедиться в том, что ресурсы макетов для портретного и альбомного режимов включают одинаковые дочерние представления (чтобы избежать ситуаций, когда используемое представление отсутствует в макете для одной из ориентаций).

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Квалификаторы в указанном названии каталога с ресурсами перечислены в правильном порядке: **/res/drawables-rUS-en**.

2. Для каких из следующих конфигураций мобильного устройства могут быть определены ресурсы?

A. Язык и регион/локаль.
B. Способы ввода информации, например клавиатуры, сенсорные экраны и навигационные клавиши.
C. Размеры экрана, разрешение и ориентация.

D. Доступность клавиатуры и навигационных клавиш.
E. Всего вышеперечисленного.

Ответы

Неверно. Регион должен указываться после языка. Таким образом, правильное название каталога будет выглядеть следующим образом: **/res/drawables-en-rUS**.

E. Для всех перечисленный конфигураций доступны соответствующие квалификаторы, которые могут использоваться для определения ресурсов приложения. Кроме того, существуют и другие квалификаторы. Полный список квалификаторов можно найти на веб-сайте, посвященном разработке на платформе Android, по адресу **developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources**.

Упражнения

1. Протестируйте приложение «Been There, Done That!» на мобильном телефоне. Перейдите в альбомный режим и убедитесь, что все экраны приложения отображаются корректно. При необходимости добавьте ресурсы макета для альбомного режима (по крайней мере, это нужно сделать для экрана-заставки и игрового экрана).

2. Скомпилируйте приложение «Been There, Done That!» с использованием различных целевых версий инструментария SDK и протестируйте приложение на мобильных телефонах или на эмуляторе с соответствующим AVD-профилем.

Час 21. БОЛЕЕ ГЛУБОКОЕ ЗНАКОМСТВО С ПЛАТФОРМОЙ ANDROID

Вопросы, рассматриваемые в этом часе:

- **рассмотрение дополнительных возможностей платформы Android;**
- **разработка сложных пользовательских интерфейсов;**
- **работа с мультимедиа;**
- **хранение и предоставление общего доступа к данным;**
- **взаимодействие с нижележащим аппаратным обеспечением устройства.**

При знакомстве с новой платформой для мобильных устройств знание возможностей этой платформы может оказаться весьма полезным. Этот час представляет собой экспресс-курс по некоторым из дополнительных возможностей инструментария Android SDK. В частности, вы узнаете об использовании базовых возможностей приложений, разработке сложных пользовательских интерфейсов, использовании мультимедиа, управлении хранилищами и данными, а также о взаимодействии с нижележащим аппаратным обеспечением устройства.

РАССМОТРЕНИЕ ДРУГИХ БАЗОВЫХ ВОЗМОЖНОСТЕЙ ПЛАТФОРМЫ ANDROID

Двадцать четыре часа — это, безусловно, очень короткий промежуток времени, чтобы успеть рассказать обо всех интересных и полезных возможностях платформы Android и инструментария Android SDK. В процессе знакомства с этой книгой вы уже добавили множество возможностей в приложение «Been There, Done That!». В этом часе вы познакомитесь со многими другими дополнительными возможностями платформы Android.

Объявление разрешений, необходимых для использования приложения

Как вы знаете, приложения должны указывать требуемые для их функционирования разрешения в файле манифеста Android. Приложения также могут объявлять и заставлять использовать свои собственные разрешения при помощи тега <permission>. Каждое разрешение должно быть определено в файле манифеста Android и может применяться к отдельным компонентам -- в основном, к деятельностям или к службам — внутри приложения Кроме тою. разрешения также могут применяться на уровне методов.

Предупреждение пользователя при помощи уведомлений

Приложение может предупреждать пользователя о некоторых событиях даже в тех случаях, когда оно не активно, при помощи уведомлений. Например, приложение, предназначенное для обмена сообщениями, может уведомлять пользователей, когда приходит новое сообщение, как показано на рис. 21.!.
Уведомления могут принимать различную форму. Приложение может использовать несколько различных типов уведомлений при условии, что оно имеет соответствующие разрешения, указанные в файле манифеста Android:

- отображение текстового уведомления в строке состояния;
- воспроизведение звука;

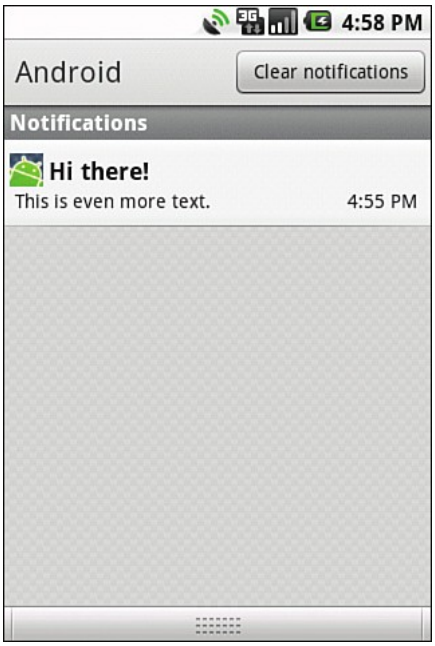

Рис. 20.1. Уведомление в строке состояния

- вибрация устройства;
- изменение цвета светового индикатора и частоты его мигания

ВНИМАНИЕ!
Не все устройства поддерживают все перечисленные типы уведомлений. Например, некоторые устройства могут не иметь возможности вибрировать или воспроизводить

звуки, а у некоторых может отсутствовать световой индикатор.

Уведомления создаются и отображаются при помощи системной службы, представляемой классом `NotificationManager` (`android.app.NotificationManager`). Получив экземпляр класса для данной службы, вы можете создать объект типа `Notification` (указав подходящий текст уведомления, а также настройки вибрации, светового индикатора и звука) и использовать метод `notify()` класса `NotificationManager`, чтобы представить созданное уведомление пользователю.

ВНИМАНИЕ!
Вы должны проявлять крайнюю осторожность при использовании уведомлений, чтобы не досаждать пользователю. Некоторые типы уведомлений, например вибрация устройства, должны быть протестированы на реальном устройстве, поскольку эмулятор Android не позволяет имитировать этот тип действия.

РАЗРАБОТКА СЛОЖНЫХ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Самые лучшие и наиболее популярные приложения на платформе Android имеют одну общую деталь: каждое из них обладает прекрасным, хорошо спроектированным пользовательским интерфейсом. Вы работали со многими базовыми возможностями платформы Android, относящимися к пользовательскому интерфейсу, например с макетами и элементами пользовательского интерфейса. Тем не менее инструментарий Android SDK обладает множеством других потрясающих возможностей для разработки пользовательских интерфейсов, включая следующие:

- возможность применять единообразные настройки к множеству элементов или к целым экранам при помощи стилей и тем;
- возможность разрабатывать и повторно использовать собственные компоненты пользовательского интерфёйса;
- мощная платформа методов ввода;
- возможность определять разнообразные движения пальцев пользователя по экрану;
- движок преобразования текста в речь (text-to-speech — TTS);
- поддержка распознавания речи.

Использование стилей и тем

Инструментарий Android SDK предоставляет два мощных механизма для разработки согласованных пользовательских интерфейсов: стили и темы. Вы можете использовать темы и стили, чтобы придать экранам приложения единообразный вид и упростить их сопровождение.

Стиль — это группа настроек распространенных атрибутов представления `View`, которая может примениться к любому числу элементов `View`. Например, ны можете захотеть сделать так, чтобы все элементы `View` в вашем приложении, такие как элементы

TextView и EditText, использовали один и тот же цвет текста, гарнитуру и размер. В этом случае можно создать стиль, который определяет ли три атрибут, и применить ею к каждому элементу TextView и EditText, входящему в состав макетов вашего приложения.

Тема - это коллекция, состоящая из одного или более стилей. Если стиль применяемся к конкретному элементу пользовательского интерфейса, например, к элементу TextView, тему вы применяете ко всем объектам типа View, относящимся к определенной деятельности. Применение темы сразу ко всем объектам типа View упрощает создание согласованных пользовательских интерфейсов; отличным вариантом может оказаться определение цветовых схем и других настроек распространенных атрибутов представления View для всего приложения. Вы можете указать тему программным путем, вызвав метод setTheme() класса Activity. Вы также можете применить темы к определенной деятельности при помощи файла манифеста Android.

ЗНАЕТЕ ЛИ ВЫ ЧТО...

Инструментарий Android SDK включает ряд встроенных тем, которые доступны в классе android.R.style. Например, класс android.R.style.Theme представляет системную тему, используемую по умолчанию. Также существуют темы с черным фоном, темы с отображением и без отображения строк заголовков приложений, темы для элементов управления диалоговых окон и многие другие.

Разработка пользовательских элементов View и ViewGroup

Вам уже знакомы многие элементы пользовательского интерфейса, например, макеты и элементы View, которые доступны в инструментарии Android SDK. Вы также можете создавать собственные элементы пользовательского интерфейса. Для этого вы можете просто взять подходящий элемент View (или ViewGroup) из пакета android.view и реализовать специфическую функциональность, необходимую дня вашего элемента или макета.

Вы можете использовать пользовательские элементы View в XML-файлах макетов или создавать их программным путем во время выполнения приложения. Вы можете создавать новые типы элементов пользовательского интерфейса или просто расширять функциональность существующих элементов, например элементов TextView или Button.

Дополнительную информацию по реализации пользовательских элементов View можно найти по адресу **developer.android.com/guide/topics/ul/custom-components.html**.

Использование различных методов ввода

Платформа Android представляет удобную программную клавиатуру (изображенную на рис. 21.2) для устройств, не имеющих аппаратных клавиатур. Кроме того, инструментарий Android SDK также включает мощную поддержку методов ввода текста с предикативными возможностями и загружаемые редакторы методов ввода (input method editors - IME).

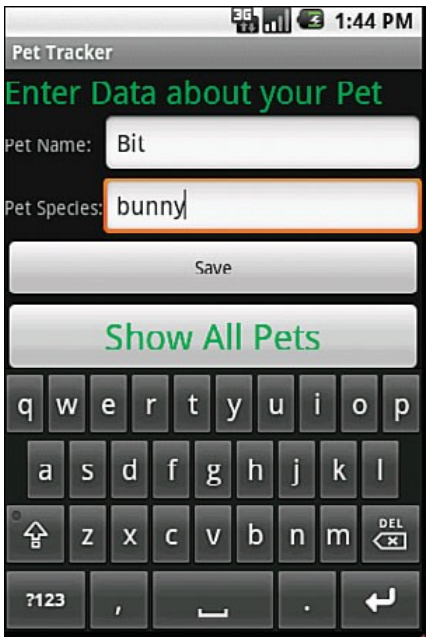

Рис. 21.2. Программная клавиатура Android

Обработка движений пальцев пользователя по экрану

Вы уже знаете, как получать события нажатий. Вы также можете определять движения пальцев пользователя по экрану, например быстрые движения, перемещения и нажатия, при помощи класса `GestureDetector`(`android.view.GestureDetector`). Вы можете использовать класс `GestureDetector` реализовав `onTouchEvent()` для нужной деятельности.

Ниже перечислены некоторые события, связанные с движениями пальцев пользователя но экрану, которые может распознавать и обрабатывать приложение:

`onDown`— происходит, когда пользователь в первый раз коснулся сенсорного экрана.

`onShowPress` — происходит, когда пользователь в первый раз коснулся сенсорного экрана, но перед тем, как пользователь убрал палец или перемести его по экрану.

`onSignleTapUp` — происходит, когда пользователь убирает палец с сенсорного экрана, как часть события одиночного нажатия.

`onSignleTapConfirmed` — вызывается при возникновении события одиночного нажатия.

`onDoubleTap` — вызывается, когда возникает событие двойного нажатия.

`onDoubleTapEvent` — вызывается, когда при двойном нажатии происходит дополнительное событие, включая движение вниз, перемещение или движение вверх.

`onLongPress` — аналогично событию `onSingleTapUp`, но вызывается, когда пользователь удерживал свой палец достаточно длительное время, чтобы это движение не считалось стандартным нажатием, и при этом не перемещал палец.

`onScroll` — вызывается после того, как пользователь нажал на экран и переместил свой палец с постоянной скоростью, после чего убрал палец с экрана.

`onFling` — вызывается после того, как пользователь нажал на экран и переместил свой палец с ускорением перед тем, как убрать палец с экрана

Кроме того, функциональность, предоставляемая пакетом `android.gesture`, позволяет приложению распознавать произвольные движения пальцем, а также сохранять, загружать и отображать их на экране. Это значит, что почти любой символ, нарисованный пользователем на экране, можно преобразовать в образ движения с определенным значением. Некоторые версии инструментария SDK включают приложение `GestureBuilder`, которое позволяет упростить процесс создания образов движений для приложений, не имеющих возможности самостоятельной записи движений пальцем пользователя.

Дополнительную информацию по пакету android.gesture можно найти 110 адресу **developer.android.com/resources/articles/gestures.html**.

Преобразование текста в речь

Платформа Android включает TTS-движок (android.speech.tts), который позволяет мобильным устройствам осуществлять синтез речи. Вы можете использовать TTS-движок для того, чтобы ваше приложение «читало» текст пользователю. Эта возможность часто применяется в приложениях, использующих геолокационные (Location-Based Services, LBS) сервисы и позволяющих автоматически получать голосовую информацию о направлениях движения. В других приложениях эта возможность предназначена для пользователей, имеющих проблемы с чтением или со зрением.

TTS-движок платформы Android поддерживает несколько языков, включая английский (с американским и британским произношением), французский, немецкий, итальянский и испанский языки.

Синтезированную речь можно сразу воспроизвести или записать в аудиофайл, который ничем не отличается от любого другого аудиофайла.

ВНИМАНИЕ!

Для предоставления TTS-сервисов пользователям устройство Android должно не только иметь TTS-движок (доступный в инструментарии Android SDK 1.6 и более поздних версиях), но и соответствующие языковые файлы ресурсов. В некоторых случаях пользователь должен установить соответствующие языковые файлы ресурсов (при условии, что на устройстве пользователя достаточно свободно пространства для размещения этих файлов) из удаленного источника. Пользователи могут выполнить эту процедуру самостоятельно, запустив приложение **Settings** (Настройки) и затем выбрав в меню команду **Text-to-speech** => **Install Voice Data** (Преобразование текста в речь => Установка голосовых данных). Вам также понадобиться проделать эту операцию на ваших устройствах. Кроме того, приложение может проверить правильность установки голосовых данных и загустить процесс установки, если данные не установлены или установлены неправильно. Дополнительная информация — в документации по пакету `android.speech.tts`.

Преобразование речи в текст

Вы можете улучшить ваше приложение путем реализации поддержки возможности распознавания речи, используя соответствующую платформу (`android.speech.RecognizerIntent`). Вы используете этот интент, чтобы записать речь и отправить ее на сервер распознавания речи для последующей обработки, поэтому применять эту функциональность на устройствах, которые не имеют приемлемого подключения к Интернету, не совсем целесообразно.

ВНИМАНИЕ!
В инструментарии Android SDK 2.1 и более поздних версиях возможность распоз навания речи встроена в большинство программных клавиатур. Таким образом, приложение уже в некотором роде может поддерживать возможность распознавания речи без необходимости внесения каких-либо изменений. Тем не менее непосредственный доступ к распознавателю речи может позволить реализовать более интересное голосовое управление приложениями.

РАБОТА С МУЛЬТИМЕДИА

Мобильные устройства все больше и больше используются в качестве муль- тимедийных устройств. Многие устройства Android имеют встроенные камеры, микрофоны и громкоговорители, позволяя воспроизводить и записывать мультимедийные данные в различных форматах. Инструментарий Android SDK обеспечивает всестороннюю поддержку мультимедийных данных, позволяя разработчикам встраивать звуковые и визуальные элементы (изображения и видео) в приложения. Соответствующие интерфейсы API — это часть пакета `android.media`.

ВНИМАНИЕ!
Эмулятор Android не позволяет записывать аудио или видео. Тестирование воз- можностей записи аудио и видео должно осуществляться на реальном устройстве Android Кроме того, возможности записи отдельно взятого устройства могут зависеть от используемых аппаратных и программных компонент. Например, в устройствах Android, которые не являются телефонами, могут отсутствовать микрофоны и, что встречается гораздо чаще, камеры.

Воспроизведение и запись аудио

Инструментарий Android SDK предоставляет механизмы для воспроизведения и записи аудио в различных форматах. Аудиофайлы могут представлять собой ресурсы, локальные файлы или объекты типа URI, ссылающиеся на совместно используемые или сетевые ресурсы. Класс `MediaPlayer` (`android.media.MediaPlayer`) может быть использован для воспроизведения аудио, а класс `MediaRecorder` (`android.media.MediaRecorder`) — для записи аудио. Для записи аудио приложению требуется разрешение `android.permission.RECORD_AUDIO`.

Воспроизведение и запись видео

Вы можете использовать элемент пользовательского интерфейса `VideoView` для воспроизведения видеофайлов на экране. Элемент `MediaController` позволяет добавить к элементу `VideoView` основные элементы управления воспроизведением, например кнопки «Воспроизведение», «Пауза» и «Стоп» (рис. 21.3).

Как и в случае с записью аудио, вы можете использовать класс `MediaRecorder` для записи видео при помощи встроенной камеры. Приложения, обращающиеся к аппаратному обеспечению камеры, должны иметь разрешение `android.permission.camera`, а те из них, которые записывают аудио при помощи класса `MediaRecorder`, должны иметь разрешение `android.permission.REC0RD_AUDIO`, указанные в файле манифеста Android. Таким образом, чтобы записать видео с использованием микрофона и камеры, в файл манифеста Android должны быть добавлены оба разрешения.

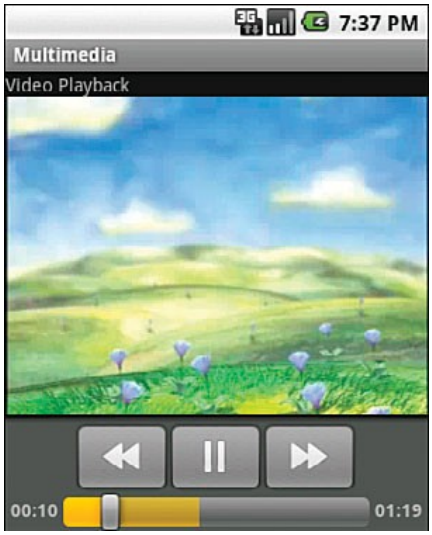

Рис. 21.3. Элемент пользовательского интерфейса VideoView с элементом MediaController

РАБОТА С ДВУХМЕРНОЙ И ТРЕХМЕРНОЙ ГРАФИКОЙ

Если вы знакомы с программированием компьютерной графики, вам будет приятно узнать, что платформа Android обладает довольно развитыми для мобильного устройства графическими возможностями.

Использование графических библиотек платформы Android

Инструментарий Android SDK поставляется вместе с пакетом `android.graphics`, который содержит ряд удобных классов для рисования на экране (рис. 21.4). К возможностям пакета `graphics` инструментария Android SDK можно отнести инструменты для работы с растровыми изображениями

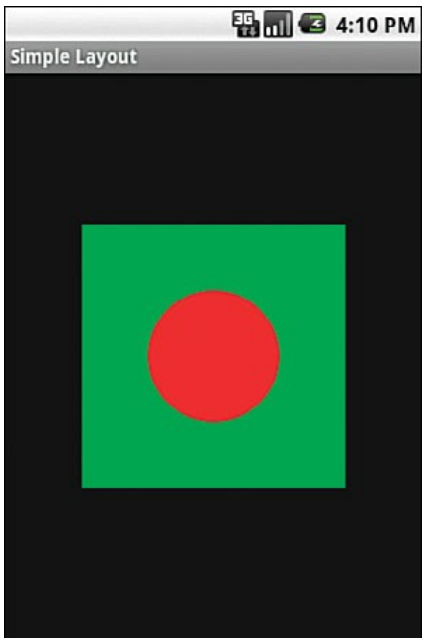

Рис. 21.4. Простое двухмерное изображение, созданное с использованием графической библиотеки платформы Android

Рис.21.5. Изображение, созданное при помощи интерфейса OpenGL ES на платформе Android

Использование графического интерфейса OpenGL ES

Для создания более сложной графики платформа Android использует популярный графический интерфейс OpenGL ES (версия 1.0), а также обеспечивает ограниченную поддержку интерфейса OpenGL ES версии 1.1. Приложения могут использовать поддержку интерфейса OpenGL ES на платформе Android для рисования, анимации, освещения, создания теней и текстуриро- вания графических объектов в трехмерном пространстве (рис. 21.5).

ПЕРСОНАЛИЗАЦИЯ УСТРОЙСТВ ANDROID

Персонализация мобильного устройства подразумевает предоставление пользователю возможности изменять внешний вид и поведение его рабочей среды. С точки зрения программного обеспечения, персонализация подразумевает настройку таких элементов, как обои, рингтоны и многое другое. Платформа Android обеспечивает высокую степень настройки и персонализации. Пользователь может устанавливать альтернативные домашние экраны, темы, изображения и звуки. Android-приложения могут предоставлять пользователям многие из этих возможностей персонализации. Например, приложения, выпущенные под некоторой торговой маркой, могут позволять пользователям устанавливать рингтоны и обои, поддерживающие эту торговую марку.

Установка рингтона

Приложение может изменять рингтон мобильного устройства, используя класс RingtoneManager. Чтобы приложение могло изменять рингтон, в файле манифеста Android

должно быть указано соответствующее разрешение (`android.permission.WRITE_SETTINGS`). Вы также можете запустить приложение, позволяющее выбрать рингтон, используя интент `ACTION_RINGTONE_PICKER`.

Установка обоев

Приложение может установить обои на заднем плане домашнего экрана при помощи класса `WallpaperManager`. Этот класс предоставляет различные методы для получения текущих обоев и для установки новых обоев, используя растровое изображение, ресурс или другой вид обоев.

Помимо использования статических изображений в качестве обоев, платформа Android поддерживает концепцию «живых» обоев. По существу, это анимированные обои, но которые могут содержать почти все. что приложение может нарисовать на некоторой поверхности. Например, можно создать обои, которые будут визуально представлять текущую погоду, время дня, информацию о воспроизводимом музыкальном треке, слайд-шоу, или определенную видео- или анимированную демонстрацию. «Живые» обои похожи на виджеты, однако детали их реализации отличаются.

Дополнительную информацию по обоям можно найти в документации инструментария Android SDK, связанной с пакетом `android.service.wallpaper`.

Создание «живых» обоев

Помимо использования статических изображений в качестве обоев, платформа Android поддерживает концепцию «живых» обоев. Вместо отображения статического изображения, например в JPEG-формате, «живые» обои могут отображать все, что может быть нарисовано на поверхности с использованием всех графических возможностей устройства и инструментария Android SDK (как было рассмотрено в разделе, посвященном работе с двухмерной трехмерной графикой, ранее в этом часе).

«Живые» обои похожи на службы платформы Android. В результате их работы вы получаете поверхность, которую может отображать хост-процесс. Вы можете создавать «живые» обои любой сложности, однако необходимо принимать во внимание скорость реакции устройства на действия пользователя и ресурс батареи. Вот несколько примеров «живых» обоев:

- трехмерная анимированная сцена, на которой рисуются абстрактные фигуры;
- служба, которая в режиме слайд-шоу отображает изображения, хранящиеся в интернет-сервисе для обмена изображениями, с использованием анимированных переходов;
- интерактивный пруд с водой, на поверхности которого появляются волны от прикосновения пальца пользователя к экрану;
- обои, которые изменяются в зависимости от поры года, погоды и времени дня.

КСТАТИ

Дополнительную информацию по созданию «живых» обоев можно найти в статье, т священной «живым» обоям, на сайте с документацией по платформе Android (**bit. ly/bngiaP**), а также можно взглянуть на пример приложения Cube Live Wallpaper, вводящего в состав инструментария Android SDK.

ХРАНЕНИЕ И ПРЕДОСТАВЛЕНИЕ ОБЩЕГО ДОСТУПА К ДАННЫМ

Вам уже знакомы несколько способов, при помощи которых приложения могут обеспечивать постоянное хранение данных.

- Приложения могут хранить простые, примитивные типы данных с использованием экземпляра класса `SharedPreferences` на уровне приложения и на уровне деятельности.

- Приложения могут хранить данные на удаленном сервере приложения.

Приложения также могут хранить и предоставлять общий доступ к данным, используя следующие подходы:

- Приложения могут использовать структуру файлов и каталогов на устройстве для хранения собственных файлов в любом формате.

- Приложения могут хранить структурированные данные в собственных базах данных SQLite.

- Приложения могут обращаться к данным других приложений, выступающих в роли контент-провайдеров.

- приложения могут предоставлять общий доступ к своим внутренним данным, выступая в роли контент-провайдеров.

Вы уже знаете, как работать с экземпляром класса `SharedPreferences` как хранить данные на сетевом сервере приложения, поэтому давайте рассмотрим другие способы хранения и предоставления общего доступа к данным

Работа с файлами и каталогами

Каждое Android-приложение имеет свои собственные каталоги и файлы. Вы можете использовать стандартный Java-пакет `java.io`, который обеспечивает поддержку операций ввода/вывода, для управления файлами и каталогами. Для хранения файлов Android-приложений используется стандартная иерархия каталогов в файловой системе Android. В файловой системе Android данные Android-приложения хранятся в следующем каталоге верхнего уровня:

/data/data/<Название пакета>/

Для хранения баз данных, настроек и файлов внутри каталога приложения верхнего уровня создается несколько специальных подкаталогов. При необходимости вы также можете создавать собственные каталоги и файлы, используя соответствующие методы объекта типа `Context` приложения. Ниже перечислено несколько важных методов класса `Context`, предназначенных для управления файлами и каталогами:

- openFileInput() — открывает файл приложения, находящийся в подкаталоге **/files**, для чтения.
- openFileOutput() — открывает файл приложения, находящийся в подкаталоге **/files**, для записи.
- deleteFile () — удаляет файл приложения из подкаталоге **/files**, используя имя этого файла.
- fileList() — позволяет получить список всех файлов, находящихся в подкаталоге **/files**.
- getFilesDir() — позволяет получить объект типа File для подкаталога **/files**.
- getCacheDir() — позволяет получить объект типа File для подкаталога **/cache**.
- getDir() — создает или получает объект типа File для подкаталога с указанным именем.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...
Вы можете просмотреть содержимое файловой системы Android (эмулятора или подключенного устройства), используя панель **File Explorer** (Проводник) перспективы DDMS.

Хранение структурированных данных в СУБД SQLIte

Android-приложения могут взаимодействовать с собственной локальной базой данных под управлением СУБД SQLite. Реляционные базы данных SQLite легковесны и файлоориентированны — идеальный вариант для мобильных устройств. Инструментарий Android SDK включает ряд полезных классов для работы с базами данных SQLite. Классы, обеспечивающие поддержку баз данных SQLite на платформе Android, доступны в пакете `android.database.sqlite`. В частности, вы найдете вспомогательные классы для выполнения следующих операций:

- создание, контроль версий и управление базами данных;
- построение правильных SQL-запросов;
- обработка результатов выполнения запросов с использованием объектов типа `Cursor`;
- выполнение транзакций в базе данных;
- обработка специализированных исключений, связанных с использованием базы данных.

ЗНАЕТЕ ЛИ ВЫ ЧТО...
Инструментарий Android SDK имеет встроенную поддержку СУБД SQLite. Тем не

менее в пакете `android.database` вы также найдете общие классы для работы с базами данных.

Помимо возможности создания и использования баз данных SQLite программным путем, разработчики могут использовать для отладочных целей инструмент командной строки `sqlite3`, который доступен через командую оболочку ADB.

Предоставление доступа к данным из других приложений

Приложение может использовать данные, доступные в других Android- приложениях, если они предоставляют доступ к своим данным, выступая в роли контент-провайдеров. Вы также можете позволить вашему приложению предоставлять доступ к своим данным из других приложений, превратив его в контент-провайдера.

ИСПОЛЬЗОВАНИЕ КОНТЕНТ-ПРОВАЙДЕРОВ

В состав платформы Android входит ряд полезных приложений — например, приложение для работы с контактами и браузер, — которые предоставляют доступ к части или ко всем своим данным, выступая в роли контент- провайдера. Другое приложение может обращаться к данным этих приложений, используя интерфейс контент-провайдера для работы с данными. Некоторые контент-провайдеры предоставляют доступ к данным только в режиме чтения, а другие приложения позволяют создавать, обновлять и удалять записи, как, например, приложение для работы с контактами.

В большинстве случаев доступ к контент-провайдерам осуществляется в виде запросов к конкретному предопределенному объекту типа URf, содержащему адреса. После того, как запрос будет сформирован, результатом его выполнения может быть список контактов или пропущенных вызовов или же конкретная запись, например вся контактная информация для Ивана Петрова. Приложения могут обращаться к интерфейсам контент-провайдеров во многом так же, как они обращаются к любой базе данных.
Объект типа URI можно рассматривать как адрес местоположения, где существует необходимый контент. Вы можете использовать метод тападеаС-иегу*()* для получения данных от контент-провайдера и затем последовательно обработать результаты запроса при помощи курсора, точно так же, как это происходит при обработке результатов запроса к базе данных.

РАССМОТРЕНИЕ НЕКОТОРЫХ НАИБОЛЕЕ ЧАСТО ИСПОЛЬЗУЕМЫХ КОНТЕНТ-ПРОВАЙДЕРОВ

Контент-провайдеры, доступные на платформе Android, можно найти в пакете `android.provider`. Вот несколько наиболее полезных контент-провайдеров:

- `MediaStore` — применяется для доступа к мультимедийным данным (аудио, видео и изображениям) на телефоне и на внешних устройствах хранения информации;

- `CallLog` — применяется для доступа к информации о набранных, входящих и пропущенных телефонных вызовах;

- `Browser` — применяется для доступа к журналу посещенных пользователем веб-страниц и к веб-сайтам, добавленным в закладки;

- `Contacts` — используется для доступа к базе данных контактов пользователя;

- `UserDictionary` — словарь определенных пользователем слов для использования в функции предикативного ввода текста.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Вы можете связать данные из курсора базы данных или курсора контент-провайдера непосредственно с элементами пользовательского интерфейса `View`, например с элементами `ListView`. Для этого используйте один из элементов `Adapter`, например элемент `ArrayAdapter` или `CursorAdapter`, и элемент пользовательского интерфейса `View`, унаследованный от класса `AdapterView`, например элемент `ListView` или `Spinner`.

ПРЕДОСТАВЛЕНИЕ ДАННЫХ В КАЧЕСТВЕ КОНТЕНТ-ПРОВАЙДЕРА

Приложение может предоставлять доступ к своим внутренним данным другим приложениям, выступая в роли контент-провайдера. Чтобы предоставлять информацию другим приложениям, приложение должно реализовать интерфейс контент-провайдера и зарегистрироваться в качестве контент-провайдера в файле манифеста Android.

ОРГАНИЗАЦИЯ КОНТЕНТА С ИСПОЛЬЗОВАНИЕМ «ЖИВЫХ» ПАПОК

«Живая» папка - по особый тип объекта, при нажатии на который отображаются данные из приложения, выступающего в роли контент-провайдера. Например, музыкальное приложение может предоставлять пользователю возможность создания «живых» папок для определенных списков воспроизведения музыкальных композиций, которые могут быть помещены на домашний экран (для создания «живой» папки нажмите и удерживайте на любой точке домашнего экрана и в появившемся меню выберите команду **Folders** (Папки)). Чтобы создать «живую» папку, приложение должно создать класс `Activity`, который будет отвечать на интент-действие `ACTION_CREATE_LIVE_FOLDER`, и должно иметь соответствующий объект типа `ContentProvider`, предоставляющий содержимое для «живой» папки. Дополнительную информацию можно найти в документации по пакету `android.provider.LiveFolders`.

Интеграция с механизмом глобального поиска

Операционная система Android позволяет приложениям делать свои данные доступными для поиска на уровне системы. Для этого необходимо соответствующим образом настроить приложение и предоставить пользовательские классы Activity, принимающие различные команды, которые требуются для обработки поисковых запросов и результатов поиска. Кроме того, приложения могут предоставлять варианты окончания поискового запроса, которые будут отображаться при вводе пользователем критериев поиска в соответствующее поле ввода (поле ввода **Quick Search Box** (Окно быстрого поиска)).

Если ваше приложение имеет богатое информационное наполнение, которое либо создается пользователями, либо предоставляется вами, как разработчиком, интеграция

этого приложения с механизмом глобального поиска операционной системы Android может обеспечить множество преимуществ для пользователя и повысить ценность вашего приложения. Данные приложения становятся частью общей рабочей среды мобильного устройства, оказываются более доступными, и ваше приложение сможет попадаться на глаза пользователю гораздо чаще, чем когда пользователь сам запускает приложение.

КСТАТИ

Информацию о том, как встраивать функциональность глобального поиска в Android-приложения, можно найти в документации по классу `SearchManager` (`android.app.SearchManager`), а также можно взглянуть на пример приложения `Searchable Dictionary`, входящего в состав инструментария Android SDK.

ВЗАИМОДЕЙСТВИЕ С НИЖЕЛЕЖАЩИМ АППАРАТНЫМ ОБЕСПЕЧЕНИЕМ УСТРОЙСТВА

Разработчики на платформе Android имени беспрецедентный доступ к нижележащему аппаратному обеспечению устройства. Помимо такого аппаратного обеспечения, как камера и LBS-сервисы, инструментарий Android SDK предоставляет ряд интерфейсов API для доступа к низкоуровневым возможностям аппаратною обеспечения мобильного телефона, включая следующее:

- чтение исходных данных с датчиков (например, с магнитною датчика и датчика ориентации);
- доступ к модулям Wi-Fi и Bluetooth;
- мониторинг использования батареи устройства и управление параметрами питания.

ВНИМАНИЕ!

Не все датчики и аппаратное обеспечение доступны на каждом устройстве Android.

Датчики, доступные на каждом отдельном устройстве, могут отличаться с точки зрения доступности и чувствительности. Некоторые датчики непосредственно предоставляют исходные данные, а другие работают вместе со службами или программным обеспечением, снабжая приложение полезными данными.

Чтение исходных данных с датчика

Вот несколько датчиков устройства, поддерживаемых инструментарием Android SDK:
Акселерометр — измеряет ускорение в трех плоскостях.

- **Датчик освещенности** — измеряет освещенность (этот показатель полезно использовать для определения необходимости включения вспышки камеры).
- **Датчик магнитного поля** — измеряет магнетизм в трех плоскостях.
- **Датчик ориентации** — определяет ориентацию устройства.

- **Температурный датчик** — измеряет температуру.

- **Датчик расстояния** — измеряет расстояние от устройства до некоторой точки в пространстве.

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Эмулятор Android не имеет встроенной поддержки имитации датчиков устройства, однако компания Openintents предоставляет удобный эмулятор датчиков (**www.openintents.org/en/node/23**). Этот инструмент имитирует акселерометр, компасный датчик и датчик ориентации, а также температурный датчик, и передает данные в эмулятор. Вы также можете протестировать функциональность, относящуюся к датчикам, на целевом устройстве.

Для сбора данных от датчиков устройства используется объект типа `SensorManager`. Вы можете получить экземпляр класса `SensorManager` при помощи метода `getSystemService()`.

Работа с модулем Wi-Fi

Приложения, имеющие соответствующие разрешения (`ACCESS_WIFI_STATE` и `CHANGE_WIFI_STATE`) могут обращаться к встроенному модулю Wi-Fi на устройстве, используя объект типа `WifiManager`. Вы можете получить экземпляр класса `WifiManager` при помощи метода `getSystemService()`.
Инструментарий Android SDK предоставляет ряд интерфейсов API для получения информации о доступных для устройства сетях Wi-Fi, а также о свойствах подключения к сети Wi-Fi. Эта информация может быть использована для отслеживания уровня сигнала, поиска точек доступа или выполнения некоторых операций при подключении к определенным точкам доступа.

ВНИМАНИЕ!

В эмуляторе Android отсутствует поддержка модуля Wi-Fi, поэтому вы должны выполнять все тестирование интерфейса API для работы с сетью Wi-Fi на реальном устройстве.

Работа с модулем Bluetooth

Инструментарий Android SDK содержит классы для работы с модулем Bluetooth, которые доступны в пакете `android.bluetooth`. В этом пакете вы найдете классы для поиска устройств с включенным модулем Bluetooth и получения списка связанных устройств, а также классы для осуществления передачи данных.

Управление параметрами питания и временем работы батареи

Большинство мобильных устройств в основном работают от батареи питания. Чтобы контролировать заряд батареи, приложение должно иметь разрешение `BATTERY_STATS`, зарегистрироваться для получения интента `Intent.ACTION_BATTERY_CHANGED` типа `BroadcastIntent`, а также реализовать класс `BroadcastReceiver` для получения информации о батарее и выпол -нении любых необходимых действий. Вот некоторые сведения о батарее и параметрах питания, которые может контролировать ваше приложение:

- наличие в данном устройстве батарее;
- исправность батареи, статус (состояние зарядки), напряжение и температура;
- уровень заряда батареи (в процентах) и соответствующий значок;
- подключено ли устройство к сети переменного тока или получает питание через шину USB.

Приложение может использовать информацию о состоянии питания устройства, чтобы управлять своим собственным потреблением питания. Например, приложение, которое обычно использует большое количество вычислительной мощности, может отключать некоторые возможности, потребляющие огромное количество энергии, в ситуации с зарядом батареи.

ИТОГИ

В этом часе вы узнали о некоторых дополнительных возможностях платформы Android. Вы познакомились с несколькими более сложными архитектурными компонентами Android-приложений, например, как могут использоваться службы и уведомления и как приложения могут объявлять собственные разрешения, которые требуются для обращения к этим приложениям. Вы узнали, как разрабатывать пользовательские интерфейсы с согласованным внешним видом при помощи стилей и тем. Теперь вам известно, какими мощными мультимедийными возможностями обладают устройства Android, включая возможность воспроизводить и записывать аудио и видео, и что вы можете разрабатывать приложения с активным использованием трехмерной графики, благодаря поддержке интерфейса OpenGL ES. Android-приложения могут воспользоваться преимуществами удобных баз данных SQLite, а также могут обмениваться данными с другими приложениями, либо обращаясь к приложениям, выступающими в роли контент-провайдеров, либо самостоятельно выступая в роли контент-провайдеров. Наконец, приложение может обращаться и взаимодействовать с множеством разнообразных датчиков на устройстве.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Какие мультимедийные форматы поддерживаются платформой Android?

Ответ: Различные устройства Android поддерживают разные форматы. Сама платформа поддерживает ряд базовых форматов, однако конкретные устройства при необходимости также могут расширять этот список. Полный список поддерживаемых форматов можно

найти в документации по платформе Android на странице **http://devetoper.android.com/guide/appendlx/medla- formats.html**.

Вопрос: Где я могу найти примеры кода по использованию дополнительных возможностей, рассмотренных в этом часе?

Ответ: Рассмотрение деталей реализации возможностей, обсуждаемых в этом часе, выходит за рамки данной книги. Тем не менее вопросам профессиональной разработки приложений на платформе Android посвящена наша другая книга Android Wireless Application Development. Вы также можете найти множество примеров из инструментария Android SDK на вебсайте, посвященном разработке на платформе Android, по адресу **https//developer.android.com**.

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Для обращения к контент-провайдерам Android- приложение всегда должно указывать необходимые разрешения в файле манифеста Android.

2. Какие мультимедийные возможности доступны на платформе Android?

A. Возможность воспроизводить аудио.
B. Возможность воспроизводить видео.
C. Возможность записывать аудио.
D. Возможность записывать видео.
E. Все вышеперечисленные.

3. Верно ли это? Инструментарий Android SDK позволяет обращаться к световому индикатору на устройстве Android.

4. Верно ли это? В этом часе рассматриваются все дополнительные возможности инструментария Android SDK, которые не были рассмотрены ранее в этой книге.

Ответы

1. Неверно. Для обращения к контент-провайдерам могут требоваться определенные разрешения. Однако это зависит от конкретного контент-провайдера. Обратитесь к документации по выбранному контент-провайдеру, чтобы выяснить, какие разрешения требуются для доступа к его интерфейсу.

2. E. Пакет android.media включает поддержку возможностей воспроизведения и записи аудио и видео в различных форматах. Различные устрой' ства Android имеют разное аппаратное обеспечение, поэтому вы должны проверять конкретные целевые устройства, чтобы убедиться, что они поддерживают мультимедийные возможности, необходимые вашему приложению.

3. Верно. Вы можете использовать класс Notification Manager для обращения к светодиодному индикатору на устройстве Android.

4. Неверно. Инструментарий Android SDK имеет множество других возможностей и нюансов. Кроме того, платформа Android стремительно обновляется и развивается. Различные ресурсы, блоги, статьи и руководства разработчиков можно найти на сайте **http://developer.android.com**. Также в нашем блоге, доступном по адресу **http://andraidbook.blogspot.com**, вы можете найти разнообразные советы, трюки, руководства и ссылки на другие ресурсы.

Упражнения

1. Подумайте над тремя различными способами применения локальной базы данных SQLite с целью улучшения приложения «Been There, Done That!».

2. Познакомьтесь с различными системными службами, которые могут быть запрошены при помощи метода `getSystemService()`. Эти службы определены в классе `android.content.Context`.

3. Проанализируйте приложение «Been There, Done That!» и определите три функциональные области, где можно было бы разработать и использовать пользовательские элементы `View`. Какие функции выполняли бы эти пользовательские элементы?

4. Многие Android-приложения внешне выглядят похоже, поскольку они используют стандартную тему, предоставляемую платформой. Добавьте определения тем в макеты экранов приложения «Been There, Done That!». В этом случае ваше приложение будет иметь уникальный согласованный внешний вид на всех устройствах Android, независимо от стандартной темы, используемой на этих устройствах.

Час 22. ТЕСТИРОВАНИЕ ANDROID-ПРИЛОЖЕНИЙ

Вопросы, рассматриваемые в этом часе:

- **лучшие практики тестирования приложений для мобильных телефонов;**
- **разработка платформы тестирования приложений для мобильных телефонов;**
- **решение других проблем, связанных с тестированием.**

Каждый разработчик приложений для мобильных телефонов мечтает создать «революционное приложение». Многие люди считают, что если они придумали какую-то потрясающую идею, то могут вздохнуть свободно. Однако это в корне неверно. Дело в том, что придумать потрясающую идею может каждый. А вот детально продумать эту идею, чтобы получить ясное представление о ее реализации, лаконично «донести» идею до пользователей, разработать интуитивный пользовательский интерфейс и быстро передать готовое приложение пользователям — до того, как это сделает кто-то еще — в этом и заключается основная проблема! Революционное приложение должно оптимально сочетать в себе все эти ингредиенты, однако плохая реализация потрясающей идеи не позволит вашему приложению стать революционным, поэтому перед тем, как опубликовать приложение, вы должны тщательно его протестировать. В этом часе вы узнаете, как выполнять тестирование приложений для мобильных телефонов различными способами.

ЛУЧШИЕ ПРАКТИКИ ТЕСТИРОВАНИЯ

Мобильные пользователи ожидают многого от современных приложений для мобильных телефонов. В частности, они ожидают, что устанавливаемые ими приложения стабильны, быстро реагируют на действия пользователя и безопасны. Стабильность означает, что приложение работает, и не приводит к аварийным сбоям и не выводит телефон пользователя из строя. Быстрая реакция на действия пользователя подразумевает, что телефон всегда реагирует на нажатия клавиш, а для информирования пользователя об операциях, выполняющихся продолжительное время, используются индикаторы хода выполнения процесса. Безопасность подразумевает, что приложение не будет злоупотреблять доверием пользователя, как намеренно, так и ненамеренно. Пользователи ожидают, что приложение имеет достаточно простой пользовательский интерфейс, и что приложение будет работать в режиме «24 часа и 7 дней в неделю» (особенно, когда речь идет о сетевых приложениях, взаимодействующих с сервером).

Может показаться, что пользователи ожидают слишком многого от приложения, которое стоит 0,99 доллара США, однако, если разобраться, разве их ожидания — что-то сверхъестественное? Мы так не думаем. Тем не менее, эти ожидания налагают серьезную ответственность на разработчика в плане тестирования и контроля качества приложения.

Независимо от того, состоит ваша команда из одного или сотни человек, любой проект, связанный с разработкой приложений для мобильных телефонов, может выиграть от хорошо организованного процесса разработки с продуманным планом тестирования. Вот несколько показателей качества, которые могут существенно улучшить процесс разработки:

- стандарты кодирования и рекомендации;
- регулярное выполнение сборки проекта для определенной версии приложения;
- система отслеживания дефектов вместе с отдельным процессом исправления обнаруженных дефектов;
- систематическое тестирование приложения (с использованием плана тестирования).

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Вы можете отдать приложение для тестирования стороннему исполнителю Однако имейте в виду, что успех любого проекта, выполняемого сторонним исполнителем очень сильно зависит от качества предоставленной ему документации (например функциональных спецификаций, сценариев использования).

Разработка стандартов кодирования

Когда у разработчиков на руках есть некоторые рекомендации и они следуют им, написанный код становится более согласованным и более простым для сопровождения. Создание ряда четких стандартов кодирования для разработчиков может помочь втолковать им некоторые важные требования, связанные с разработкой приложений для мобильных телефонов. Например, разработчики могут захотеть сделать следующее:

- обсудить и принять решение относительно общего для всех разработчиков способа обработки ошибок и исключений;

- выносить операции, занимающие продолжительное время или большой объем вычислительных ресурсов, за пределы основного потока, управляющего пользовательским интерфейсом;

- освобождать используемые объекты и ресурсы;

- практиковаться в разумном управлении памятью и в обнаружении утечек памяти;

- использовать ресурсы надлежащим образом. Например, не встраивать данные в код или в файлы макетов.

Регулярное выполнение сборки проекта для определенной версии приложения

Реализация воспроизводимого процесса сборки проекта для определенной версии приложения — неотъемлемая часть успешного проекта на платформе Android. Это особенно справедливо для любого приложения, для которого планируется поддержка несколько версий инструментария Android SDK, мобильных телефонов или языков. Для регулярного выполнения сборки проекта дня определенной версии приложения придерживайтесь следующих рекомендации:

- Используйте систему контроля исходного кода для отслеживания версий файлов проекта.

- Создавайте версии файлов проекта через регулярные интервалы времени и выполняйте сборку проекта.

- Проверяйте (путем тестирования), что каждая сборка работает именно так, как необходимо.

Существует много прекрасных систем контроля исходного кода для разработчиков, и многие системы, которые отлично подходят для традиционной разработки, могут быть использованы при разработке приложений для мобильных телефонов. Многие популярные системы контроля исходного кода — например, Perforce, Subversion и CVS — отлично работают со средой разработки Eclipse.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Из-за постоянно увеличивающейся скорости разработки проектов приложений для мобильных телефонов, в целом наиболее успешны стратегиями при разработке таких приложений итеративные подходы. Быстрое создание прототипа ддот разработчикам и специалистам по контролю качества отличные возможности для оценки приложения до того, как оно попадет к пользователям.

Использование системы отслеживания дефектов

Система отслеживания дефектов предоставляет способ организации и отслеживания обнаруженных ошибок в приложении, называемых дефектами, и в основном дополняет процесс исправления этих дефектов. Исправление дефекта обычно подразумевает устранение проблемы я проверки того, что данное решение будет работать в следующей версии системы.

Когда дело касается приложений для мобильных телефонов, дефекты могут принимать различные формы. Некоторые дефекты возникают на всех мобильных телефонах, а другие — только на конкретных мобильных телефонах. Функциональные дефекты, т. е. функциональные возможности приложения, которые работают неправильно, — это всего лишь один из типов дефектов. Вы не должны останавливаться только на поиске этих дефектов — вы должны полностью протестировать работоспособность приложения в операционной системе Android с точки зрения производительности, скорости реакции на действия пользователя, удобства использования и управления состояниями.

Разработка хороших планов тестирования

Чтобы определить, какие возможности и функциональность были реализованы правильно, тестировщики должны в большой степени полагаться на функциональную спецификацию приложения, а также на документацию по пользовательскому интерфейсу. Возможности и рабочие процессы приложения должны быть тщательно описаны на уровне экранов и в дальнейшем проверены на этапе тестирования. В больших командах нередки ситуации, когда существуют разночтения функциональной спецификации, в результате чего возникают различия между функционалом, реализованным разработчиком, и результатами, ожидаемыми тестировщиками. Подобные различия должны устраняться в процессе исправления дефекта.

Для тестирования Android-приложений в распоряжении тестировщиков есть множество инструментов. И хотя ручное тестирование — неотъемлемая часть данного процесса, в настоящее время существуют многочисленные возможности для включения в планы автоматизированного тестирования.

Планы тестирования должны покрывать различные области, включая следующие:

- **Функциональное тестирование** — этот тип тестирования позволяет убедиться в том, что все возможности и функции приложения работают надлежащим образом, как определено в функциональной спецификации приложения.

- **Системное тестирование** — этот тип тестирования позволяет убедиться в том, что программное обеспечение хорошо интегрируется с другой основной функциональностью телефона. Например, приложение должно правильно приостанавливать и продолжать свое выполнение, а также оно должно корректно реагировать на вмешательства со стороны операционной системы (например, входящие сообщения, вызовы, выключение телефона).

- **Тестирование клиент/сервер** — при тестировании приложений для мобильных телефонов, использующих сетевые возможности, зачастую предъявляются более строгие требования, чем при тестировании автономных приложений. Так происходит потому, что помимо функциональности «клиента», выполняющегося на мобильном телефоне, вы должны также проверить функциональность на стороне сервера.

- **Тестирование обновлений** — телефоны Android часто получают обновления системы, что приводит к необходимости обновлять приложение. Когда это возможно, выполняйте тестирование обновлений приложения как на стороне клиента, так и на стороне сервера, чтобы убедиться, что обноааения не вызывают проблем у пользователей.

- **Тестирование интернационализации** — этот тип тестирования позволяет на раннем этапе процесса разработки убедиться в поддержке интернационализации — главным образом, в языковой поддержке. Если приложение поддерживает несколько языков, скорее всего, вы столкнетесь с некоторыми проблемами, связанными с нехваткой свободного пространства на экране, а также с такими проблемами, как, например, форматирование дат.

- **Тестирование удобства пользования** — этот тип тестирования позволяет выявить любые части приложения, которые недостаточно привлекательны, или вызывают затруднения в навигации или использовании. Данный тип тестирования позволяет убедиться, что модель потребления ресурсов приложением соответствует целевой аудитории. Например, для любителей игр может оказаться приемлемым более быстрый разряд батареи в игре с большим количеством графики, однако офисные приложения без необходимости не должны вызывать чрезмерное потребление энергии.

- **Нагрузочное тестирование** — этот тип тестирования подразумевает использование инструментов отладки, входящих в состав инструментария Android SDK, для наблюдения за использованием памяти и системных ресурсов; кроме того, нагрузочное тестирование позволяет выявить «узкие» места в приложении, связанные с производительностью, а также обнаружить опасные утечки памяти и устранить их.

- **Аттестационное тестирование** — этот тип тестирования используется для изучения необходимых стандартов, лицензионных соглашений и условий использования, которым должно соответствовать приложение, и для проверки соответствия приложения этим стандартам.

- **Случайное тестирование** — приложение должно быть достаточно надежным, чтобы корректно реагировать на возникновение случайных и непредсказуемых событий. Мы все иногда забываем заблокировать наши телефоны, чтобы потом обнаружить, что на телефоне были нажаты случайные клавиши, запушены случайные приложения или сделаны ненужные звонки, хотя все это время телефон находился нашем кармане. Приложение должно элегантно реагировать на подобные типы событий. Иначе говоря, приложение не должно завершаться аварийным сбоем. Вы можете использовать инструмент Exerciser Monkey, входящий в состав инструментария Android SDK, чтобы провести стресс-тест вашего приложения.

МАКСИМИЗАЦИЯ ТЕСТОВОГО ПОКРЫТИЙ

И хотя добиться 100%-ного тестового покрытая нереально, наша цель — максимально полно протестировать приложение. Для этого вам, вероятнее всего, потребуется выполнять тесты как на эмуляторе, так и на целевых мобильных телефонах, и, возможно, вы захотите рассмотреть возможность применения методов ручного и автоматизированного тестирования.

Настройка среды тестирования

Не думайте, что приложения для мобильных телефонов проще тестировать просто потому, что они «меньше» настольных приложений или имеют меньше возможностей. Тестирование приложений для мобильных телефонов ставит перед тестировщиками множество уникальных проблем, особенно в плане настройки конфигурации.

ОПРЕДЕЛЕНИЕ СПИСКА И ПОЛУЧЕНИЕ ЦЕЛЕВЫХ МОБИЛЬНЫХ ТЕЛЕФОНОВ

Чем раньше вы определите список целевых мобильных телефонов для вашего приложения и чем раньше получите их в свое распоряжение, тем лучше. В некоторых случаях для этого бывает достаточно пойти в магазин и приобрести новый телефон с новым тарифным планом; в других случаях все может оказаться гораздо сложнее.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

У ряда компаний есть свои программы поддержки разработчиков со специальными лабораториями, оснащенными разнообразными телефонами. В этих лабораториях разработчики могут получить на время в свое распоряжение определенные мобильные телефоны — по почте, удаленно (через Интернет) или лично посетив лабораторию. Это позволяет разработчикам получать доступ к широкому кругу мобильных телефонов в различных сотовых сетях, и для этого им не нужно покупать каждый раз новый телефон. В некоторых лабораториях даже есть специалисты, которые могут помочь в решении проблем, возникающих на отдельных телефонах.

Для опытных образцов мобильных телефонов могут пройти месяцы, прежде чем у вас в руках окажется необходимое устройство, полученное напрокат от производителя или от оператора в рамках программ поддержки разработчиков. Сотрудничество с операторами сотовой связи в рамках программ получение телефонов на прокат и покупка телефонов в розничных сетях может приводить в расстройство, однако в некоторых случаях без этого не обойтись. Вы не должны ждать до последней минуты, подбирая необходимые устройства для тестирования.

ВНИМАНИЕ!

Нет никакой гарантии, что опытный образец мобильного телефона будет вести себя точно так же, как и серийная копия, которая со временем появится у потребителей. Зачастую ряд возможностей устройства в последнюю минуту «урезается», чтобы успеть подготовить устройства к назначенной дате выпуска.

ПРОБЛЕМА ФРАГМЕНТАЦИИ УСТРОЙСТВ

Одной из основных проблем, с которыми сталкиваются тестировщики приложений для мобильных телефонов, — это бурный рост количества новых устройств Android на рынке. Эта проблема, называемая фрагментацией мобильных телефонов, делает чрезвычайно сложной задачу отслеживания всех доступных устройств, на которых используются различные версии инструментария Android SDK и которые имеют экраны с различными размерами, различный функционал и аппаратное обеспечение (рис. 22.1).

Рис. 22.1. Фрагментация мобильных телефонов

ВЕДЕНИЕ БАЗЫ ДАННЫХ МОБИЛЬНЫХ ТЕЛЕФОНОВ

Использование базы данных отслеживания информации о мобильных телефонах, которая может применятся при разработке и тестировании приложений, а также в маркетинговых целях, - отличная идея. Подробная база данных может содержать информацию наподобие следующей:

- информация о мобильных телефонах (модели, функционал, версию инструментария Android SDK, особенности аппаратного обеспечения, например, имеет ли мобильный телефон камеру или аппаратную клавиатуру);

- какие мобильные телефоны есть в наличии (и где они находятся, если их приобрели или взяли напрокат, и т.д.);

- какие мобильные телефоны являются целевыми для каждого приложения;

- мобильные телефоны, для которых ваши приложения продаются лучше всего.

Тестирование на эмуляторе

Команда тестировщиков может оказаться не в состоянии подготовить сред* тестирования для каждого оператора сотовой связи в каждой стране где пользователи будут использовать разрабатываемое приложение. Бывают ситуации, когда использование эмулятора Android позволяет со1фзтить накладные расходы и улучшить тестовое покрытие. Вот некоторые из преимуществ использования эмулятора:

- оперативное тестирование приложения, когда целевой мобильный телефон недоступен (или оказывается в дефиците);

- эмуляция мобильных телефонов, которые еще не появились на рынке (например, опытные образцы телефонов);

- тестирование сложных или опасных сценариев, которые невозможно или не рекомендуется проверять на реальных мобильных телефонах (например, тесты, которые каким-либо образом мот вывести телефон из строя или нарушить условия соглашения с оператором сотовой связи)

ВНИМАНИЕ!
Эмулятор предоставляет полезную, но весьма ограниченную эмуляцию типового устройства Android. Используя настройки AVD-конфигурзции, вы можете настроить эмулятор таким образом, чтобы он как можно точнее представлял целевой мобильный телефон. Тем не менее эмулятор не может в точности воспроизвести ту же аппаратную — или программную — реализацию, которая будет присутствовать на реальном мобильном телефоне. Эмулятор просто имитирует реальное устройство. Чем большее число аппаратных возможностей используется в приложении (например осуществление звонков, работа с сетью, использование LBS-сервисов, камеры и Bluetooth-соединений), тем важнее протестировать приложение на реальном устройстве.

Тестирование на целевых мобильных телефонах

Вот мантра для тестирования приложений, разрабатываемых для мобильных телефонов, которую следует повторять регулярно: «Приступать к тестированию как можно раньше, тестировать постоянно, тестировать на реальном устройстве».

Важно получить меленые мобильные телефоны в наше распоряжение как можно раньше. Следующая фраза как нельзя лучше описывает важность тестирования на мобильных телефонах: «Тестировать на эмуляторе полезно. однако тестирование на мобильном устройстве — это необходимость». На самом деле. го. что ваше приложение работает на эмуляторе, не значит практически ничего; пользователи будут запускать приложения на реальных мобильных телефонах.

ВНИМАНИЕ!
Важно проверять допущения, сделанные при проектировании приложения, как можно раньше и по несколько раз на целевом мобильном телефоне(ах). Этот процесс называется проверкой технической осуществимости. Ситуация, когда вы проектируете и разрабатываете приложение, а затем оказывается, что это приложение не работает на реальном мобильном телефоне, вызывает разочарование. Только то. что приложение работает на эмуляторе, не гарантирует, что оно будет работать на мобильном телефоне.

Тестирование на целевом мобильном телефоне — это самый верный способ убедиться в правильном функционировании приложения, поскольку вы выполняете приложение на том же аппаратном обеспечении, которое будет у ваших пользователей. Воспроизведя среду, в которой будут работать ваши пользователи, вы можете гарантировать, что в реальном мире приложение будет функционировать так, как задумано.

ВНИМАНИЕ!
Несмотря на то, что тестировать приложение на мобильном телефоне, подключенном к вашему компьютеру, может быть очень удобно, большинство пользователей будут

использовать ваше приложение иначе. В основном они будут использовать только батарею устройства. Вы должны отсоединить мобильный телефон от компьютера и протестировать приложение так, как его будет использовать большинство пользователей.

Выполнение автоматизированного тестирования

Сбор информации в приложении и создание автоматизированных тестов может помочь вам создать лучшее, «пуленепробиваемое» приложение. Инструментарий Android SDK предоставляет ряд пакетов, помогающих обнаруживать ошибки в коде. Способы обнаружения ошибок в коде можно разделить на две категории:

- журналирование информации в приложении для сбора статистики о его производительности и использовании;
- использование наборов автоматизироманных тестов построенных на базе среды тестирования JUnit.

ЖУРНАЛИРОВАНИЕ ИНФОРМАЦИИ В ПРИЛОЖЕНИИ

В начале этой книги вы узнали, как реализовать встроенный класс `Log` (`android.util.Log`), предназначенный для журналирования информации, чтобы реализовать различные уровни диагностического журналирования. Вы можете наблюдать за журналируемой информацией непосредственно из среды разработки Eclipse или при помощи утилиты поставляемой вместе с инструментарием Android SDK.

ВНИМАНИЕ!
Не забудьте убрать любые инструкции связанные с получением диагностической информации, например, инструкции журналирования, из приложения перед его публикацией. Инструкции журналирования и инструкции для получения диагностической информации могут отрицательно сказываться на производительности приложения.

АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СРЕДЫ ТЕСТИРОВАНИЯ JUNIT И СРЕДЫ РАЗРАБОТКИ ECLIPSE

Инструментарий Android SDK включает расширения среды тестирования JUnit для тестирования Android-приложений. Автоматизированное тестирование осуществляется путем написания сценариев тестирования на языке Java, которые проверяют, что приложение работает именно так. как предполагается. Данное автоматизированное тестирование может применяться как для модульного тестирования, так и для функционального тестирования, включая тестирование пользовательского интерфейса.

Этот раздел ни в коем случае не претендует на роль полной документации по написанию сценариев тестирования для среды JUnit. Данной теме посвящено множество книг и онлайн-ресурсов, включая сайт **www.junit.org**.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...
Некоторые люди следуют парадигме первоочередного создания сценариев тестирования, когда сначала создаются сценарии тестирования, а затем разрабатывается код, который должен успешно выполнятся для всех заранее созданных сценариев тестирования. Этот подход может отлично работать в ситуации, когда результаты выполнения

приложения и его поведения известны до начала работ по разработке приложения и в дальнейшем изменяются незначительно или не изменяются совсем.

Автоматизированное тестирование Android-приложений подразумевает выполнение нескольких простых шагов:

1) создание тестового проекта;
2) добавление сценариев тестирования в новый проект;
3) выполнение тестового проекта.

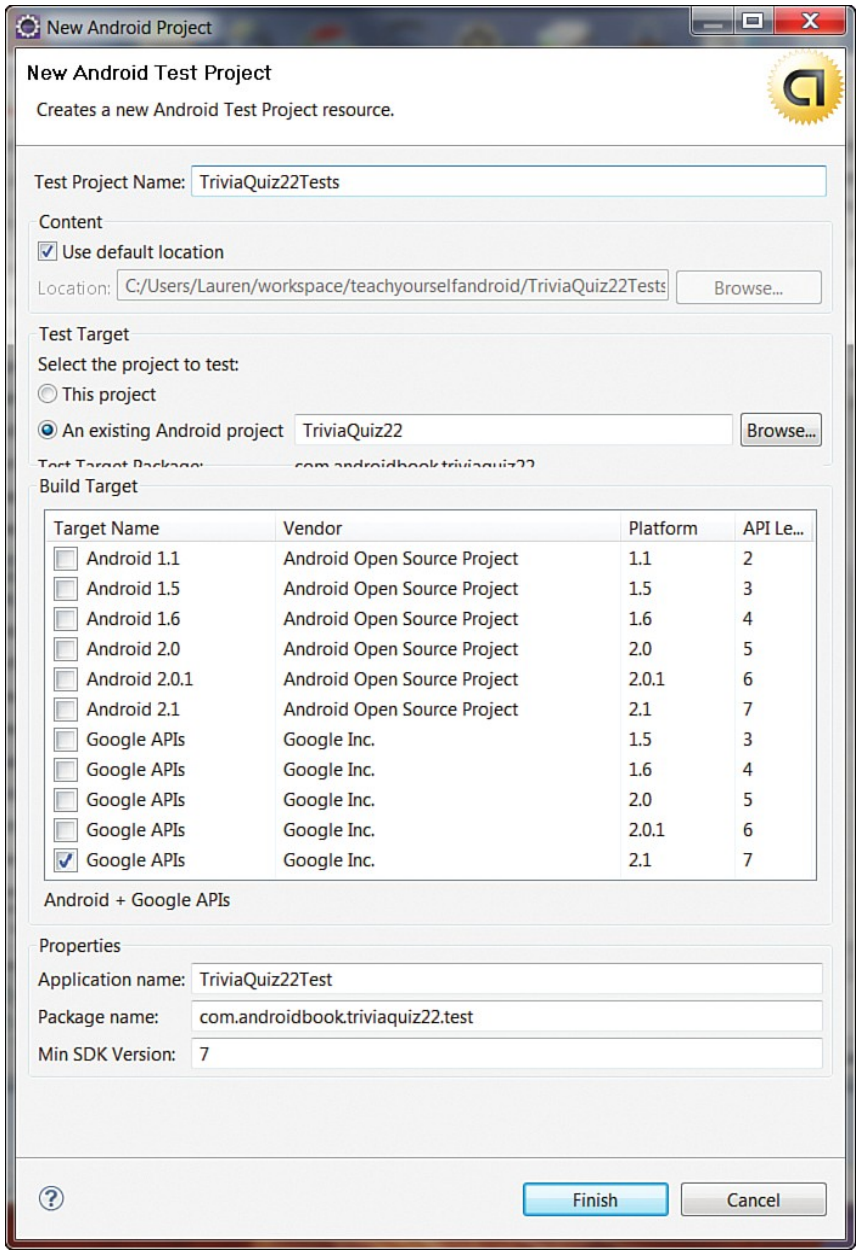

Рис. 22.2. Значения по умолчанию, установленные мастером создания текстового проекта в среде разработки Eclipse

В следующих разделах подробно рассматривается каждый из яих шагов на примере тестирования одной из возможностей экрана с настройками приложения «Веси There, Done That!».

СОЗДАНИЕ ТЕСТОВОГО ПРОЕКТА

Вспомните, как в часе 1, когда вы впервые создавали проект в среде разработки Eclipse, в мастере была доступна возможность создания тестового проекта. Теперь вы воспользуетесь этой возможностью, чтобы быстро добавить необходимый проект и приступить к созданию сценариев тестирования.

Удобно, что возможность создания тестового проекта также доступна и для уже существующих проектов. Чтобы создать тестовый проект для существующего Android-проекта в среде разработки Eclipse, выполните следующие шаги:

1. Выберите команду меню **File** => **New** => **Project** (Файл => Создать => Проект).

2. В появившемся диалоговом окне **New Project** (Новый проект) разверните элемент списка **Android** и выберите значение **Android Test Project** (Тестовый проект Android). Нажмите на кнопку **Next** (Далее).

3. В группе элементов управления **Test Target** (Цель тестирования) установите переключатель в положение **An Existing Android Project** (Существующий проект Android) и нажмите на кнопку **Browse** (Обзор).

4. В открывшемся диалоговом окне **Project Selection** (Выбор проекта) найдите проект, который вы хотите протестировать, и выберите его. Мастер автоматически заполнит оставшиеся поля ввода подходящими значениями по умолчанию, как показано на рис. 22.2.

5. Нажмите на кнопку **Finish** (Готово). В результате будет создан ваш новый тестовый проект.

СОЗДАНИЕ СЦЕНАРИЯ ТЕСТИРОВАНИЯ

Теперь, когда у вас есть тестовый проект, вы можете приступать к написанию сценариев тестирования. Вы создадите сценарий тестирования, который будет проверять поведение поля ввода **Nickname** (Ник) экрана с настройками, управляемого классом `QuizSettingsActivity`. Для этого сначала выполните следующие шаги, чтобы создать файл пустого сценария тестирования:

Щелкните правой кнопкой мыши по названию пакета внутри папки **src** вашего тестового проекта.

В открывшемся контекстном меню выберите команду **New** => **JUnlt Test Case** (Создать => Сценарий тестирования JUnit).

В ввода **Name** (Имя) появившегося диалогового окна **New JUnlt Test Case** (Новый сценарий тестирования JUnit) введите значение **QuizSettingsActivityTests**.

В пою ввода **Superclass** (Суперкласс) введите знамение **androld.test.ActivityInstrumentationTestCase2<QuizSettingsActivity>**. (Не обращайте внимания на предупреждение «Superclass does not exist* (Суперкласс не существует).)

В поле ввода **Class Under Test** (Тестируемый класс) введите значение **com.androidbook.triviaquiz22.QuizSettingsActivity**.

Нажмите на кнопку **Finish** (Готово).

В только что созданном файле вручную добавьте инструкцию `import` для класса `QuizSettingsActivity` (или организуйте ваши инструкции `import`)

Наконец, добавьте следующий конструктор в только что созданный класс:

```
public QuizSettingsActivityTests() {
   super("com.androidbook.triviaquiz22", QuizSettingsActivity.class);
}
```

Теперь, когда файл вашего сценария тестирования готов, вы можете протестировать функциональность поля ввода **Nickname** (Ник) и убедиться, что его значение совпадает со значением ника пользователя, которое хранится в экземпляре класса `SharedPreferences`, и что это значение обновляется после ввода новой строки. Сначала вы должны изменить метод `setup()`, чтобы выполнить некоторые общие операции. Вы получаете объект типа `EditText` для параметра **Nickname** (Ник), который будет использоваться в двух других тестах. Это делает следующий код:

```
@Override
protected void setUp() throws Exception {
    super.setUp();
    final QuizSettingsActivity settingsActivity = getActivity();
    nickname =
    (EditText) settingsActivity.findViewById(R.id.EditText_Nickname);
}
```

Вызов метода `getActivity()` позволяет получить деятельность, над которой будет выполняться тест. Деятельность создается внутри класса `ActivityInstrumentationTestCase2`, как если бы она запускалась в обычной ситуации.

Обычно также переопределяется метод `teardown()`. Однако для данных тестов вы не создаете никаких объектов, которые должны были бы удалится после выполнения тестов.

Тесты среды тестирования JUnit должны начинаться со слона **test**. Таким образом, при написании конкретных тестов вы должны содать мстолм, на звания которых начинаются со слова **test** и включают описание тестируемой операции. Сначала давайте убедимся, что значение, отображаемое в поле ввода **Nickname** (Ник), соответствует значению, хранящемуся в экземпляре класса `SharedPreferences`. Добавьте следующий кол в класс `QuizSettingsActivityTests`, чтобы реализовать данный тест:

```
public void testNicknameFieldConsistency() {
    SharedPreferences settings =
        getActivity().getSharedPreferences(QuizActivity.GAME_PREFERENC
        ES,
           Context.MODE_PRIVATE);
    String fromPrefs =
        settings.getString(QuizActivity.GAME_PREFERENCES_NICKNAME,
        "");
    String fromField = nickname.getText().toString();
    assertTrue("Field should equal prefs value",
        fromPrefs.equals(fromField));
}
```

Первые несколько строк представляют собой стандартный код Android- приложения, который должен быть вам уже знаком. Используя среду тестирования Android, вы можете использовать различные объекты Android- приложения в коде тестов. А вот в последней строке непосредственно выполняется тест. Метод `assertTrue()` проверяет, действительно ли второй параметр равен значению `true`. Если это не так, к результатам теста добавляется соответствующая строка. В данном случае сравниваются две строки. Они должны быть одинаковыми.

Следующий тест проверяет, что при изменении значения в поле ввода происходит обновление соответствующего значения в экземпляре класса `SharedPreferences`. Добавьте следующий код в класс `QuizSettingsActivity`, чтобы убедиться в справедливости данного утверждения:

```
private static final String TESTNICK_KEY_PRESSES = "T E S T N I C K
ENTER";
// ...
public void testUpdateNickname() {
    Log.w(DEBUG_TAG, "Warning: " +
        "If nickname was previously 'testnick' this test is
        invalid.");
    getActivity().runOnUiThread(new Runnable() {
       public void run() {
           nickname.setText("");
           nickname.requestFocus();
    }
    });
    sendKeys(TESTNICK_KEY_PRESSES);
    SharedPreferences settings =
           getActivity().getSharedPreferences(QuizActivity.GAME_PREFER
           ENCES,
              Context.MODE_PRIVATE);
    String fromPrefs =
           settings.getString(QuizActivity.GAME_PREFERENCES_NICKNAME,
           "");
    assertTrue("Prefs should be testnick", fromPrefs
    .equalsIgnoreCase("testnick"));
}
```

Как и в предыдущем случае, большая часть представленного кода — это стандартный код Android-приложения, который должен быть вам уже знаком. Тем не менее обратите

внимание, что данный код выполняет несколько вызовов методов в потоке, управляющем пользовательским интерфейсом. Эти вызовы обязательны для данного сценария; если убрать эти вызовы из потока, управляющего пользовательским интерфейсом, сценарий тестирования завершится неудачей.

ВНИМАНИЕ!

Чтобы весь тестовый метод выполнялся в потоке, управляющем пользовательским интерфейсом, добавьте аннотацию `@UiThreadTest` перед реализацией вашего метода. Но стоит отметить, что этот подход не будет работать в продемонстрированном примере, поскольку метод `sendKeys()` не может выполняться в основном потоке. (В этом случае возникнет исключение «This method cannot be called from the main application thread» (Этот метод не может вызываться из основного потока приложения).) Вместо этого в потоке, управляющем пользовательским интерфейсом, могут выполняться лишь части теста, как показано выше.

ВЫПОЛНЕНИЕ АВТОМАТИЗИРОВАННЫХ ТЕСТОВ

Теперь, когда вы написали ваши тесты, необходимо запустить их. чтобы протестировать ваш код. Это можно сделать двумя способами. Первый способ наиболее прост и позволяет видеть удобочитаемые результаты непосредственно в среде разработки Eclipse: для этого просто нужно выбрать команду меню **Run** => **Debug As** => **Android JUnit Test** (Выполнение => Отлаживать как => Тест JUnit Android). На панели **Console** (Консоль) среды разработки Eclipse отображается ход типовой установки как для тестового приложения, так и для тестируемого приложения (рис. 22.3).

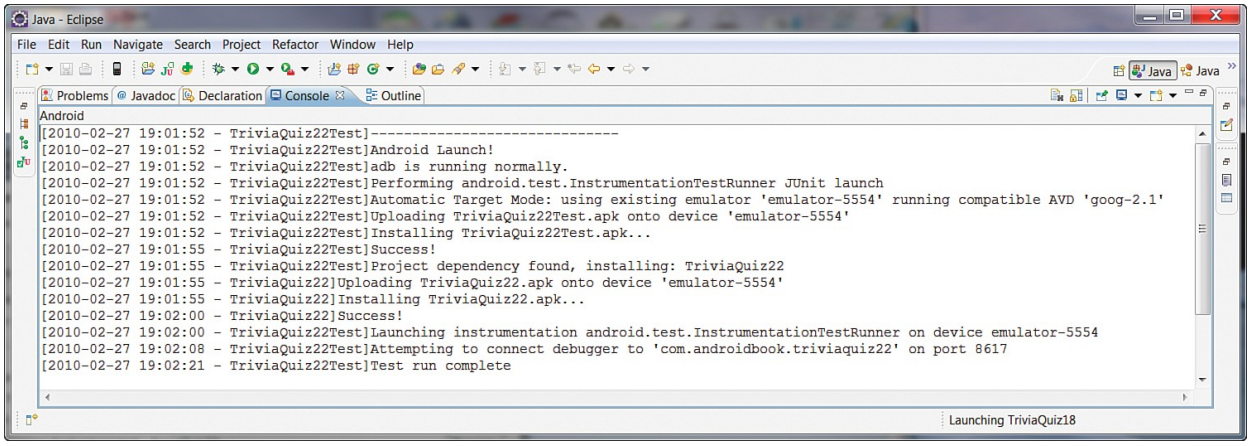

Рис. 22.3. Содержимое панели **Console** (Консоль) среды разработки Eclipse при выполнении тестов JUnit на платформе Android

ВНИМАНИЕ!

Если тестовый проект не выбран, среда разработки Eclipse может попытаться запустить обычное приложение в качестве тестового приложения JUnit, что приведет к появлению множества предупреждений и ошибок. Чтобы избежать этой проблемы, щелкните правой кнопкой мыши по названию проекта на панели **Package Explorer** (Проводник пакетов) среды разработки Eclipse и выберите в появившемся контекстном меню команду **Debug As** => **Android JUnit Test** (Отлаживать как => Тест JUnit Android). В качестве альтернативы вы можете открыть диалоговое окно **Debug Configurations** (Конфигурации для отладки) и дважды щелкнуть по элементу списка **Android JUnit Test** (Тест Android JUnit), чтобы создать новую конфигурацию для тестирования, а затем настроить необходимые параметры.

На панели **LogCat** можно увидеть как обычную отладочную информацию Android-приложения, так и новую информацию по выполняемым тестам. Благодаря этой панели вам будет проще отлаживать проблемы или ошибки, которые возникают в результате сбоев, или даже находить новые сбои, которые должны быть устранены и протестированы.

Тем не менее наиболее полезной может оказаться панель JUnit. На этой панели собрана информация обо всех прогонах тестов и о времени, которое занимал каждый прогон, а для каждого обнаруженного сбоя включена трассировка стека. На рис. 22.4 показано, как выглядит эта панель в среде разработки Eclipse.

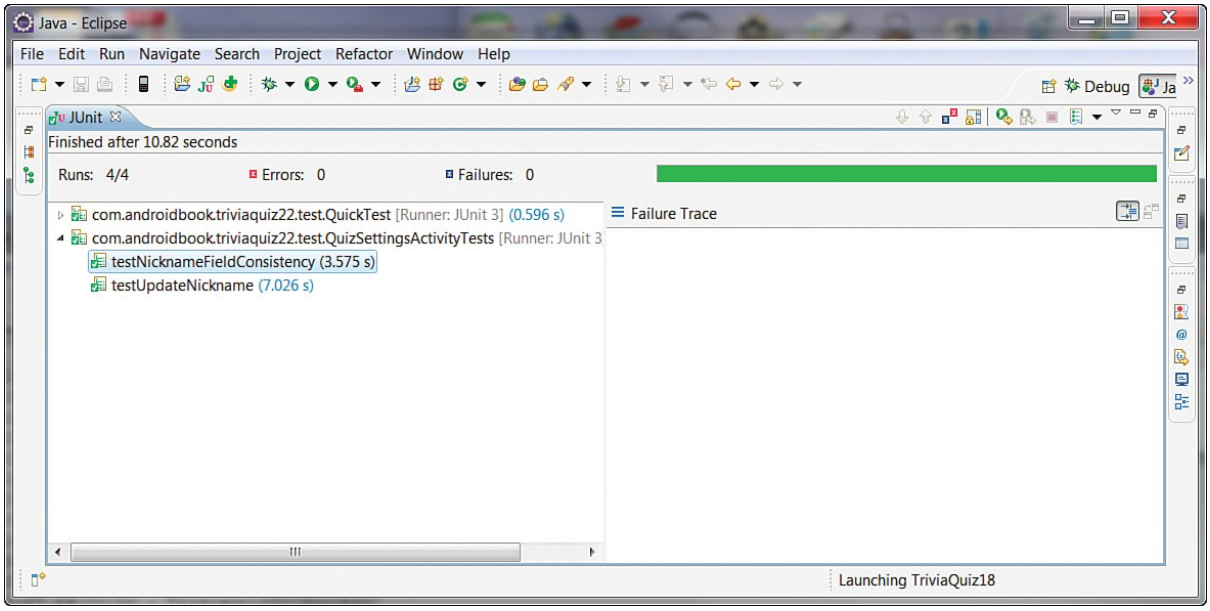

Рис. 22.4. Панель JUnit в среде разработки Eclipse с информацией о выполненных тестах Android-приложения

Второй способ выполнения тестов доступен только на эмуляторе. Чтобы воспользоваться данным методом, запустите приложение Dev Tools и в появившемся списке выберите пункт **Instrumentation** (Средства контроля). Если вы следовали всем нашим инструкциям и не устанавливали никаких других тестов, вероятнее всего, вы увидите в открывшемся списке лишь один элемент **android.test.InstrumentationTestRunner**. Нажатие на этот элемент приведет к запуску тестов. При использовании данного метода единственный способ

увидеть результаты (за исключением наблюдения за действиями, происходящими на экране, при тестировании пользовательского интерфейса) — просмотр содержимого панели **LogCat**.

Описание элемента в списке **Instrumentation** (Средства контроля) может быть изменено. В файле **AndroidManifest.xml** тестового приложения измените элемент `<instrumentation>` следующим образом:

```
<instrumentation
    android:targetPackage="com.androidbook.triviaquiz22"
    android:name="android.test.InstrumentationTestRunner"
    android:label="TriviaQuiz22 Tests" />
```

Теперь, когда вы снова запустите приложение Dev Tools и в появившемся списке выберете пункт **Instrumentation** (Средства контроля), вы увидите на экране указанную метку (атрибут `label`), а не название тестов.

ДОБАВЛЕНИЕ ДОПОЛНИТЕЛЬНЫХ ТЕСТОВ

Теперь в вашем распоряжении есть все инструменты, которые нужны для добавления дополнительных модульных тестов в ваше приложение Инструментарий Android SDK предоставляет ряд классов, которые мот быть реализованы для выполнения широкого спектра тестов, характерных для платформы Android. Вот некоторые из них:

- `ActivityUnitTestCase` — использование данного класса похоже на пример тестирования из предыдущего раздела в том плане, что проверяется деятельность, но на более низком уровне. Этот класс может быть использован для модульного тестирования отдельных аспектов деятельности, например, как она реагирует на событие `onPause()` после вызова метода `onFinished()`, и т. п. Это отличный способ протестировать жизненный цикл деятельности.

- `ApplicationTestCase` — как и класс `ActivityUnitTestCase`, ЭТОТ класс позволяет тестировать классы `Application` в полностью контролируемой среде.

- `ProviderTestCase2` — выполняет изолированное тестирование контент-провайдера.

- `ServiceTestCase` — выполняет изолированное тестирование службы.

Помимо этих классов, представляющих различные сценарии тестирования, существуют вспомогательные классы для создания тестовых объектов (т. е. объектов, которые не являются настоящими, однако могут быть использованы для воспроизведения определенных трассировок стека, полученных на реальных объектах), вспомогательные классы для имитации событий, возникающих в результате взаимодействия с сенсорным экраном, и другие похожие утилиты. Исчерпывающую документацию по этим классам можно найти в пакете `android.test`.

ИТОГИ

В этом часе вы узнали о различных способах тестирования и улучшения Android-приложений, позволяющих получить продукт более высокого качества, который по достоинству будет оценен пользователями. Вы познакомились со многими лучшими практиками тестирования приложений для мобильных телефонов, включая создание продуманного и полного плана тестирования. Вы узнали о нескольких способах получения мобильных телефонов для тестирования приложения. Вы также узнали, как создавать автоматизированные тесты для Android-приложений с использованием среды тестирования JUnit. Наконец, вы познакомились с некоторыми специфическими проблемами, связанными с тестированием, которые должны быть включены в план тестирования любого хорошего продукта.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Существуют ли какие-нибудь программы сертификации дня Android-приложений?

Ответ: В настоящий момент программ сертификации для Android- приложений нет. Том не менее провайдеры, операторы и отдельные сервисы. занимающиеся продажей программного обеспечения для мобильных телефонов, могут по своему усмотрению устанавливать свои собственные требования к качеству приложений.

Вопрос: Где я могу найти дополнительную информацию по созданию наборов автоматизированных тестов с использованием среды тестирования JUnit?

Ответ: Посетите веб-сайт, посвященный среде тестирования JUnit, по адресу **www.junit.org** или прочтите одну из множества книг, посвященных этому вопросу.

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Разработчики могут создавать автоматизированные тесты для выполнения Android-приложений программным путем.

2. Что из перечисленного свидетельствует о дефекте или ошибке в приложении?

A. Запуск приложения занимает слишком много времени.
B. Происходит аварийный сбой при входящем звонке.
C. Текст на немецком языке содержит очень много символов и не умещается на экране.
D. Кнопки слишком маленькие или расположены слишком близко друг к другу, что не позволяет нажимать на них пальцем.
E. Приложение входит в бесконечный цикл при выполнении определенных условий.
F. Все вышеперечисленное.

3. Верно ли это? Автоматизированное тестирование Апс1г(мс1-приложений может выполняться только на эмуляторе.

4. Верно ли это? Среда тестирования JUnit, поставляемая вместе с инструментарием Android SDK, может быть использована для тестирования многих вещей. Что из следующею она не позволяет делать"

A. Без устали выполнять одни и те же тесты на протяжении всего дня
B. Проводить тестирование на старых устройствах.
C. Перемещать мобильный телефон по стране, чтобы проверять сигнал со спутников GPS.
D. Тестировать поведение приложения в различных сотовых сетях / в сетях различных операторов.

Ответы

1. Верно. Инструментарий Android SDK включает ряд пакетов для разработки наборов тестов для автоматизированного тестирования приложения.

2. F. Все это дефекты различных типов — дефекты, связанные с производительностью, интеграцией, интернационализацией, удобством пользования и функциональностью. (A) Приложение, для запуска которого требуется продолжительное время, представляет серьезную проблему в плане производительности, поскольку операционная система Android может завершить его. (B) Большинство устройств, работающих под управлением операционной системы Android, в первую очередь телефоны; приложение должно хорошо взаимодействовать с остальной частью системы, что подразумевает элегантную обработку входящих звонков и текстовых сообщений. (C) Хорошо написанное приложение не будет разочаровывать пользователей, использующих иностранные языки, нестандартным пользовательским интерфейсом. (D) Хорошо спроектированный пользовательский интерфейс — важная составляющая успеха приложения. (E) Функциональный дефект, т. е. проблема в бизнес-логике приложения, — это всегда дефект, независимо от вероятности его проявления.

3. Неверно. Автоматизированные тесты могут выполняться на любом устройстве Android, подключенному к компьютеру для отладки.

4. C и D. К сожалению, сама по себе среда тестирования JUnit не позволяет физически перемещать мобильные телефоны по свету. Кроме того, она не позволяет воспроизводить нюансы конкретных операторов сотовой связи по всему миру. Тем не менее настройки эмулятора могут быть использованы для имитации определенных характеристик сети в плане производительности, но не имитации совершенно другой сетевой среды.

Упражнения

1. Подготовьте высокоуровневый план тестирования для приложения «Been There, Done That!».

2. Напишите сценарий тестирования для проверки корректности загрузки аватара пользователя.

3. Изучите различные соглашения, с которыми вы столкнулись после начала разработки Android-приложений (например, лицензионное соглашение для инструментария Android SDK и условия использования интерфейса Google Maps API). Определите любые сценарии тестирования, которые могут потребоваться для доказательства соответствия приложения этим соглашениям.

ЧАСТЬ V. ПУБЛИКАЦИЯ ВАШЕГО ПРИЛОЖЕНИЯ
Час 23. ПОДГОТОВКА К ПУБЛИКАЦИИ ПРИЛОЖЕНИЯ

Вопросы, рассматриваемые в этом часе:

- **подготовка к публикации приложения;**
- **тестирование и проверка выпускаемой версии приложения;**
- **упаковка и подписание вашего приложения для выпуска.**

С функциональной точки зрения, приложение может быть полностью готово, однако вам необходимо пройти еще один этап: программу нужно упаковать, чтобы ее могли установить пользователи. В этом часе вы узнаете, как подготовить и упаковать приложение для размещения на сервисе Android Market.

ПОНИМАНИЕ ПРОЦЕССА ВЫПУСКА ПРИЛОЖЕНИЯ

Подготовка и упаковка приложения для последующей публикации называется процессом выпуска приложения (рис. 23.1). Это волнующий этап: приложение работает без проблем, все ошибки, причиняющие беспокойство, исправлены (по крайней мере, в разумных пределах), и вы готовы представить ваше приложение пользователям.

Последняя версия приложения — версия, которую вы планируете отдать пользователям, — называется кандидатом на выпуск. Версия-кандидат на выпуск должна быть досконально протестирована и проверена перед тем, как попадет в руки пользователей. Если версия-кандидат на выпуск прошла все тесты, она становится финальным выпуском — официальным выпуском для публикации.

КСТАТИ

Разные люди применяют разную терминологию при описании процесса выпуска приложения. Разные методологии используют разные термины. Некоторые компании придумывают для подобных событий кодовые названия, например «отправка на золото». С годами мы остановились на использовании терминов «выпуск» и «версия-кандидат», поскольку, независимо от выбранной методологии, эти термины понятны большинству разработчиков

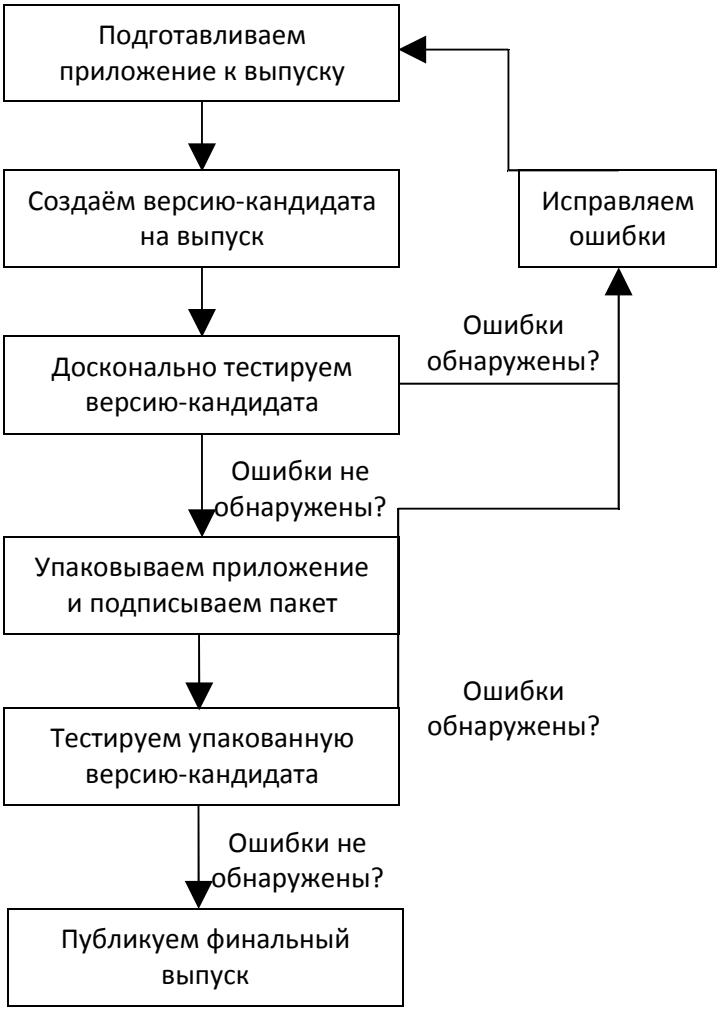

Рис. 23.1. Обзор процесса выпуска приложения

1. Для публикации Android-приложения необходимо выполнить следующие шаги:
2. Подготовить и создать версию-кандидата на выпуск для данного приложения.
3. Досконально протестировать созданную версию-кандидата.
4. Упаковать и подписать приложение цифровой подписью.
5. Досконально протестировать упакованную версию приложения.
6. Опубликовать приложение.

Давайте рассмотрим каждый из этих шагов более подробно.

ПОДГОТОВКА ВЕРСИИ-КАНДИДАТА НА ВЫПУСК

Важно довести до блеска ваше приложение и подготовить ею к публичному использованию. Это значит, что вы должны устранить любые открытые и ожидающие решения проблемы в приложении, которые «опт помешать его выпуску. Вся функциональность должна быть реализована и протестирована. Все ошибки должны быть исправлены или отложены Наконец, вы должны удалить из приложения любой ненужный

кол лля получения диагностической информации и убедиться, что в файле манифеста Android указаны настройки конфигурации, соответствующие выпуску приложения.

Вот небольшая контрольная таблица действий, которые необходимо выполнить перед выпуском типового Android-приложения:

- В полной мере протестировать приложение, как описано в плане тестирования, включая тестирование на целевых мобильных телефонах.

- Исправить и проверить исправление всех дефектов и ошибок в приложении.

- Отключить в финальной версии генерацию любой отладочной информации, включая ненужные инструкции журналирования, которые могут негативно отразиться на производительности приложения.

Подготовка файла манифеста Android для выпуска приложения

Перед выпуском приложения вы должны внести ряд изменений в настройки конфигурации приложения, которые представлены в файле манифеста Android. Некоторые из этих изменений имеют общий практический смысл, а другие изменения обусловлены требованиями, налагаемыми торговыми площадками, например сервисом Android Market.

Вы должны проверить файл манифеста Android на предмет следующего:

- Убедиться, что значок приложения (PNG-изображения с различными размерами) установлен надлежащим образом. Этот значок будут видеть пользователи, и он зачастую используется на торговых площадках дня представления приложения.

- Убедиться, что указана надлежащая метка приложения. Эта метка будет использоваться в качестве названия приложения, которое будут видеть пользователи.

- Убедиться, что указано надлежащее название версии приложения. Название версии — это дружественная метка версии, которую используют разработчики (и торговые площадки).

ВНИМАНИЕ

Инструментарий Android SDK позволяет использовать в качестве значения атрибута android:versionName ссылку на строковый ресурс. Сервис Android Market не позволяет делать этого.При проверке пакета а процессе выгрузки приложения на сервер вы увидит ошибку. Такой пакет не будет принят сервисом.

- Убедиться, что указан надлежащий код версии приложения. Код версии – это число, которое используется платформой Android для управления обновлениями приложения. Рекомендуется увеличить значение кода версии для версии-кандидата, чтобы иметь возможность отличить ее от предыдущих версий приложения.

- Убедиться, что указано надлежащее значение настройки приложения uses-sdk. Вы можете указать минимальную, целевую или максимальную версию инструментария Android SDK, поддерживаемую данной версией приложения. Эти значения

представляют идентификаторы API level, присваиваемые каждой версии инструментария Android SDK. Например, дня инструментария Android SDK 2.1 используется идентификатор API level равный 7.

ЗНАЕТЕ ЛИ ВЫ ЧТО...
Сервис Android Market отбирает приложения, доступные для конкретных пользователей. на основании информации, представленной в каждом файле Манифеста приложения, включая информацию, указанную для настройки uses-sdk.

- Отключить настройку debuggable.

- Убедиться, что указаны надлежащие разрешения для приложения. Запрашивайте только те разрешения, которые требуются приложению, используя настройку конфигурации uses-permission, и запрашивайте разрешения, которые требуются для использования вашего приложения, независимо от поведения мобильного телефона в случае отсутствия этих разрешений.

Подготовка связанных сервисов к выпуску приложения

Если Android-приложение использует любые внешние технологии или сервисы, например, сервер приложения, то эти элементы также должны быть подготовлены к выпуску приложения.

Во многих крупных проектах имеется как «тестовый» сервер приложения (часто называемый песочницей), так и настоящий «рабочий» сервер. Финальная версия приложения должна быть протестирована на «рабочем» сервере, поскольку пользователи будут использовать именно его.

ТЕСТИРОВАНИЕ ВЕРСИИ-КАНДИДАТА НА ВЫПУСК

Решив все проблемы, связанные с подготовкой приложения к выпуску и перечисленные ранее, вы можете приступать к сборке версии-кандидата на выпуск. В этом процессе сборки нет ничего особенного, за исключением того, что в среде разработки Eclipse для запуска приложения нужно выбрать команду **Run** (Выполнить) вместо команды **Debug** (Отладить)

Вы должны протестировать версию-кандидата настолько досконально. нисколько это возможно. Помимо выполнения обычных процедур тестирования, вы должны убедиться, что приложение соответствует требованиям, предъявляемым рынками распространения приложений (например, сервисом Android Market), на которых вы планируете опубликовать ваше приложение.

Если вы обнаружили какие-либо дефекты или проблемы в версии-кандидате на выпуск, вам нужно решить, являются ли они достаточно серьёзными, чтобы остановить процесс выпуска приложения. Если вы решите, что проблема достаточно серьёзная и для её решения потребуется создать новую версию, вы просто запускаете процесс выпуска приложения сначала (рис. 23.2).

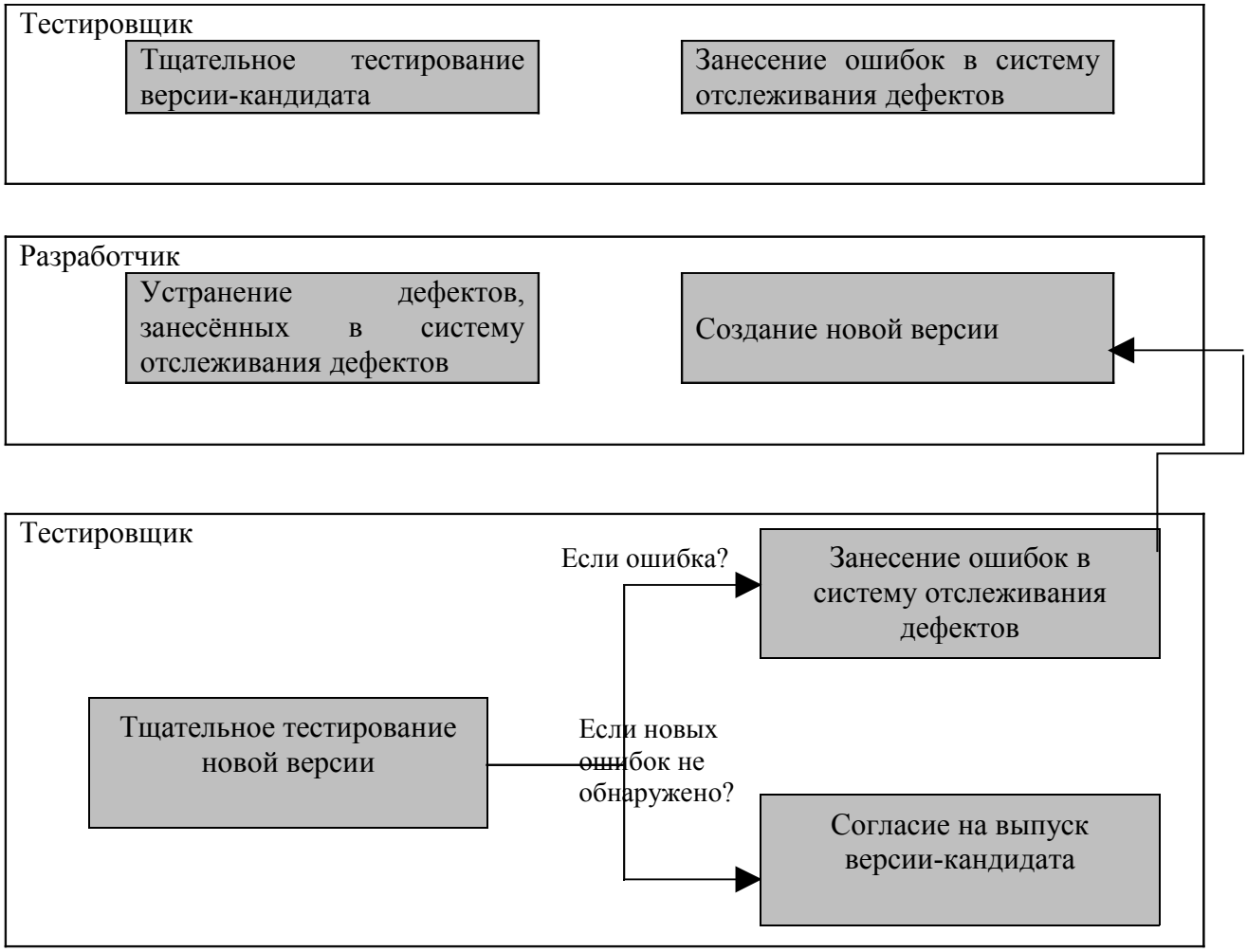

Рис. 23.2 Цикл тестирования версии-кандидата на выпуск

УПАКОВКА И ПОДПИСАНИЕ ПРИЛОЖЕНИЯ

Теперь, когда у вас есть отличная версия-кандидат на выпуск, которая была тщательно протестирована и готова к передаче пользователям, вам нужно упаковать приложение для его дальнейшей публикации. Этот процесс подразумевает генерацию файла пакета Android (файла с расширением .apk) и его подписание цифровой подписью.

Процесс упаковки и подписания приложения ещё никогда не был таким простым. К новейшим улучшениям плагина Android для среды разработки Eclipse относится мастер, который делает именно то, что нужно!

Подписание приложений цифровой подписью

Пакеты Android-приложений должны быть подписаны цифровой подписью, чтобы менеджер пакетов Android мог установить их. На протяжении процесса разработки среда разработки Eclipse использовала отладочный ключ для решения данного вопроса. Тем не менее, для финальной версии приложения вы должны использовать настоящую цифровую подпись — которая принадлежит вам и вашей компании. Для этого вы должны сгенерировать закрытый ключ.

ВНИМАНИЕ!

Закрытый ключ идентифицирует разработчика и имеет решающее значение для установления доверительных отношений между разработчиками и пользователями. Очень важно обеспечить конфиденциальность информации о закрытом ключе.

Закрытый ключ может быть использован для подписания выпускаемых файлов пакетов вашего Android-приложения, а также любых последующих обновлений приложения. Это гарантирует, что приложение (как целая сущность) поставляется именно вами, разработчиком.

КСТАТИ

Вам не нужно обращаться в сертифицирующий орган, например, в компанию Verisign, Equifax или любую другую компанию, которые удостоверятся в том, что вы именно тот, за кого себя выдаете, перед выдачей сертификата. Использование самоподписываемых сертификатов — стандартная практика для Android- приложений. Это означает, что вам не нужно доказывать, кто вы есть на самом деле, однако, когда в следующий раз вы опубликуете другое приложение (или что- либо иное) и ключи, которые использовались для подписания данного приложения и вашего предыдущего приложения, совпадут, пользователи (и операционная система Android) будут знать, что все это было подписано одним и тем же человеком или компанией. Поэтому не передавайте ваш закрытый ключ кому бы то ни было)

Обновления приложения должны подписываться тем же закрытым ключом. В целях безопасности менеджер пакетов Android не установит обновление поверх существующего приложения, если ключи, использованные для подписания обновления и исходного приложения, будут отличаться. Это значит, что вы должны хранить ключ, который применялся для подписании нашего приложения, и безопасном, легкодоступном месте для последующего использования.

ЗНАЕТЕЛИ ВЫ ЧТО...

Платформа Android проверяет цифровую подпись только на этапе установки приложения. Следовательно, если срок действия подписи истечёт после установки приложения, само приложение будет продолжать работать.

Экспортирование и подписание файла пакета

Теперь вы готовы к тому, чтобы экспортировать и подписать файл пакета вашего Android-приложения. При использовании мастера, являющегося частью плагина среды разработки Eclipse, данная процедура включаем следующие шаги:

1. В среде разработки Eclipse щёлкните правой кнопкой мыши по проекту нужного приложения и в открывшемся контекстном меню выберите команду **Export** (Экспортировать).

2. В списке появившегося диалогового окна **Export** (Экспорт) разверните раздел Android и выберите пункт **Export Android Application** (Экспортировать приложение Android).

3. Нажмите на кнопку **Next** (Далее).

4. Выберите проект для экспорта и нажмите на кнопку **Next** (Далее). (По умолчанию будет выбран проект, по названию которого вы ранее щелкнули правой кнопкой мыши.)

5. На экране **Keystore selection** (Выбор хранилища ключей) мастера установите переключатель в положение **Create New Keysto**re (Создать новое хранилище ключей) и укажите местоположение файла (местоположение, где вы хотите сохранить ключ) в поле ввода **Location** (Местоположение), а также введите пароль для доступа к хранилищу ключей (поля ввода **Password** (Пароль) и **Confirm** (Подтверждение пароля)). (Если у вас уже есть хранилище ключей, нажмите на кнопку **Browse** (Обзор), чтобы выбрать файл вашего хранилища ключей, и затем введите пароль для этого хранилища.)

ВНИМАНИЕ!

Всегда используйте надёжные пароли для хранилища ключей. Также запомните, где находится ваше хранилище ключей. Ключ, хранящийся в этом хранилище, потребуется вам снова при публикации обновлений для вашего приложения. Если файл хранилища ключей помещён в систему контроля версий, пароль поможет защитить этот файл, однако желательно добавить дополнительную степень защиты для доступа к этому файлу.

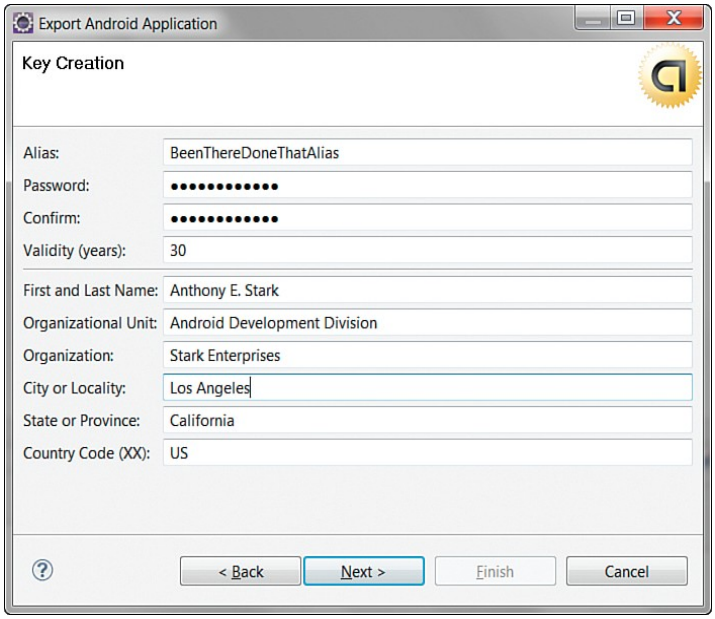

Рис. 23.3. Экспортирование Android-приложения

6. Нажмите на кнопку **Next** (Далее).

7. На экране **Key Creation** (Создание ключа) мастера введите информацию, необходимую для создания ключа, как показано на рис. 23.3.

ВНИМАНИЕ!
Команда разработчиков операционной системы Android рекомендует указывать в качестве срока действия ключа (поле ввода Validity (Срок действия)) значение, равное 25 годам или больше. На самом деле, сервис Android Market отклонит любое приложение, подписанное ключом, срок действия которого истекает до 22 октября 2033 года, поэтому срок действия ключа, равный 25 годам, также удовлетворяет данному требованию.

8. Нажмите на кнопку **Next** (Далее).

9. На экране **Destination and Key/Certificate checks** (Выбор местоположения файла пакета и проверка ключа/сертификата) мастера укажите местоположение файла пакета вашего приложения.

10. Нажмите на кнопку **Finish** (Готово).

Вы создали подписанный и сертифицированный файл пакета приложении, который полностью готов к использованию.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...
Для создания подходящего ключа и подписания файла пакета приложения (.apk) вы также можете использовать приложения keytool и jarsigner, доступные в иистру ментарии JDK, в дополнение к утилите zipalign, входящей в состав инструментария Android SDK. И хотя утилита zipalign не имеет непосредственного отношения к подписанию файлов пакетов, она позволяет оптимизировать пакет для его более эффективного использования в операционной системе Android. Плагин AOT для среды разработки Eclipse запускает утилиту zipalign автоматически после этапа подписания файла пакета.

ТЕСТИРОВАНИЕ ПОДПИСАННОГО ПАКЕТА ПРИЛОЖЕНИЯ

Теперь, когда вы подписали и упаковали ваше приложение, и оно готово к публикации, вы должны выполнить последний цикл тестирования, уделив особое внимание незначительным изменениям, связанным с процессом установки подписанных приложений.

Установка подписанного пакета приложения

Вплоть до этого момента вся работа по упаковке и переносу приложения на мобильные телефоны и эмуляторы для последующей отладки выполнялась средой разработки Eclipse. Теперь на жестком диске вашего компьютера имеется финальная версия приложения, которую вы должны загрузить и протестировать.

КСТАТИ

Перед тем, как установить финальную версию вашего приложения на эмулятор или мобильный телефон, вы должны полностью удалить отладочную версию приложения. Для этого на мобильном телефоне (или эмуляторе), находясь на домашнем экране, нажмите кнопку **Menu**, затем в появившемся меню выберите команду **Settings** (Настройки), в открывшемся списке выберите элемент **Applications** (Приложения), в следующем списке выберите элемент **Manage Applications** (Управление приложениями), и выбрав нужное приложение в списке, нажмите на кнопку **Uninstall** (Удалить) и подтвердите, что вы действительно хотите удалить выбранное приложение. Стоит отметить, что совместно используемые файлы или данные, например изображения, хранящиеся в галерее, могут остаться в системе. Тем не менее, внутренние файлы и данные приложения, включая настройки, будут удалены.

Простейший способ вручную установить (или удалить) файл пакета приложения (.apk) на мобильном телефоне или на эмуляторе — это использовать инструмент командной строки adb. Следующая команда позволяет установить пакет с использованием утилиты adb:

```
adb install <путь_к_файлу_.apk>
```

Эта команда будет работать только в том случае, если вы используете только одно устройство или эмулятор. Однако, если к вашему компьютеру подключено несколько устройств или запущено несколько эмуляторов, вам нужно указать конкретное устройство, на котором должно быть установлено приложение. Вы можете использовать команду devices утилиты adb, чтобы получить список устройств, подключённых к вашему компьютеру:

```
adb devices
```

Список, возвращаемый данной командой, содержит все эмуляторы н устройства, подключённые к компьютеру. Результаты выполнения этой команды могут выглядеть следующим образом:

```
$ adb devices
List of devices attached
emulator-5554 device
HT9CSP801234 device
```

Вы можете указать конкретное устройство, на которое хотите установить файл пакета приложения, используя параметр -s. Например, чтобы установить файл пакета приложения BeenThereDoneThat.apk на эмуляторе, можно использовать следующую команду:

```
adb -s emulator-5554 install BeenThereDoneThat.apk
```

Дополнительную информацию по использованию инструмента командной строки adb можно найти на веб-сайте, посвящённом разработке на платформе Android, по адресу **developer.android.com/guide/developing/tools/adb.html**.

Проверка подписанного приложения

Все почти готово. Осталось выполнить несколько последних проверок, чтобы убедиться, что приложение функционирует надлежащим образом:

- Проверить, что установка подписанного пакета приложения завершилась успешно.
- Убедиться, что весь отладочный функционал был отключён.
- Проверить, что приложение использует «рабочие» сервисы, а не «тестовые».
- Убедиться, что конфигурационные данные приложения, например, название приложения и значки, а также информация о версии приложения, отображаются корректно.

Если вы обнаружили какие-либо проблемы с функционированием подписанного приложения, вы должны решить, являются ли эти проблемы достаточно серьёзными, чтобы остановить процесс выпуска приложения и начать его заново. Когда вы досконально протестируете пакет приложения и будете уверены, что пользователи получат положительные впечатления от использования вашего приложения, вы можете приступать к публикации приложения!

ИТОГИ

В этом часе вы узнали о процессе подготовки приложения к публикации. В частности, вы познакомились с шагами, которые необходимо выполнить, чтобы убедиться в готовности вашего приложения к публикации, в том числе с отключением отладочной информации и проверкой конфигурационных настроек приложения. После этого вы узнали, как экспортировать неподписанный файл пакета приложения, сгенерировать закрытый ключ и подписать приложение цифровой подписью для его последующей публикации.

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Подойдёт ли описанный в этом часе процесс выпуска приложения для любой торговой площадки, распространяющей приложения для платформы Android?

Ответ: Вообще говоря, да. Мы сфокусировались на требованиях, предъявляемых сервисом Android Market. Чтобы получить дополнительную информацию по требованиям, выдвигаемым другими торговыми площадками, обратитесь к описанию соответствующих программ для разработчиков. Обычно все отличия в выдвигаемых требованиях связаны с файлом манифеста Android приложения и с цифровой подписью, которой подписывается пакет приложения.

Вопрос: Почему ключ должен быть действителен до 22 октября 2033 года?

Ответ: Цифровая подпись приложения может использоваться после выхода последующих обновлений приложения. Когда срок действия цифровой подписи заканчивается далеко в будущем, между разработчиком приложения и третьими сторонами (включая пользователей) могут налаживаться и поддерживаться продолжительные доверительные отношения.

Вопрос: Могу ли я получить информацию о пакете приложения программным путем?

Ответ: Да, вы можете использовать метод `getPackageInto()` класса `PackageManager`, чтобы получить информацию о пакете приложения. Этот метод воз-

вращает объект типа `PackageInfo`, который содержит всю информацию о файле манифеста этого приложения, начиная с настроек конфигурации и заканчивая списком конкретных деятельностей и разрешениями приложения.

ПРАКТИКУМ
Контрольные вопросы

1. Верно ли это? Процесс выпуска приложения имеет важное значение только для крупных проектов.

2. Какие наш в файле манифеста Android, связанные с версией приложения. должны быть проверены в процессе выпуска приложения?

A. `inviteid:versionCode.`
B. `android:versionLabel.`
C. `android:versionName.`
D. `android:version.`
E. Все вышеперечисленные.

3. Верно ли это? Нельзя опубликовать приложение, подписанное подписью, применяемой для отладки.

Ответы

1. Неверно. Независимо от того, являетесь вы человеком, разрабатывающим приложения для своего удовольствия, или членом большой команды разработчиков, время, потраченное для того, чтобы убедиться, что приложение готово к выпуску, — это важная составляющая успеха приложения.

2. A и C. Платформа Android использует код версии для выполнения обновлений приложения, а название версии используется разработчиками и торговыми площадками для сопровождения продукта.

3. Верно. Менеджер пакетов Android устанавливает только приложения, которые подписаны надлежащим образом.

Упражнения

1. Выберите одну из версий приложения «Been There, Done That!», доступных на диске, прилагаемом к данной книге (из любого часа). Экспортируйте APK-файл пакета и подпишите его цифровой подписью.

2. Установите пакет приложения «Been There, Done That!» на мобильном телефоне, используя утилиту командной строки `adb`.

Час 24. ПУБЛИКАЦИЯ ПРИЛОЖЕНИЯ В СЕРВИСЕ ANDROID MARKET

Вопросы, рассматриваемые в этом часе:

- **продажа Android-приложений через сервис Android Market;**
- **исследований вариантов публикации Android-приложений;**
- **защита вашей интеллектуальной собственности.**

Поздравляем! Вы дошли до завершающего часа, узнав за все это время, как создавать и тестировать Android-приложения. Следующий логический шаг — публикация вашего приложения. В этом часе вы узнаете, как опубликовать приложение в популярном сервисе Android Market, и познакомитесь с другими вариантами распространения приложений.

Платформа Android поддерживает варианты платного, бесплатного и даже самостоятельного распространения приложений. Благодаря этому разработчик получает огромную свободу, а пользователи — широчайшие возможности.

ПРОДАЖА ПРИЛОЖЕНИЙ ЧЕРЕЗ СЕРВИС ANDROID MARKET

В настоящее время сервис Android Market — основной механизм распространения Android-приложений. Именно здесь обычные пользователи покупают и загружают приложения. Доступ к сервису Android Market имеет большинство мобильных телефонов, работающих под управлением ОС Android. Ниже мы покажем вам, как убедиться в готовности пакета к публикации, создать аккаунт разработчика и отправить ваше приложение для продажи через сервис Android Market.

Создание аккаунта разработчика

Для публикации приложений в сервисе Android Market вы должны зарегистрироваться в качестве разработчика. Регистрация в качестве разработчика позволяет компании Google идентифицировать вашу личность и создать аккаунт в сервисе Google Checkout, который используется сервисом Android Market для возврата разработчикам части дохода (части **дохода!? блеать! да они охуели!)** от продаж приложений.

Чтобы создать аккаунт в сервисе Android Market, выполните следующие шаги:

1. Откройте страницу **http://market.android.com/publish/signup**, изображенную на рис.24.1.

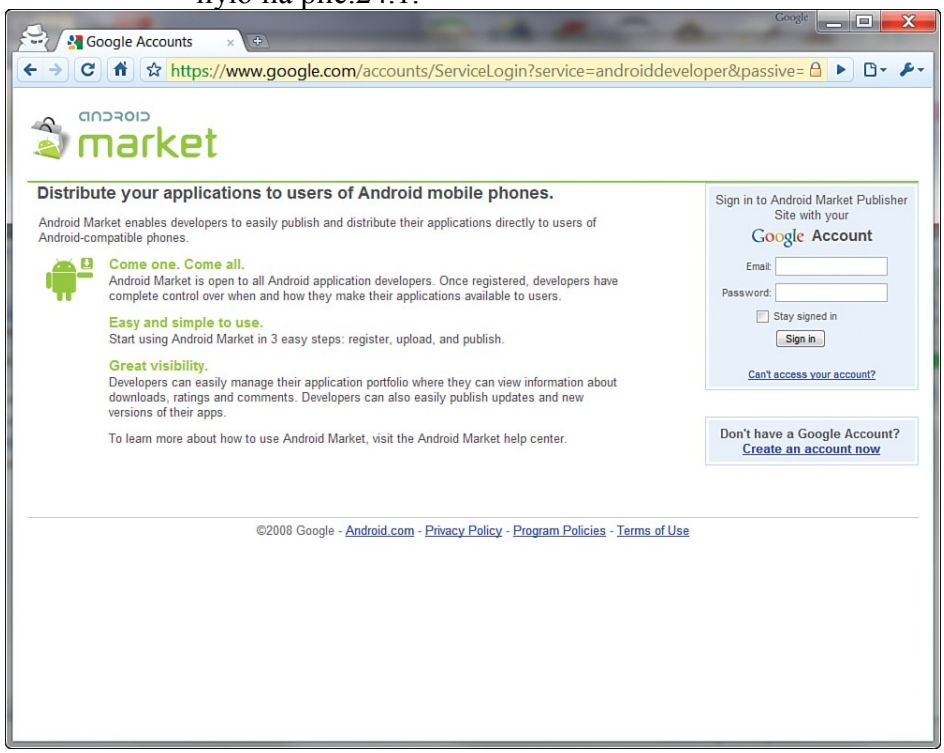

Рис. 24.1. Страница создания аккаунта в сервисе Android Market

2. Войдите в систему, используя аккаунт Google. (Нельзя изменить аккаунт Google, связанный с аккаунтом разработчика, но вы можете указывать разные контактные адреса электронной почты для приложений.)

3. Введите информацию о себе — имя, адрес электронной почты и вашего веб-сайта, как показано на рис. 24.2.

4. Подтвердите ваш платёж за регистрацию (25 долларов США). Обратите внимание, что для выполнения платежа за регистрацию используется сервис Google Checkout.

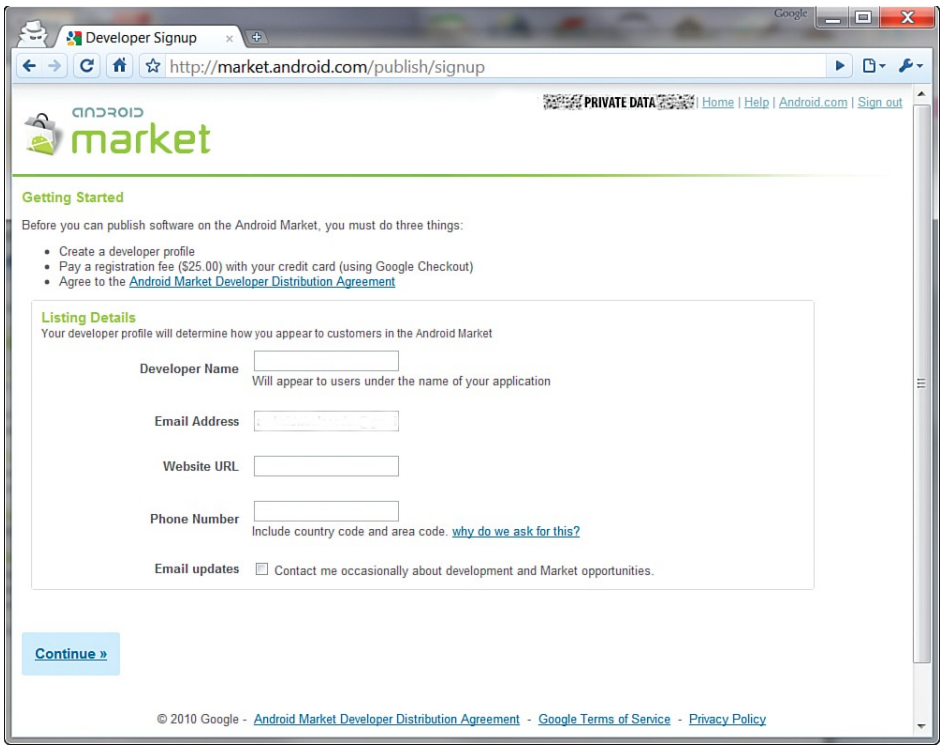

Рис. 24.2. Страница с информацией о разработчике в сервисе Android Market

5. Укажите информацию для создания аккаунта в сервисе Google Checkout. Это обязательный шаг при создании аккаунта разработчика Android- приложений и платеже за регистрацию.

6. Согласитесь связать вашу кредитную карту и зарегистрированный аккаунт с Соглашением о распространении приложений в сервисе Android Market (Android Market Developer Distribution Agreement).

Успешно выполнив перечисленные шаги, вы окажетесь на домашней странице сервиса Android Market, на которой вы также увидите подтверждение создания аккаунта в сервисе Google Checkout.

Загрузка приложения на сервис Android Market

Теперь, когда у вас есть аккаунт для публикации приложений через сервис Android Market и подписанный пакет приложения, все готово для загрузки этого пакета в сервис. Открыв домашнюю страницу разработчика на вебсайте сервиса Android Market, войдите в систему и нажмите на кнопку **Upload Application** (Загрузить приложение), как показано на рис. 24.3.

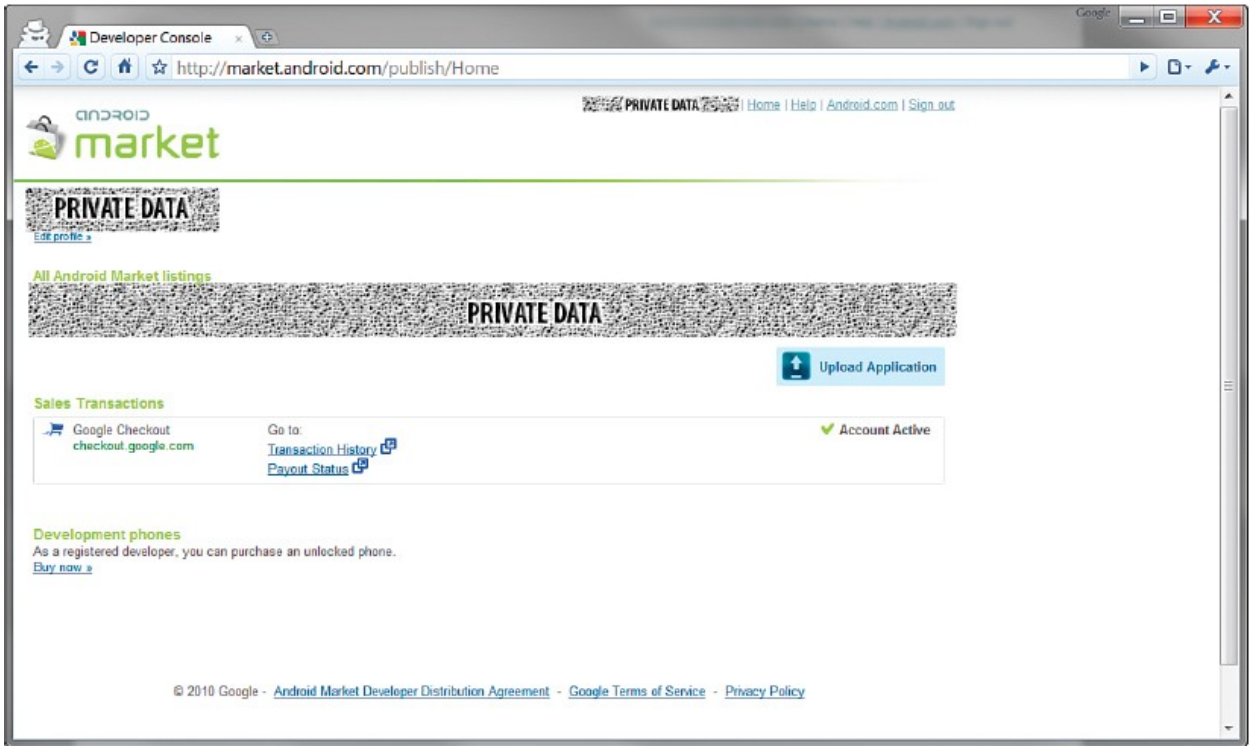

Рис. 24.3. Домашняя страница разработчика на веб-сайте сервиса Android Market

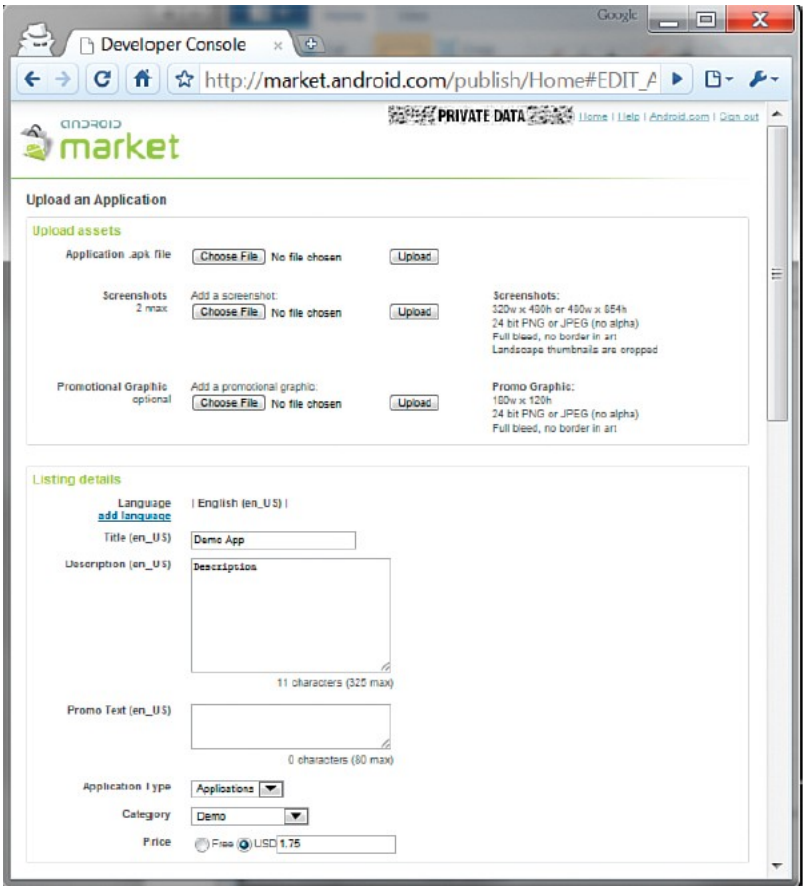

Рис. 24.4A. Форма для загрузки пакета приложения

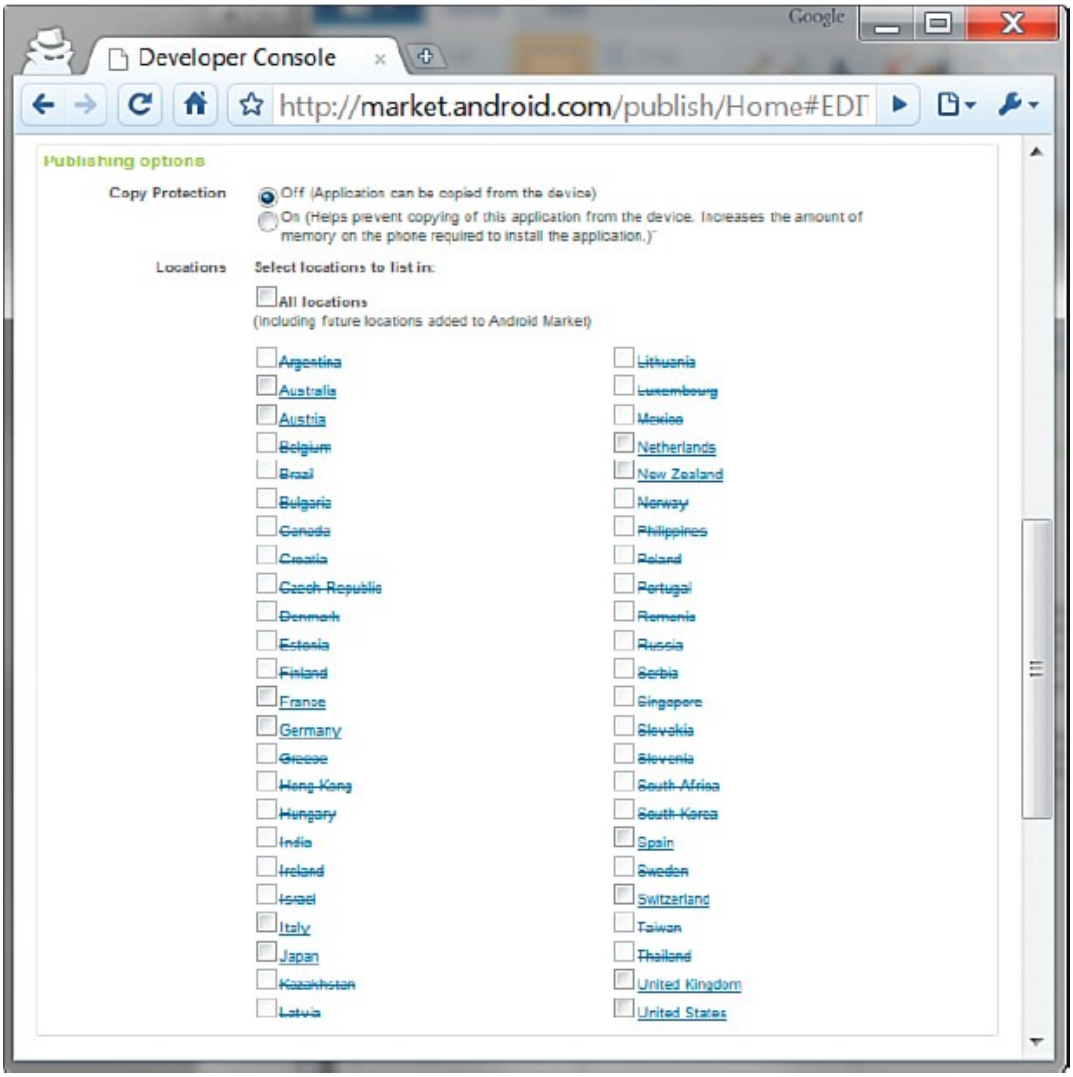

Рис. 24.4Б. Выбор стран, в которые будет экспортироваться ваше приложение

Теперь вы увидите форму, изображенную на рис. 24.4А и предназначенную для загрузки пакета приложения.

На рис. 24.4Б изображена страница, позволяющая разработчику указать страны, где будет продаваться его приложение. Все страны, на территории которых запрещено продавать платные приложения, будут недоступны в списке.

Ниже представлена часть информации, которую вы должны указать на этой форме.

- **Название и описание приложения на нескольких языках** — по умолчанию используется английский.

- **Страны (местоположения), где будет публиковаться приложение** - доступный список местоположений регулируется действующими законами об экспорте, поэтому вы должны ответственно подходить к выбору вашего местоположения. На момент публикации этой книги было доступно около 50 местоположений, при этом регулярно добавляются новые местоположения, кроме того, вы можете выбрать определенных операторов сотовой связи в каждом местоположении, чтобы еще больше ограничить распространение вашего приложения. В качестве альтернативы

вы можете установить флажок **All Locations** (Все местоположения), чтобы вьючить все новые местоположения, которые в будущем будут поддерживаться сервисом. Полный список местоположений, где мот продавайся платные и распространяться бесплатные Android- приложения, можно найти по адресу **http://market.android.com/support/bin/answer.py?hl=en&answer=138294**.

- **Тип и категория приложения** — потратьте некоторое время, чтобы указать для этих полей подходящие значения, которые определены в сервисе Android Market, — благодаря этому ваше приложение сможет достичь целевой аудитории. Неправильно классифицированные приложения продаются не очень хорошо.

- **Стоимость приложения** — сервис Android Market в настоящее время поддерживает только одну модель оплаты: единовременный платеж. В сервисе Android Market пока отсутствует модель подписки, основанная на регулярных платежах пользователей. Вы должны искать другие механизмы. если вас интересует модель с использованием регулярных платежей. Обратите внимание, что за размещение приложений сервис Android Market взимает операционный сбор, равный 30% от стоимости приложения. Цены на приложения могут варьироваться от 0,99 до 200 долларов США, и аналогичные диапазоны доступны в евро и английских фунтах.

- **Функция защиты от копирования** — установив переключатель **Copy Protection** (Зашита от копирования) в положение **On** (Включить), вы можете предотвратить копирование вашего приложения с устройства и его дальнейшее распространение без вашего ведома или разрешения.

- **Контактная информация для поддержки приложения** — по умолчанию используется информация, указанная в вашем аккаунте разработчика. Однако вы можете изменять эту информацию для каждого отдельного приложения. благодаря чему обеспечивается потрясающая гибкость в плане сопровождения при публикации нескольких приложений.

- **Согласие** — вы должны установить флажки, чтобы согласиться с условиями текущего (на тот момент, когда вы публикуете приложение) соглашения о допустимых типах контента Android-приложений, а также законами об экспорте Соединенных Штатов, независимо от вашего местоположения.

ВНИМАНИЕ!

Вы должны внимательно ознакомится с дополнительной информацией (на которую указывает ссылка Learn More (Узнать больше)) по соблюдению законов об экспорте. На соответствующей странице компания Google также включила ссылки на несколько веб-страниц правительства США, где содержится достаточное количество информации, чтобы вы могли определить, будут нарушены эти законы или нет.

В то время, пока вы заполняете поля формы, происходи загрузка приложения на сервер и его проверка.

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Как только пакет приложения будет успешно загружен на сервер, введенная информация может быть сохранена в качестве черновика, чтобы вы могли еще раз убедиться в ее правильности перед окончательной публикацией приложения. Кроме того, на странице появится значок приложения, его название, версия, информация о локализации и требуемые разрешения, благодаря чему вы сможете убедиться в том, что файл манифеста Android настроен надлежащим образом

Публикация в сервисе Android Market

После того, как вы нажмете на кнопку **Publish** (Опубликовать), приложение появится в сервисе Android Market практически мгновенно. Как только ваше приложение будет опубликовано, вы сможете просматривать разнообразную статистику, включая рейтинг приложения, отзывы, количество загрузок, активные установки и тому подобное, в разделе **All Android Market Listings** (Все позиции в каталоге сервиса Android Market) на основной странице вашего аккаунта разработчика. Стоит отметить, что статистика обновляется нечасто, и вы не можете просматривать отзывы, оставленные пользователями, непосредственно из списка позиций.

Нажав в списке на позиции приложения, вы можете отредактировать различные поля. И хотя некоторая информация может быть изменена, изменить ценовую политику для вашего приложения невозможно. Например, если вы опубликовали ваше приложение в качестве бесплатного, оно останется таким и в дальнейшем. Но вы всегда можете загрузить другую версию приложения в качестве платной, включающей новые функции. Стоимость платного приложения может изменяться в любой момент, однако она должна находиться в определенных пределах. (В долларах США этот диапазон составляет от 99 центов до 200 долларов США.)

ПОНИМАНИЕ МЕХАНИЗМА ОСУЩЕСТВЛЕНИЯ РАСЧЕТОВ

В отличие от некоторых других платформ для мобильных устройств, платформа Android в настоящее время не предоставляет никаких встроенных интерфейсов API для осуществления расчетов непосредственно из приложения или путем включении соответствующих сумм в счета пользователей на оплату услуг соговой связи. Вместо этого для совершения платежей сервис Android Market использует сервис Google Checkout. Произведя оплату стоимости приложения, пользователь становится его владельцем.

Если ваше приложение должно взимать плату за услуги или через него должна осуществляться продажа других товаров (например, рингтонов, музыки, электронных книг), вам придется разработать собственный механизм осуществления расчетов. Большинство устройств, работающих под управлением операционной среды Android, могут подключаться к Интернету, поэтому, вероятнее всего, простейшим решением будет использование онлайн-сервисов осуществления расчетов и интерфейсов API — например, компаний PayPal, Google и Amazon. Убедитесь, что предпочитаемый вами сервис осуществления расчетов может использоваться на мобильных телефонах.

ВНИМАНИЕ!

В настоящий момент соглашение, заключаемое с сервисом Android Market, не пзволяет взимать дополнительные платежи в самом приложении. И хотя на рынке доступны приложения, которые делают подобные вещи для улучшения самой привлекательности и удобства использования, с технической точки зрения они нарушают существующее Соглашения о распространении приложений в сервисе Android Market. Однако данные ограничения могут не затрагивать другие варианты дистрибуции приложений.

Другой способом получения денег от пользователей — показ рекламы в приложениях для мобильных устройств. Эта схема начала использоваться относительно недавно. Платформа Android не запрещает использовать рекламу в приложениях. И в этом нет ничего удивительного, учитывая популярность сервиса Google AdSense.

Понимание политики возврата приложений сервиса Android Market

Несмотря на нешуточную полемику, которая разыгрывалась вокруг данного вопроса, до недавнего времени сервис Android Market придерживался 24-часовой политики возврата приложений. Это значит, что пользователь мог использовать приложение в течение 24 часов и затем вернуть его обратно, и сервис Android Market возмещал пользователю все затраты в полном объеме. Для вас, как разработчика, это означало, что сделка по продаже приложения не считалась окончательно состоявшейся в течение первых 24 часов с момента загрузки. Теперь вернуть приложение можно только в течение 15 минут с момента его загрузки. Эта политика начинает действовать с момента первой загрузки приложения и до первого возврата этого приложения. Если конкретный пользователь уже вернул ваше приложение и теперь хочет «попробовать его снова», он должен совершить окончательную покупку — и он уже не сможет вернуть приложение во второй раз.

И хотя данный подход ограничивает возможности для неправильного использования приложений, вы все равно должны иметь в виду, что, если в вашем приложении отсутствуют очевидные причины для его повторного использования или если пользователь может получить от приложения все, что ему нужно, в течение нескольких минут, вы можете столкнуться с высоким количеством возвратов и вам придется искать другие способы взимания платы за ваше приложение.

УДАЛЕНИЕ ВАШЕГО ПРИЛОЖЕНИЯ ИЗ СЕРВИСА ANDROID MARKET

Вы можете использовать функцию отмены публикации приложения, доступную в вашем аккаунте разработчика, чтобы удалить приложение из сервиса Android Market. Функция отмены публикации приложения действует мгновенно, однако для удаления соответствующей информации из всей системы может потребоваться некоторое время.

Использование других преимуществ аккаунта разработчика

Зарегистрированный аккаунт разработчика Android-приложений позволяет вам управлять вашими приложениями в сервисе Android Market. Кроме того, если у вас есть аккаунт разработчика, вы можете приобретать опытные образцы мобильных телефонов под управлением операционной системы Android. Эти мобильные телефоны полезны

для обычной разработки и тестирования, однако они могут оказаться бесполезными при окончательном тестировании приложения на реальных целевых мобильных телефонах, поскольку на серийных телефонах часть исходной функциональности может быть «урезана», а версия программного обеспечения экспериментальной версии телефона может отличаться от версии программного обеспечения серийных телефонов.

РАССМОТРЕНИЕ ДРУГИХ ВАРИАНТОВ ПУБЛИКАЦИИ ANDROID-ПРИЛОЖЕНИЙ

Платформа Android — это открытая платформа, которая, благодаря своей открытости, предоставляет разнообразные варианты публикации приложений. Вы узнали, как можно публиковать приложения в сервисе Android Market, но существуют и другие варианты публикации. Возможно, вы захотите воспользоваться преимуществами этих альтернативных вариантов, чтобы распространять приложения для целевых мобильных телефонов и устройств, которые не поддерживаются сервисом Android Market, распространять мобильные телефоны среди более узкой целевой аудитории, распространять приложения, которые не отвечают требованиям сервиса Android Market, или же просто самостоятельно контролировать процесс распространения приложения.

ЗНАЕТЕ ЛИ ВЫ, ЧТО…

Существуют рынки альтернативных устройств, работающих под управлением операционной системы Android, например ARCHOS 5 Internet Tablet. Приложения, разработчиков специально для этих типов устройств, обычно (но не всегда) распространяются отдельно от приложений, разработанных для мобильных телефонов, работающих под управлением операционной системы Android.

Продажа вашего приложения через ваш собственный сайт

Вы распространять Android-приложения непосредственно через ваш собственный веб-сайт или сервер. Данный вариант наиболее целесообразен для распространения приложении для вертикального рынка, для компаний развивающих торговые площадки для мобильною контента, для веб-сайтов крупных брендов, которые хотят предоставлять своим пользователям брендированные Android - приложения. Этот вариант также может оказаться хорошим способом установления обратной связи с конечными пользователями

И хотя самостоятельное распространение приложений, — возможно, самый простой вариант, он также может оказаться наиболее сложным с точки зрения маркетинга, защиты прав и получения дохода. Для самостоятельного распространения нужно выполнить всего одно условие - найти подходящее место для размещения файла пакета приложения.

Есть еще одно дополнительное условие: настройки устройства, на котором конечный пользователь будет запускать ваше приложение, должны допускать загрузке пакетов из неизвестных источников. Данный параметр доступен на экране **Application Settings** (Настройки приложения) приложения **Settings** (Настройки), как показано на рис. 24.5

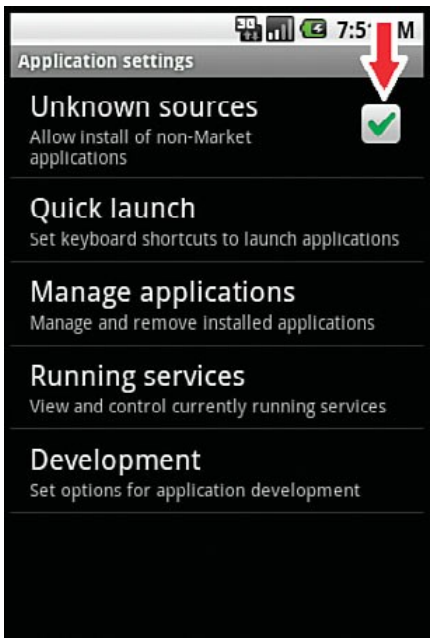

Рис. 24.5. Экран Application Settings (Настройки приложения) с параметром, отвечающим за возможность загрузки пакетов из известных источников

После этого пользователь должен ввести адрес URL пакета приложения в веб-браузере на мобильном телефоне и выгрузить данный файл (или нажать на ссылку на этот файл). Как только файл будет выгружен на телефон, запустится стандартный процесс установки Android – приложений, в течении которого пользователю нужно будет подтвердить разрешения, запрашивающие приложением, и, в некоторых случаях, подтвердить операцию обновления или замещения существующего приложения, если на телефоне уже установлена другая версия загруженного приложения.

ВНИМАНИЕ!

Не все устройства предоставляют настройку для включения возможности установки приложений из неизвестных источников. Например, у планшета ARCHOS 5 Internet Tablet
отсутствует такая настройка, однако (к счастью) данная функциональность автоматически включена, поэтому пользователи могут устанавливать приложения из любого выбранного ими источника. Вы должны учитывать эти различия между устройствами при подготовке инструкций для пользователей.

Продажа вашего приложения через другие сервисы

Сервис Android Market — не единственный консолидированный рынок продаж Android-приложений. Поскольку платформа Android — это открытая платформа, ничто не мешает производителю мобильного телефона или оператору (или даже вам) запустить собственный веб-сайт для продажи Android-приложений или разработать Android-приложение, которое будет представлять подобную торговую площадку.

Вот несколько торговых площадок, которые можно использовать для распространения ваших Android-приложений:
PocketGear — этот сайт позволяет распространять приложения для широкою спектра мобильных устройств, используя различные модели осуществления расчетов (**http://www.pocketgear.com**).

SlideME — это сообщество пользователей устройств, работающих под управлением операционной системы Android, и торговая площадка для распространения бесплатных и коммерческих приложений, для доступа к которой на мобильном устройстве устанавливается соответствующее клиентское приложение (**http://slideme.org**).

AndAppStore — этот сайт предоставляет возможности для распространения бесплатных Android-приложений, для доступа к которому на мобильном устройстве устанавливается соответствующее клиентское приложение (**http://www.andappstore.com**).

SHOP4APPS — эта торговая площадка компании Motorola, ориентированная на рынок собственных мобильных телефонов Android в Китае (**http://developer.motorola.com/shop4apps/**).

MobiHand — лот сайт распространяет бесплатные и коммерческие приложения для широкого спектра мобильных устройств (**http://www.mobihand.com**).

Этот список не полон, и мы специально не перечисляли преимущества одной торговой площадки перед другой, однако важно понимать, что в распоряжении разработчика есть ряд альтернативных механизмов для распространения их приложений. Помимо перечисленных, многие производители мобильных устройств и операторы сотовой имеют свои собственные торговые площадки, главным образом для устройств, которые не имеют установленного пакета приложений Google Experience (т. е., для тех устройств, на которых не установлены приложения Google, в том числе сервис Android Market).

Сторонние торговые площадки могут самостоятельно устанавливать любые требования к публикуемым приложениям, поэтому внимательно читайте конкретные условия на каждом сайте. Например, отдельные сайты могу? устанавливать требования к контенту приложения, требовать предоставления дополнительной технической поддержки и предъявлять специфически условия к цифровой подписи. Только вы и ваша команда можете определить, какие сайты подходят для ваших конкретных нужд.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Любой человек может разработать новую торговую площадку для распространения Android-приложений на своих собственных условиях. Это одно из преимуществ окрытой и бесплатной платформы.

ИТОГИ

В этом заключительном часе вы узнали, как опубликовать Android-приложение, чтобы его могли найти и использовать пользователи по всему миру. Теперь вам известно, что для публикации приложений существует несколько различных подходов, включая самостоятельное распространение приложений через ваш веб-сайт, а также множество сторонних торговых площадок, которые помогут вам продать вашу работу (обычно за

часть вознаграждения, полученного от продажи приложения). Вы также узнали, как создать аккаунт разработчик в сервисе Android Market — одной из наиболее популярных торговых площадок — и начать продавать ваши собственные приложения через этот сервис.

Возможно, у вас уже появились отличные идеи для воплощения в ваших приложениях. Запускайте среду разработки Eclipse и начинайте кодировать! Когда вы приступите к разработке приложений, отправьте нам сообщение с кратким рассказом о вашем приложении. (Наша контактная информация представлена в приложении B.) Мы будем с нетерпением ждать ваших сообщений!

ВОПРОСЫ И ОТВЕТЫ

Вопрос: Какие языки поддерживаются сервисом Android Market?

Ответ: В настоящее время сервис Android Market поддерживает более языков, и регулярно добавляется поддержка новых языков. Вот некоторые из языков, поддерживаемые сервисом Android Market в настоящий момент.

- американский вариант английского языка (en_US);
- французский язык/ Francais (fr_FR) ;
- немецкий язык/ Deutsch (de_DE);
- итальянский язык/ Italiano (it_IT);
- испанский и язык/ Уызфтщд (es_ES);
- голландский язык/ Nederlands (nl_NL);
- польский язык/ Polski (pl_PL);
- чешский язык/ Ceske (cs_CZ);
- португальский язык/ Portugues (pt_PT);
- тайваньский язык/ (zh_TW)
- японский язык/ (ja_JP);
- корейский язык/ (ko_KR);
- русский язык/русский (ru_RU).

Вопрос: Как я могу защитить результаты своего упорного труда от про граммного пиратства?

Ответ: Потратив время, деньги и усилия на создание ценного Android- приложения, имеет смысл защитить себя и результаты своего труда от декомпиляции кода с целью кражи используемых технологий и от программного пиратства. Поскольку Android-приложения компилируются для виртуальной машины Dalvik, большинство традиционных Java-инструментов для «запутывания» кода окажутся непригодными в данной ситуации. Тем не менее, платформа Android поддерживается некоторыми инструментами, например инструментом ProGuard (**http://proguard.sourceforge.net/**). На экране публикации приложения в сервисе Android Market также присутствует таинственный (недокументированный) флажок, который позволяет включить защиту от копирования вашего приложения.

ПРАКТИКУМ

Контрольные вопросы

1. Верно ли это? Для распространения приложений через сервис Android Market нет необходимости создавать аккаунт.

2. Какие из следующих утверждений верны?

A. Сервис Android Market пошатяет распространять как платные и бесплатные приложения.
B. Сервис Android Market позволяет разработчикам продавать приложения только на территории Соединенных Штатов.
C. Сервис Android Market - это единственная торговая площадка для распространении Android-приложений.
D. Сервис Android Market взимает операционный сбор, равный 30$ с каждой продажи приложения.
E. Все вышеперечисленные.

3. Верно ли это? Вы можете продавать Android-приложения через ваш собственный веб-сайт.

4. Перед тем, как загрузить приложение в сервис .Android Market, какие из перечисленных действий необходимо выполнить?

A. Сертифицировать ваше приложение в соответствии с утвержденное дорогостоящей программой сертификации.
B. Предоставить нотариально заверенный Акт о проведении тестирования, подтверждающий, что вы протестировали каждый аспект приложения во всех сценариях.
C. Подписать пакет вашего приложения с использованием подтоке, полученной от широко известного сертифицирующего органа, утвержденного сервисом .Android Market.
D. Записать видео, демонстрирующее работу вашего приложения.
E. Предоставить Word-документ с подробным описанием функциональности приложения и полным ру ководством пользователя.
F. Получить письменное разрешение от каждого оператора сотою! связи, в сети которого вы планируете использовать ваше приложение. перед загрузкой пакета вашего приложения.

Ответы

1. Неверно. Вы должны создать удостоверенный аккаунт разработчик Google перед тем. как приступить к публикации Android-приложений в сервисе Android Market.

2. A и D. Сервис Android Market — самая популярная торговая плошав для распространения Android-приложений — позволяет разработчикам публиковать бесплатные и платные приложения в ряде стран, а за размещение приложений взимается операционный сбор, равный 30% с каждой продажи приложения.

3. Верно. Вы можете продавать ваши Androtd-приложения через различные Интернет-магазины приложений, включая ваш собственный Имейте в виду, что пользователи должны включить в настройках мобильного телефона возможность установки приложений из неизвестных источников.

4. Ничего из перечисленного! Но, несмотря на то, что ни одно из /тих требований не обязательно, некоторые можно принять в качестве совета. например, тщательное тестирование вашего приложения Другие могут оказаться полезными для маркетинговых целей. Тем не менее пя сервиса Android Market не требуется выполнение ни одного из этих требований. Сервис Android Market чрезвычайно открыт!

Упражнения

1. Создаете свой собственный аккаунт разработчика в сервисе Android Market.

2. Ознакомьтесь с приложениями, доступными в сервисе Android Market (на мобильном телефоне или на веб-сайте сервиса Android Market). Придумайте идею для вашего будущего приложения и определите категорию и ценовой диапазон для этого приложения в сервисе Android Market.

3. Разработайте изумительное и захватывающее приложение, и пусть о нем узнает весь мир.

ЧАСТЬ VI. ПРИЛОЖЕНИЯ

Приложение А. НАСТРОЙКА СРЕДЫ ДЛЯ РАЗРАБОТКИ ANDROID-ПРИЛОЖЕНИЙ

В этом приложении описываются шаги по установке и настройке всех необходимых инструментов, которые понадобятся вам для разработки Android- приложений:

- подходящая версия инструментария Java Development Kit (JDK);
- интегрированная среда разработки Eclipse (IDE);
- инструментарий Android Software Development Kit (SDK) и другие инструменты;
- драйверы, необходимые для конкретных устройств Android.

Эти пакеты программного обеспечения доступны для бесплатной загрузки с веб-сайтов их производителей. Для начала работы вы можете использовать конкретные версии перечисленных инструментов, доступные на диске, прилагаемом к данной книге.

ТРЕБОВАНИЯ К КОМПЬЮТЕРУ РАЗРАБОТЧИКА

Разработчики Android-приложений могут использовать ряд различных операционных систем и конфигураций программного обеспечения. В этом приложении описывается процесс установки инструментов, используемых в данной книге. Если вы устанавливаете вашу среду для разработки с нуля, имеет смысл выбрать самые последние версии пакетов программного обеспечения, необходимых для разработки.

Полный список требований к программному обеспечению и системных требований можно найти на веб-сайте, посвященном разработке на платформе Android, по адресу **developer.android.com/sdk/requirements.html**.

Android-приложении можно разрабатывать в следующих системах:

- Windows XP и более поздние версии;
- Mac OS X 10.5.8 и более поздние версии (только для архитектур x86):
- Linux.

Доступное дисковое пространство

Вам потребуется около 2 Гб свободного дискового пространства для успешной установки всех инструментов, которые требуются для разработки Android-приложений. Сюда относится установка инструментария JDK. интегрированной среды разработки Eclipse, инструментария Android SDK и других инструментов и плагинов.

УСТАНОВКА ИНСТРУМЕНТАРИЯ JAVA DEVELOPMENT KIT

Для разработки Android-приложений могут применяться инструментарии JDK 5 и JDK 6 компании Oracle. Ознакомиться с лицензионным соглашением и загрузить последнюю версию инструментария Java Standard Edition JDK можно на веб-сайте компании Oracle по адресу **www.oracle.conn/technetwork/java/javase/downloads/index.html** (инструментарий Java Standard Edition JDK можно найти на диске, прилагаемом к данной книге). Инструкции по установке инструментария в вашей операционной системе можно найти в документации, поставляемой вместе с выбранным вами пакетом установки.

УСТАНОВКА ИНТЕГРИРОВАННОЙ СРЕДЫ РАЗРАБОТКИ ECLIPSE

Большинство разработчиков предпочитает использовать популярную интегрированную среду разработки Eclipse для разработки Android-приложений; эта среда разработки доступна для операционных систем Windows, Mac и Linux. Вы можете разрабатывать Android-приложения с использованием либо среды разработки Eclipse 3.4 (Ganymede), либо среды разработки Eclipse 3.5 (Galileo).

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Если вы будете работать со средой разработки Eclipse впервые, возможно, имеет смысл остановить свой выбор на версии Eclipse IDE for Java EE Developers. Эта версия среды разработки Eclipse поставляется вместе с плагином Eclipse Java Development Tools (JDT) и необязательным компонентом Web Tools Platform IWTP).

Вы можете ознакомится с лицензионным соглашением и загрузить версию Eclipse IDE for Java EE Developers по адресу **www.eclipse.org/downloads/** (версия Eclipse IDE for Java EE Developers (Galileo SR2 v.3.5.2) доступна на диске, прилагаемом к данной книге)

Пакет среды разработки Eclipse поставляется и виде заархивированного файла. Мастер установки для среды Eclipse отсутствует. Вы просто разархивируйте содержимое пакета и желаемую папку и затем следуете перечисленным ниже инструкциям для вашей операционной системы.

Замечания по установке в операционной системе Windows

Разархивировав файлы в желаемое местоположение, найдите исполняема файл **Eclipse.exe** и создайте ярлык на вашем **Рабочем столе**. Откройте диалоговое окно со свойствами созданного ярлыка и при необходимости измените значение параметра **Объект** (Target), указав любые желаемые аргументы командной строки.

Замечания по установке в операционной системе Mac OS X

Если вы устанавливаете среду разработки Eclipse в операционной системе Mac OS X. обязательно прочтите файл **README.html**, входящий в состав пакета среды разработки Eclipse. В этом файле описывается, как передавать аргументы командной строки в среду разработки Eclipse при помощи файла eclipse.ini и как запускать более одного экземпляра этой среды разработки, чтобы вы могли одновременно работать с несколькими проектами.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

Если вы не желаете использовать среду разработки Eclipse, вы можете найти дополнительную информацию по настройке вашего компьютера для разработки Android-приложений с использованием других интегрированных сред разработки на веб-сайте, посвященном разработке на платформе Android, по адресу **developer.android.com/guide/developing/other-ide.html**.

УСТАНОВКА ИНСТРУМЕНТАРИЯ ANDROID SDK

Для разработки Android-приложений вам необходимо установить инструментарий Android SDK. Инструментарий Android SDK включает JAR-файл платформы Android (классы для разработки Android-приложений), а также документацию по платформе Android, инструменты и файлы с исходными кодами примеров.

Последнюю версию инструментария Android SDK можно выгрузить с веб - сайта, посвященного разработке на платформе Android, по адресу developed android.com/sdk/index.html (установочный пакет приложения Android SD and AVD Manager, который используется, в том числе, и для загрузки инструментария Android SDK, доступен на диске, прилагаемом к данной кииге). Вам нужно принять лицензионное соглашение Android перед установкой пакета разработчика.

Новейшие версии и инструментария Android SDK имеют удобную программу установки. Вы просто выгружаете заархивированный файл с сайта, извлекаете его содержимое в желаемую папку и запускаете программу установки. Заархивированные файлы приложения Android SDK and AVD Manager занимают около 25 Мб дискового пространства, а после разархивации — около 40 Мб.

Инструменты и версии инструментария SDK разделены на компоненты. Это значит, что вместо установки одного огромного пакета для разработки приложений для всех поддерживаемых версий платформы Android, вы можете выбрать только те версии инструментария Android SDK, которые хотите установить и использовать, при помощи приложения Android SDK and AVD Manager. Данный инструмент позволяет разработчикам легко обновлять свою среду разработки при выходе новой версии платформы Android (что раньше случалось достаточно часто). Помимо того, что вы можете выбирать различные целевые версии платформы Android, при помощи этого приложения также можно загрузить другие инструменты и сопутствующее программное обеспечение, например, USB-драйверы для операционной системы Windows.

Для установки необходимых инструментов лучше всего использовать приложение Android SDK and AVD Manager.

УСТАНОВКА ИНСТРУМЕНТАРИЯ ANDROID SDK БЕЗ ПОДКЛЮЧЕНИЯ К ИНТЕРНЕТУ

Несмотря на то, что для установки инструментария Android SDK настоятельно рекомендуется использовать приложение Android SDK and AVD Manager и загружать необходимые версии инструментария и другие инструменты через Интернет, существует

возможность установки инструментария Android SDK без подключения к Интернету. На диске, прилагаемом к данной книге, в папке **Android SDK**, которая находится внутри папки с названием вашей операционной системы (**Windows**, **Linux** или **Mac OS**), можно найти файлы пакетов для приложения Android SDK and AVD Manager и для инструментария Android SDK версии 2.1 (описанная ниже процедура может быть использована и для других версий инструментария). Чтобы установить инструментарий Android SDK без подключения к Интернету, выполните следующие шаги 1на примере операционной системы Windows):

1. Разархивируйте содержимое файла **/Windows/Android SDK/android-sdk_r08- windows.zip**, находящегося на диске, прилагаемом к данной книге, в папку на жестком диске вашего компьютера (например, в папку **C:\Program Files\AndroidSDK**). В этом архиве находится приложение Android SDK and AVD Manager.

2. Разархивируйте содержимое файла **/Windows/Android SDK/android-sdk_r08- windows.zip** в папку **platforms**, которая находится внутри разархивированной папки **android-sdk-windows** (в нашем случае, **C:\Program Files\AndroidSDK\ android-sdk-windows\platforms**). Данный архив содержит инструментарий Android SDK 2.1.

3. Откройте папку **android-sdk-windows** и создайте внутри нее папку с именем **platform-tools**. Разархивируйте содержимое папки **platform-tools_r01-windows** из архива **/Windows/Android SDK/platform-tools_r01-windows.zip** в созданную папку.

4. Чтобы установить дополнение Google API для инструментария Android SDK 2.1, разархивируйте содержимое файла **/Common/Android SDK/google_apis-7_r01.zip** в папку **add-ons**, находящуюся внутри паки **android-sdk-windows**.

5. Чтобы установить файлы примеров для инструментария Android SDK 2.1, откройте папку **android-sdk-windows** и создайте внутри нее папку с именем **samples**. Разархивируйте содержимое файла **/Common/Android SDK/samples-2.1_r01-linux.zip** в созданную папку (пусть вас не смущает название этого файла – файлы примеров являются общими для всех операционных систем).

Теперь, если вы запустите приложение Android SDK and AVD Manager, вы увидите, что инструментарий Android SDK 2.1, дополнение Google API и файлы примеров были успешно установлены.

Замечания по установке в операционной системе Windows

Чтобы обновить значение переменной **PATH**, включив в него путь к папке **tools** инструментария Android SDK, щелкните правой кнопкой по значку **Мой компьютер** (My Computer) и в появившемся контекстном меню выберите команду Свойства (Properties). В операционной системе Vista вам также нужно будет щелкнуть по ссылке **Дополнительные параметры системы** (Advanced System Settings). Далее щелкните по вкладке **Дополнительно** (Advanced) в открывшемся диалоговом окне **Свойства системы** (System Properties) и затем нажмите на кнопку **Переменные среды** (Environment Variables).

В группе элементов управления **Системные переменные** (System Variables) появившегося диалогового окна **Переменные среды** (Environment Variables) найдите переменную **PATH** и измените ее значение, добавив путь к папке **tools** инструментария Android SDK.

Замечания по установке в операционной системе Mac OS X

Чтобы обновить значение переменной PATH, включив в него путь к каталогу **tools** инструментария Android SDK, вам потребуется внести изменения в файл **.bash_profile**, находящийся в каталоге **Home**.

Замечания по установке в операционной системе Linux

Чтобы обновить значение переменной **PATH**, включив в него путь к каталогу **tools** инструментария Android SDK, вам потребуется внести изменения в ваш файл **-/.bash_profile**, **-/.bashrc** или **-/.profile**.

УСТАНОВКА И НАСТРОЙКА ПЛАГИНА ANDROID ДЛЯ СРЕДЫ РАЗРАБОТКИ ECUPW (ADT)

Плагин Android для среды разработки Eclipse позволяет обеспечить полную интеграцию со многими инструментами платформы Android. Если вы используете среду разработки Eclipse, мы настоятельно рекомендуем установить данный плагин, поскольку он значительно упростит вашу жизнь. Этот плагин предоставляет различные мастера для создания и отладки Android- проекта.

Чтобы установить плагин Android для среды разработки Eclipse (ADT), вы должны запустить среду разработки Eclipse и произвести установку обновления пользовательского программного обеспечения. Необходимые шаги зависят от используемой версии среды разработки Eclipse. Полную инструкцию можно найти на веб-сайте, посвященном разработке на платформе Android, по адресу **developer.android.com/sdk/eclipse-adt.html**.

Чтобы установить плагин Android для среды разработки Eclipse версии 3.5 (Galileo), выполните следующие шаги:

1. Запустите среду разработки Eclipse.
2. Выберите команду меню **Help** => **Install New Software** (Справка - Установить новое программное обеспечение).
3. В открывшемся диалоговом окне **Install** (Установка) нажмите на кнопку **Add** (Добавить). На экране появится диалоговое окно **Add Site** (Добавление сайта).
4. Добавьте удаленный сайт с адресом URL **https://dl-ssl.google.com/android/eclipse/**. Если этот адрес не работает, попробуйте использовать адрес URL **http://dl-ssl.google.com/android/eclipse/**.
5. В диалоговом окне **Install** (Установка) установите флажок напротив элемента списка **Developer Tools**. (Если развернуть этот элемент списка, вы

увидите, что флажки также будут установлены напротив элементов **Android DDMS**, **Android Development To**ols и **Android Hierarchy Viewer**.)

6. Нажмите на кнопку **Next** (Далее) и следуйте указаниям мастера, чтобы установить инструменты. Примите условия лицензионного соглашения и нажмите на кнопку **Finish** (Готово).

7. Когда обновление программного обеспечения будет завершено, перезапустите среду разработки Eclipse.

УСТАНОВКА ПЛАГИНА ANDROID ДЛЯ СРЕДЫ РАЗРАБОТКИ ECLIPSE БЕЗ ПОДКЛЮЧЕНИЯ К ИНТЕРНЕТУ

Вы можете установить плагин Android для среды разработки Eclipse без подключения к Интернету. Для этого выполните следующее действие (на примере среды разработки Eclipse версии 3.5 (Galileo)):

1. Запустите среду разработки Eclipse.

2. Выберите команду меню **Help**=>**Install New Software** (Справка => Установить новое программное обеспечение).

3. В открывшемся диалоговом окне **Install** (Установка) нажмите на кнопку **Add** (Добавить). На экране появится диалоговое окно **Add Site** (Добавление сайта).

4. Нажмите на кнопку **Archive** (Архив).

5. Найдите файл **/Common/ADT Plugin for Eclipse/ADT-8.0.1.zip** на диске, прилагаемом к данной книге, и выберите его. При желании введите значение в поле ввода **Name** (Имя).

6. Нажмите на кнопку **OK**.

7. В диалоговом окне **Install** (Установка) установите флажок напротив элемента списка **Developer Tools**. (Если развернуть этот элемент списка, вы увидите, что флажки также будут установлены напротив элементов **Android DDMS**, **Android Development Tools** и **Android Hierarchy Viewer**.)

8. Нажмите на кнопку **Next** (Далее) и следуйте указаниям мастера, чтобы установить инструменты. Примите условия лицензионного соглашения и нажмите на кнопку **Finish** (Готово).

9. Когда обновление программного обеспечения будет завершено, перезапустите среду разработки Eclipse.

После того, как вы установите плагин Android для среды разработки Eclipse, вам потребуется обновить настройки среды разработки Eclipse, чтобы указать местоположение, куда ранее был установлен инструментарий Android SDK. Для этого запустите среду разработки Eclipse и выберите команду меню **Window** => **Preferences** (Окно => Настройки) (или команду меню **Eclipse** => **Preferences** (Eclipse => Настройки) в операционной системе Mae OS X). В открывшемся диалоговом окне **Preferences** (Настройки) выберите раздел Android и укажите путь к папке, в которую был установлен инструментарий Android SDK. Указав правильный путь, вы увидите в списке, расположенном под полем ввода **SDK Location** (Местоположение SDK) диалогового окна **Preferences** (Настройки), ряд целевых версий инструментария SDK (Android 1.0, 1.5, 1.6, 2.0, 2.01, 2.1 и т. д.).

ОБНОВЛЕНИЕ ИНСТРУМЕНТАРИЯ ANDROID SDK

Инструментарий Android SDK в настоящее время находится в стадии разработки, а это значит, что вам неминуемо придется обновить версию инструментария SDK, установленную на вашем компьютере. Изменения инструментария Android SDK могут включать добавление, обновление и удаление возможностей, изменение названий пакетов и обновление инструментов.

Вместе с каждой новой версией инструментария SDK компания Google поставляет следующие полезные документы:

- **Обзор изменений** - краткое описание основных изменений в данной версии инструментарии SDK.
- **Отчет об отличиях в интерфейсе API** - полный список конкретных в данной версии инструментария SDK.
- **Замечания к выпуску** — список известных проблем, исправленных и данной версии инструментария SDK.

Чтобы обновить инструментарий Android SDK, необходимо япуиить приложение Android SDK and AVD Manager (в среде разработки Eclipse можно щелкнуть по небольшому значку с изображением андроида яти выбрать команду меню **Window** => **Android SDK and AVD Manager** (Окно => Android SDK and AVD Manager)) и выполнить обновление всех пакетов, а также проверить наличие новых пакетов. Обновление инструментария Android SDK включает обновление целевых версий платформы Android (и AVD- эмуляторов) в среде разработки Eclipse, обновление переменных пути и. при необходимости, перенастройки существующих инструментов для разработки Android-приложений. Когда вы обновите среду для разработки, вам понадобится перевести ваши Android-приложения на новую версию инструментария SDK.

НАСТРОЙКА АППАРАТНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОТЛАДКИ ПРИЛОЖЕНИЙ НА НАСТОЯЩИХ УСТРОЙСТВАХ

Каждая модель телефона Android может иметь разнообразные отладочные конфигурации. На вашем устройстве Android должна быть включена возможность выполнения отладки приложений через подключение к порту USB.

Включение на устройстве Android возможности выполнения отладки через подключение к порту USB

Чтобы включить возможность отладки через подключение к порту USB. находясь на домашнем экране вашего устройства Android, нажмите кнопку **Menu**, после чего запустите приложение **Настройки** (Settings), выберите в списке элемент Приложения (Applications), затем - элемент **Разработка** (Development), и установите флажок **Отладка по USB** (USB Debugging). На Различных устройствах эта настройка может находиться в разных местах, апример, чтобы включить возможность отладки через подключение к пор- USB на планшете Archos 5 Internet Tablet, необходимо выбрать элемент списка **Device Storage & USB connection** (Память устройства и подключение по USB), затем – элемент списка **USB Connection Mode** (Режим подключения по USB) и, наконец, элемент списка **Debug Bridge (ADB)** (Отладочный мост (ADB))

Обзор изменений краткое описание основных шмсн-и/г .. .)й,!нои версии инструментарии SDK.
Отчет об отличиях в интерфейсе API - полный список кмчулнш и ж- нений в данной версии инструментария SDK.
Замечания к выпуску — список известных проблем, исправленных и данной версии инструментария SDK.

ЗНАЕТЕ ЛИ ВЫ, ЧТО...

В ходе длительных отладочных сессий ваш телефон может часто переходить в режим сна. Чтобы избежать подобного поведения, включите возможность, которая позволяет телефону оставаться в активном состоянии во время зарядки. Соответствующая настройка должна находиться среди параметров, связанных с разработкой приложений, и обычно называется «**Оставить включенным**» (Stay awake) или
как-то похоже.

Настройка вашей операционной системы для отладки на реальном устройстве

Чтобы устанавливать и отлаживать Android-приложения на таком аппаратном обеспечении, как, например, телефонах T-Mobile Gl, Motorola Droid или Nexus One, вам может потребоваться настроить вашу операционную систему для получения доступа к телефону через подключение к порту USB. Это особенно касается компьютеров под управлением операционной системы Windows. Инструментарий Android SDK поставляется вместе с драйверами для некоторых устройств, и вы можете подключить телефон к порту USB и указать в диалоговом окне **Диспетчер устройств** (Device Manager) каталог, в которую был установлен инструментарий Android SDK, после чего вы сможете запускать приложения на телефоне из среды разработки Eclipse.

ЗАМЕЧАНИЯ ПО НАСТРОЙКЕ ОПЕРАЦИОННОЙ СИСТЕМЫ WINDOWS

Вы должлы установить USB-драйверы Android. Они доступны в виде одного из пакетов в приложении Android SDK and AVD Manager под названием **USB Driver Package**. В качестве альтернативы вы можете загрузить пакет драйверов непосредственно на странице **developer.android.com/sdk/win-usb.html**. После того, как вы разархивируете драйверы, подключите ваш телефон к компьютеру с использованием USB-кабеля и выберите драйверы, которые вы желаете установить.

ЗАМЕЧАНИЯ ПО НАСТРОЙКЕ ОПЕРАЦИОННОЙ СИСТЕМЫ MAC OS X

В поддерживаемой операционной системе Mac все, что вам нужно сделать, — это подключить телефон к компьютеру Mac при помощи USB- кабеля. Никакой дополнительной настройки не требуется.

ЗАМЕЧАНИЯ ПО НАСТРОЙКЕ ОПЕРАЦИОННОЙ СИСТЕМЫ LINUX

Для операционной системы Ubuntu Linux необходимо создать rules-файл, выполнив следующие шаги:

1. Войдите в систему под учетной записью администратора (root).

2. Создайте файл **/etc/udev/rules.d/50-android.rules**.
3. Добавьте в созданный файл следующее содержимое:

    1. Для версий Gutsy (7.10) и Hardy (8.04):
    ```
    SUBSYSTEM==”usb”, SYSFS{idVendor}==”0bb4”, MODE=”0666”
    ```

    2. Для версии Dapper (6.06):
    ```
    SUBSYSTEM==”usb_device”, SYSFS{idVendor}==”0bb4”, MODE=”0666”
    ```

4. Выполните команду `a+rx /etc/udev/rules.d/50-android.rules`.

Приложение Б. ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ ECLIPSE - СОВЕТЫ И ХИТРОСТИ

В этом приложении собрано множество советов и хитростей, которые вы можете взять на вооружение для эффективной работы со средой разработки Eclipse. Эти советы и хитрости связаны с выполнением распространенных задач при разработке Android-приложений, однако они могут также применятся при разработке Java-приложений в среде разработки Eclipse.

КСТАТИ

У вас есть собственные советы и хитрости, связанные с разработкой Android- приложений е среде разработки Eclipse? Если так, то пришлите их на наш адрес электронной почты **androidwirelessdev@gmail.com** (вместе с разрешением на их публикацию! и мы добавим ваши советы и хитрости в наш блог по адресу **endroidbook.blogspot.com** . Это ваш шанс заработать славу фаната программирования!

СОЗДАНИЕ НОВЫХ КЛАССОВ И МЕТОДОВ

Вы можете быстро создать новы»» класс и соответствующий файл с исходным кодом, щелкнув правой кнопкой мыши по названию пакета, в котором должен быть создан этот класс, и выбрав в появившемся контекстном меню команду **New** => **Class** (Создать => Класс). В появившемся диалоговом окне **New Java Class** (Новый класс Java) введите имя создаваемого класса, выберите его суперкласс и интерфейсы, а также укажите, нужно ли создавать стандартные комментарии и методы-заглушки для конструкторов или абстрактных методов суперкласса.

Аналогичным образом вы можете быстро создать методы-заглушки, шелк- н>в правом кнопкой мыши по названию класса или в окне редактора с исходным кодом класса и выбрав в появившемся контекстном меню команду **Source** => **Override/Implement Methods** (Исходный код => Переопределить/ Реапнзовать методы). В открывшемся диалоговом окне **Override/Implement Methods** (Переопределение/Реализация методов) выберите методы, для которых нужно создать «заглушки», и место в исходном коде дня добавления методов-заглушек, а также укажите, нужно ли генерировать стандартные блоки комментариев.

ОРГАНИЗАЦИЯ ИНСТРУКЦИЙ IMPORT

Когда вы в первый раз обращаетесь к некоторому классу в вашем коде, вы можете установить указатель мыши на название этого нового класса и в появившемся всплывающем меню щелкнуть по ссылке «**Import** “имя_класса” (название пакета)» (Импортировать, имя_класс' (название пакета)), чтобы среда разработки Eclipse быстро добавила подходящую инструкцию import.

Помимо того, команда **Organize imports** (Организовать импорты) (комбинация клавиш **Ctrl+Shift+O** в операционной системе Windows или комбинация клавиш **Cmd+Shift+O** в операционной системе Mae) заставляет среду разработки Eclipse автоматически организовать инструкции import. Среда рачработки Eclipse удаляет неиспользуемые инструкции import и добавляет новые инструкции для используемых пакетов, которые еше не были импортированы.

Если существует неопределенность в выборе пакета для некоторого класса в процессе автоматического импортирования пакетов, например, как в случае с классом Log платформы Android, среда разработки Eclipse предложит вам выбрать нужный пакет для импортирования.

Наконец, вы можете настроить среду разработки Eclipse таким образом, чтобы она автоматически организовывала инструкции import при каждом сохранении файла. Эта настройка может быть установлена как для всего рабочего пространства, гак и для отдельного проекта. Настройка данной возможности на уровне отдельного проекта обеспечивает более высокую гибкость r тех случаях, когда вы работаете над несколькими проектами и не хотите вносить никаких изменений в некий код. даже если эти изменения являются улучшениями. Чтобы настроить данную возможность, выполните следующие шаги:

1. Щелкните правой кнопкой мыши по названию проекта и в появившемся контекстном меню выберите команду **Properties** (Свойства).

2. В появившемся диалоговом окне разверните элемент списка **Java Editor** (Редактор Java) и выберите раздел **Save Actions** (Действия при сохранении).

3. Установите флажок **Enable Project Specific Settings** (Включить настройки на уровне проекта), после этого — флажок **Perform the Selected Actions on Save** (Выполнять выбранные действия при сохранении), и. наконец, флажок **Organize Imports** (Организовать импорты).

ДОКУМЕНТИРОВАНИЕ КОДА

Польза от комментариев, регулярно встречающихся в коде, несомненна (при условии, что эти комментарии написаны толково). Комментарии в стиле Javadoc применяются во всплывающих диалоговых окнах автозаполнения кода и в других местах, делая их еще более полезными. Чтобы быстро добавить комментарий в стиле Javadoc к методу или к классу, просто нажмите комбинацию клавиш **Ctrl+Shift+J** (Windows) или **Cmd+Alt+J** (Mac). В качестве альтернативного варианта вы можете выбирать в контекстном меню команду **Source=>Generate Element Comment** (Исходный код=>Сгенерировать комментарий к элементу), чтобы автоматически добавить определенные поля в Javadoc-комментарии, например, названия параметров и имя автора, тем самым значительно повысив скорость создания подобных комментариев

ИСПОЛЬЗОВАНИЕ ФУНКЦИИ АВТОЗАПОЛНЕНИЯ

Автозаполнение — это потрясающая возможность, которая ускоряет процесс ввела текста. Если вы еще не видели соответствующее всплывающее диалоговое окно дан-

ной возможности или если это окно закрылось, вы можете отобразить его на экране, нажав комбинацию клавиш **Ctrl+Пробел**.
Автозаполнение помогает не только сэкономить время в процессе ввода текста, но также позволяет освежить в вашей памяти сведения о некоторых методах — или найти новый метод. Вы можете просмотреть список всех методов отдельного класса и даже увидеть связанные с ними Javadoc- комментарии. Вы можете легко найти статические методы, используя имя класса или имя переменной экземпляра. Для этого введите нужное имя, поставьте точку (и, возможно, нажмите комбинацию клавиш **Ctrl+Пробел**) и прокрутите список всех доступных имен. После этого вы можете начать ввод первой части имени, чтобы отфильтровать результаты.

ЭФФЕКТИВНОЕ РЕДАКТИРОВАНИЕ КОДА

В некоторых случаях вам может показаться, что окно редактора имеет слишком маленькие размеры, особенно, когда вокруг него расположено множество дополнительных маленьких панелей и вкладок. Попробуйте сделать следующее: дважды щелкните по вкладке исходного файла, который вы хотите изменить. Бум! Теперь окно редактора занимает почти все пространство окна среды разработки Eclipse! Снова дважды щелкните по вкладке, и окно редактора примет первоначальные размеры.

Вам никогда не хотелось видеть одновременно код из двух исходных файлов? Ну, это возможно! Просто щелкните по вкладке исходного файла и, удерживая кнопку мыши, перетащите его к любому краю окна редактора. Вы должны увидеть темный контур, обозначающий новое местоположение окна редактора — либо бок о бок с другим файлом, либо выше или ниже другого файла. Когда вы отпустите кнопку мыши, будет создано второе окно редактора, в которое можно также перетаскивать другие вкладки файлов.

Вам никогда не хотелось видеть одновременно два разных фрагмент и одном исходном файле? Это возможно! Щелкните правой кипкой мыши по вкладке нужного файла и в появившемся контекстном меню выберите команду **New Editor** (Новый редактор), В результате появится вторая вкладка для одного и того же файла. Теперь, воспользовавшись предыдущим советом, вы можете иметь два разных представления одного и тою же файла.

Вам никогда не казалось, что у вас открыто слишком много вкладок для файлов, которые вы уже не редактируете? Мне казалось! Существует ряд решений для данной проблемы. Во-первых, вы можете щелкнуть правой кнопкой мыши по вкладке файла и выбрать в появившемся контекстном меню команду **Close Others** (Закрыть остальные), чтобы закрыть все остальные открытые файлы, кроме выбранного файла. Во-вторых, вы можете быстро закрыть конкретные вкладки, щелкнув средней кнопкой мыши по каждой вкладке. (Этот способ работает даже на компьютерах Мае, оснащенных мышью, позволяющей совершать щелчки средней кнопкой, например, мышью с колесиком прокрутки.) Наконец, вы можете использовать настройку среды разработки Eclipse, позволяющую ограничивать количество открытых редакторов:

1. Откройте диалоговое окно **Preferences** (Настройки) среды разработки Eclipse.
2. В открывшемся диалоговом окне **Preferences** (Настройки) разверните элемент списка **General** (Общие), выберите раздел **Editors** (Редакторы) и установите флажок **Close Editors Automatically** (Автоматически закрывать редакторы).

3. Измените значение параметра **Number of Opened Editors Before Closing** (Количество открытых редакторов перед закрытием).

На мой взгляд, значение 8 — это оптимальное значение для параметра **Number of Opened Editors Before Closing** (Количество открытых редакторов перед закрытием), позволяющее избежать нагромождения вкладок редакторов и при этом иметь достаточное число открытых редакторов для выполнения работы и для просмотра кода. Также стоит отметить, что, если вы установите переключатель **When All Editors Are Dirty or Pinned** (Когда все редакторы содержат несохраненный код или закреплены) в положение **Open New Editor** (Открывать новый редактор), будет открыто большее число окон редакторов в том случае, когда вы активно редактируете файлы, количество которых превышает значение, указанное для параметра **Number of Opened Editors Before Closing** (Количество открытых редакторов перед закрытием). Таким образом, эта настройка не оказывает влияния на производительность, когда вы одновременно редактируете большое количество файлов, но помогает поддерживать порядок в среде разработки в процессе выполнения большинства повседневных задач.

ПЕРЕИМЕНОВАНИЕ ПРАКТИЧЕСКИ ВСЕГО

Инструмент **Rename** (Переименовать) среды разработки Eclipse – очень мощный. Он может быть использован для переименования переменных, методов, классов и много другого. В большинстве случаев вы можете просто щелкнуть правой кнопкой мыши но элементу, который вы хотите переименовать, и выбрать в появившемся контекстном меню команду **Refacto**r=>**Rename** (Рефакторинг=>Перемеиновать). В качестве альтернативы после гою. как вы ш w апе нужный элемент, можно нажать комбинацию клавиш **ALT+Shift+R**, чтобы запустить процесс переименования. Если вы переименовываете класс верхнего уровня в файле, имя файла также должно быть изменено Если файл находится под управлением системы контроля исходною кода, среда разработки Eclipse обычно выполняет все необходимые действия, связанные с его переименованием.

Если среда разработки Eclipse может определить, что переименовываемый элемент связан с другим элементом, имеющим точно такое же имя, все вхождения данного имени также будут переименованы. В некоторых случаях это означает, что новое имя будет указано даже в комментариях. Довольно удобно!

ФОРМАТИРОВАНИЕ КОДА

Среда разработки Eclipse имеет встроенный механизм форматирования Java-кода. Форматирование кода при помощи данного механизма полезно для поддержания согласованности стиля кода, для применения нового стиля к старому коду или для приведения стилей в соответствие с требованиями различных клиентов или задач (например, для книги или статьи).

Чтобы быстро отформатировать небольшой фрагмент кода, выделите данный код и нажмите комбинацию клавиш **Ctrl+Shift+F** (Windows) или **Cmd+Shift+F** (Mac). Код будет отформатирован в соответствии с текущими настройками. Если никакой фрагмент кода не выделен, будет отформатирован весь код в файле. В некоторых случаях вам необходимо выделить больший фрагмент кода, например целый метод, чтобы сохранить уровни отступов и расположение фигурных скобок.

Настройки форматирования среды разработки Eclipse можно найти в диалоговом окне **Properties** (Свойства), в разделе **Formatter** (Форматтер) группы **Java Code Style** (Стиль кода Java). Вы можете установить эти настройки на уровне проекта или на уровне рабочего пространства. Вы можете применить и изменить десятки правил, чтобы код соответствовал вашему собственному стилю.

ОРГАНИЗАЦИЯ КОДА

В некоторых случаях форматирования кода недостаточно, чтобы сделать его понятным и удобочитаемым. В процессе разработки сложного действия появиться множество вложенных классов и методов, разбросанных по всему файлу. В этом случае вам поможет одна из хитростей разработки Eclipse: открыв нужный файл, убедитесь, что на экране также отображается панель **Outline** (Обзор). Просто перетаскивайте методы и классы на панели **Outline** (Обзор), чтобы разместить их в подходящем логическом порядке. У вас есть метод, который вызывается лишь из определенного класса, но при этом доступен всем другим классам. Просто перетащите этот метод в данный класс. Описанный подход работает практически со всеми элементами, доступными на панели **Outline** (Обзор), включая методы и переменные.

РЕФАКТОРИНГ С УДОВОЛЬСТВИЕМ

Вы никогда не замечали за собой, что написали много повторяющихся фрагментов кода, который выглядит, например, наподобие следующего кода?

```
TextView nameCol = new TextView(this);
nameCol.setTextColor(getResources().getColor(R.color.title_color));
nameCol.setTextSize(getResources().
    getDimension(R.dimen.help_text_size));
nameCol.setText(scoreUserName);
table.addView(nameCol);
```

Этот код устанавливает цвет текста, размер текста и сам текст. Если вы написали два или более фрагментов подобного кода, вашему коду может пойти на пользу рефакторинг. Среда разработки Eclipse предоставляет два крайне полезных инструмента — команды **Extract Local Variable** (Извлечь локальную переменную) и **Extract Method** (Извлечь метод) — для ускорения данного процесса и его максимального упрощения.

Чтобы воспользоваться командой **Extract Local Variable** (Извлечь локальную переменную), выполните следующие шаги:

Выделите выражение `getResources().getColcr(R.color.titie_ color)`. Щелкните правой кнопкой мыши и в появившемся контекстном меню выберите команду **Refactor**=>**Extract Local Variable** (Рефакторинг=>Извлечь локальную переменную) (или нажмите комбинацию клавиш **Ctrl+Alt+L**).

В открывшемся диалоговом окне введите имя переменной и, не сбрасывая флажок **Replace All Occurrenes** (Заменить нее вхождении), нажмите на кнопку **OK**, чтобы понаблюдать за волшебством.

Повторите шаги 1-3 для выражения, устанавливающего размер текста.

Полученным результат должен выглядеть следующим образом:

```
int textColor = getResources().getColor(R.color.title_color);
float textSize =
getResources().getDimension(R.dimen.help_text_size);
TextView nameCol = new TextView(this);
nameCol.setTextSize(textSize);
nameCol.setText(scoreUserName);
nameCol.setTextColor(textColor);
table.addView(nameCol);
```

Изменения также отразились и в оставшихся пяти строках кода. Как это удобно, не правда ли?

Теперь настало время испытать в действии второй инструмент. Выполните следующие шаги, чтобы воспользоваться командой **Extract Method** (Извлечь метод):

1. Выделите все пять строк первого фрагмента кода.

2. Щелкните правой кнопкой мыши и в появившемся контекстном меню выберите команду **Refactor**=>**Extract Method** (Рефакторинг => Извлечь метод) (или нажмите комбинацию клавиш **Ctrl+Alt+M**).

3. В открывшемся диалоговом окне присвойте новому методу название и при желании измените имена переменных. (При желании вы также можете перемещать переменные вверх или вниз по списку.) Затем нажмите на кнопку **OK** и понаблюдайте за волшебством. По умолчанию, новый метод будет расположен сразу за вашим текущим методом.

Если другие фрагменты в коде являются полностью идентичными выделенному фрагменту, то есть выражения в других фрагментах кода идут в таком же порядке, все типы совпадают и т. д., они также будут заменены вызовом данного нового метода! Вы можете увидеть количество дополнительных вхождений в диалоговом окне команды **Extract Method** (Извлечь метод). Если это количество отличается от ваших ожиданий, убедитесь, что все фрагменты кода в точности соответствует одному и тому же шаблону.

Теперь ваш код будет выглядеть примерно так:

```
addTextToRowWithValues(newRow, scoreUserName, textColor, textSize);
```

Работать с этим кодом проще по сравнению с первоначальным кодом, и этот новый код был создан практически без нажатий клавиш! Если до реализации рефакторинга в вашем коде было десять подобных фрагментов, благодаря использованию полезной команды среды разработки Eclipse вы сэкономите уйму времени.

РЕШЕНИЕ ЗАГАДОЧНЫХ ПРОБЛЕМ КОМПИЛЯЦИИ

Порой вы можете столкнуться с ситуацией, когда среда разработки Eclipse обнаружит ошибки компиляции там, где еще несколько мгновений назад их попросту не было. В

подобной ситуации можно опробовать несколько несложных хитростей среды разработки Eclipse.

Во-первых, попробуйте обновить проект: просто щелкните правой кнопкой мыши по названию проекта и выберите команду **Refresh** (Обновить), или нажмите клавишу **F5**. Если это не помогло, попробуйте удалить файл **R.java**, который находится в каталоге **gen** под названием конкретного компилируемого пакета. (Не беспокойтесь: этот файл создается при каждой компиляции проекта.) Если напротив пункта меню **Compile Automatically** (Компилировать автоматически) установлен флажок, этот файл будет пересоздан автоматически. В противном случае, вам понадобится скомпилировать проект вручную.

Наконец, вы можете попробовать очистить проект. Для этого выберите команду меню **Project** => **Clean** (Проект => Очистить) и в открывшемся диалоговом окне **Clean** (Очистка) выберите проекты, которые вы хотите очистить. Среда разработки Eclipse удалит все временные файлы и затем перекомпилирует проект(ы).

СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ФИЛЬТРОВ ДЛЯ ЖУРНАЛИРУЕМОЙ ИНФОРМАЦИИ

Каждая инструкция журналирования в Androici-приложении имеет тег. Эти теги могут быть использованы вместе с фильтрами, определяемыми на панели **LogCat**. Чтобы добавить новый фильтр, щелкните по значку «плюс» зеленого цвета на панели **LogCat**. Укажите название для фильтра — возможно, используя имя тега — и введите тег, который вы хотите использовать. Теперь на панели **LogCat** появится другая вкладка, на которой будут отображаться сообщения, содержащие указанный тег. Помимо этого, вы можете создавать фильтры, которые отображают элементы в зависимости от степени серьезности сообщения.

Соглашение по именованию тегов, применяемое при разработке Android- приложений, подразумевает использование в качестве имени тега названия класса. Вы часто видели это в коде, прилагаемом к данной книге. Обратите внимание, что в каждом классе мы создаем константу с одним и тем же именем, чтобы упростить вызов инструкций журналирования. Вот пример:

```
public static final String DEBUG_TAG = "MyClassName";
```

Тем не менее, данное соглашение не обязательно. Теги могут быть организованы по отдельным задачам, которые выполняются в различных действиях, или вы можете использовать любой другой разумный способ организации тегов, соответствующий вашим требованиям.

ПЕРЕМЕЩЕНИЕ ВКЛАДОК

Среда разработки Eclipse предоставляет ряд довольно неплохих компоновок панелей, относящихся к стандартным перспективам. Тем не менее у каждого из нас есть свои стиль работы, а для разработки Android-приложений доступные перспективы имеют неудовлетворительные стандартные компоновки.

Например, вкладка **Properties** (Свойства) обычно находится в нижней части экрана. Для написания кода такое решение работает неплохо, поскольку эта вкладка занимает всего лишь несколько строк по высоте. Однако при разработке макетов для Android-приложений такой вариант размещения не совсем удобен.

К счастью, в среде разработки Eclipse эту проблему легко исправить: просто перетащите вкладку, щелкнув мышью по самой вкладке (ее заголовку), в новое местоположение, например, в вертикальную группу вкладок, расположенную в правой части окна среды разработки Eclipse. Это позволит получить необходимое вертикальное пространство для отображения десятков свойств, которые часто появляются на этой панели.

Вы можете поэкспериментировать, чтобы найти оптимальную для вас компоновку вкладок. Каждая перспектива также имеет свою собственную компоновку, и перспективы могут быть ориентированы на выполнение определенных задач. Если вы окончательно испортили перспективу или просто хотите начать все сначала, достаточно выбрать команду **Window** => **Reset Perspective** (Окно => Восстановить перспективу).

ИНТЕГРАЦИЯ С СИСТЕМОЙ КОНТРОЛЯ ИСХОДНОГО КОДА

Среда разработки Eclipse имеет возможность интегрироваться со многими системами контроля исходного кода путем использования дополнений. Благодаря этой интеграции среда разработки Eclipse может самостоятельно получать последнюю версию файла, позволяя вносить изменения в файл, когда вы приступите к редактированию файла, выгружать изменения в систему контроля исходного кода, обновлять содержимое файла, отображать статус файла и выполнять множество других задач, которые зависят от реализуемой поддержки функциональности конкретным дополнением. Наиболее распространенными являются дополнения для систем контроля исходного кода CVS, Subversion, Perforce, git и и многих других.

Вообще говоря, не все файлы подходят для помещения в систему исходного кода. Для проектов Android-приложений любые файлы находятся в каталогах **bin** и **gen**, не должны помешаться в систему контроля исходного кода.

Чтобы исключить добавление этих файлов в систему контроля исходного кода из среды разработки Eclipse, откройте диалоговое окне **Preferences** (Настройки), разверните элемент списка **Team** (Команлная работа; и выберите раздел **Ignored Resources** (Игнорируемые ресурсы). Поочередно добавьте в список Ignore Patterns (Игнорируемые шаблоны) маски файлов ***.apк**, ***.ap** и ***.dex**, используя кнопку **Add Pattern** (Добавить шаблон).