Игорь Борисов

PHP. Уровень 4. Расширенные возможности PHP

http://igor-borisov.ru

Темы курса

- Шаблоны проектирования
- Standard PHP Library (SPL)
- PHP Data Objects (PDO)
- Reflection
- cURL
- Регулярные выражения
- Пространства имен
- Модульное тестирование
- Шаблон проектирования MVC
- Создание REST-сервиса

Подготовка рабочего места

Создание рабочего окружения

Задание 1: Создание виртуального хоста и запуск сервера

- Откройте проводник Windows
- Перейдите в директорию **C:\Пользователи\Общие\OpenServer \domains**
  (Внимание! В некоторых ситуациях русскоязычному пути C: \Пользователи\Общие\ соответствует англоязычный путь C:\Users \Public\. Это одно и тоже.)
- В этой директории создайте папку **mysite.local**
- Запустите сервер. Для этого нажмите
  [Пуск -> Open Server]
  (На всякий случай, сама программа находится по пути C:\Пользователи\Общие\OpenServer\Open Server.exe)
- В правом нижнем углу (рядом с часами) кликните по иконке с красным флажком
- В открывшемся меню выберите первый пункт **Запустить**
- Дождитесь пока цвет иконки с флажком изменится с желтого на зеленый
- Если запуск закончился неудачей - флажок опять стал красным, то кликните по иконке, выберите последний пункт **Выход** и повторите последние 4 пункта

Задание 2: Копирование необходимых файлов

- Получите у преподавателя архив с файлами для работы на курсе
- Распакуйте архив в созданную в предыдущем упражнении директорию
  C:\Пользователи\Общие\OpenServer\domains\mysite.local
- Запустите браузер и в адресной строке наберите: **http://mysite.local/**
- Вы должны увидеть главную страницу учебного сайта

Модуль 1

PHP. Уровень 4
Шаблоны проектирования

Темы модуля

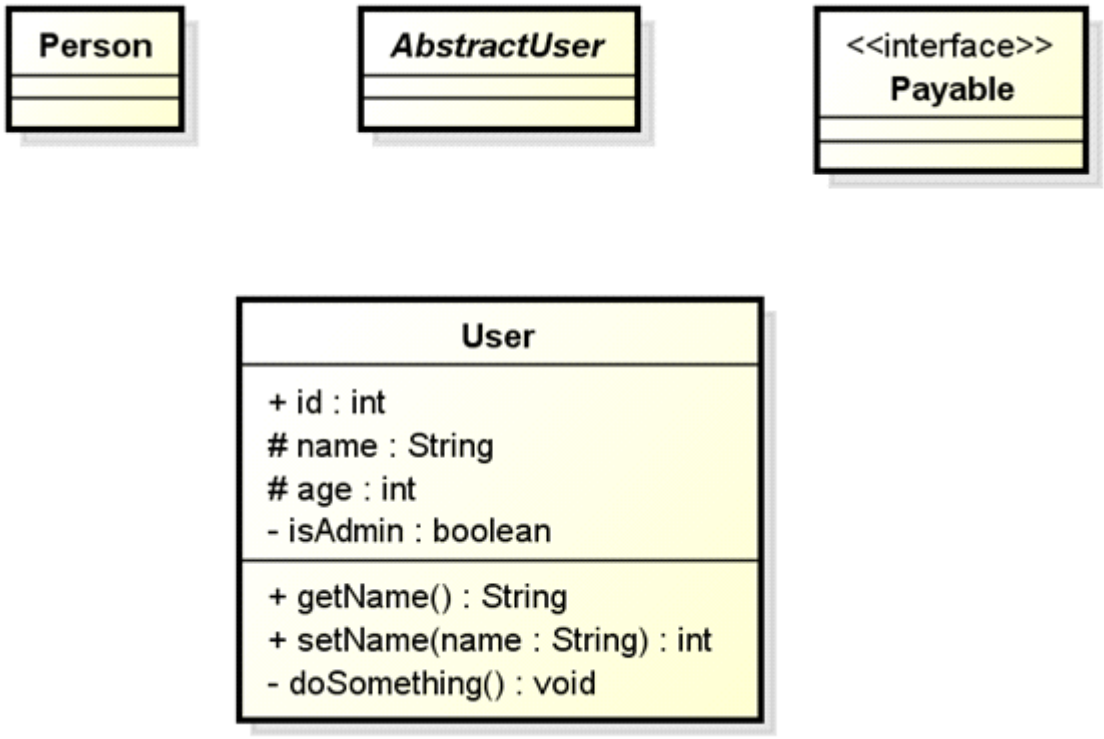
- Обзор UML
- Диаграмма классов
- Введение в шаблоны проектирования
- Шаблоны проектирования
    - Singleton Pattern
    - Factory Pattern
    - Strategy Pattern
    - Decorator Pattern
    - Adapter Pattern
- Другие шаблоны

Обзор UML

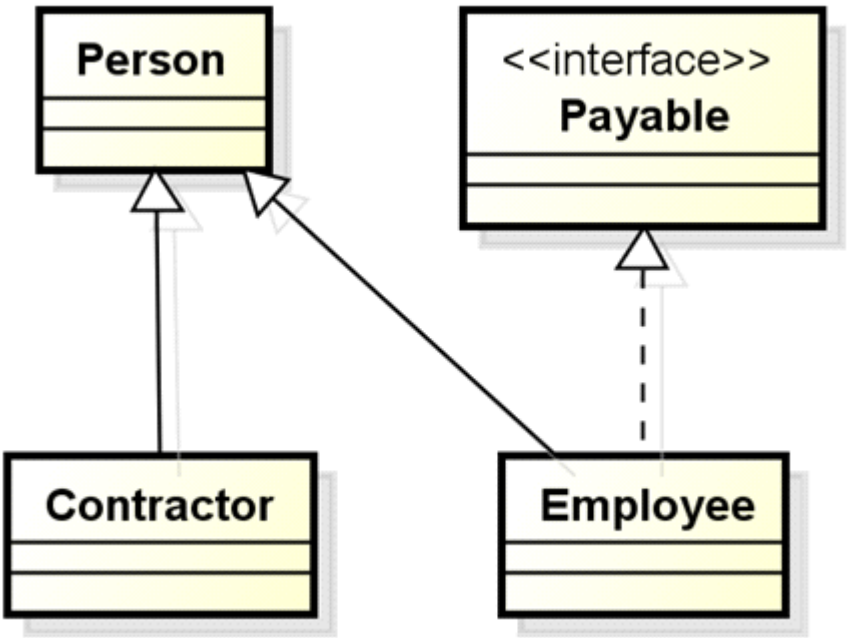
- Unified Modeling Language
- Унифицированный Язык Моделирования
  - Структурные диаграммы
  - Диаграммы поведения
  - Диаграммы взаимодействия
- Диаграммы
  - классов, объектов
  - активности, связей
  - компонентов, составных структур
  - развертывания, машин состояния
  - синхронизации, прецедентов
  - обзора, взаимодействий

Диаграмма классов

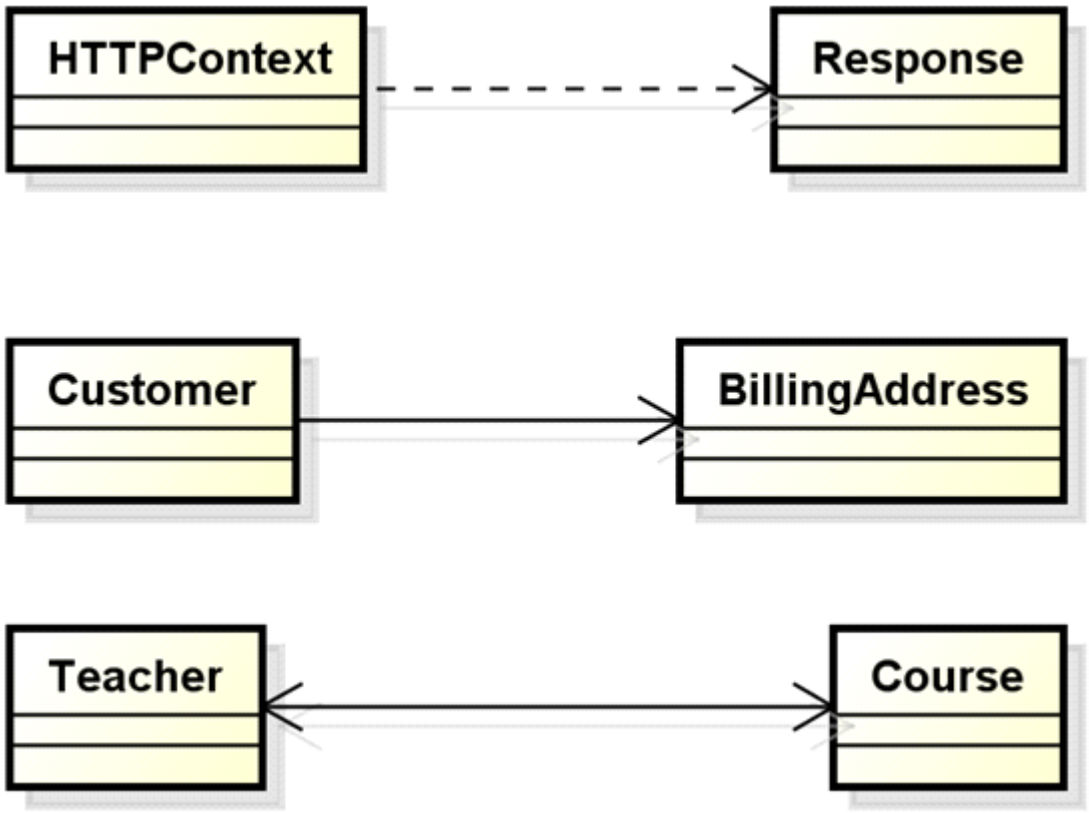
- Описание



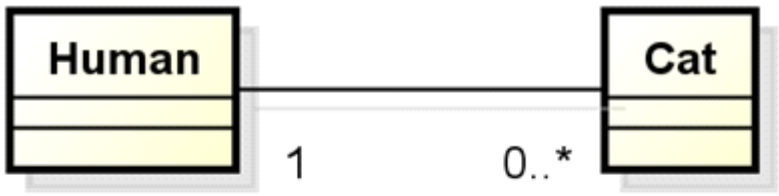
- Обобщение и реализация

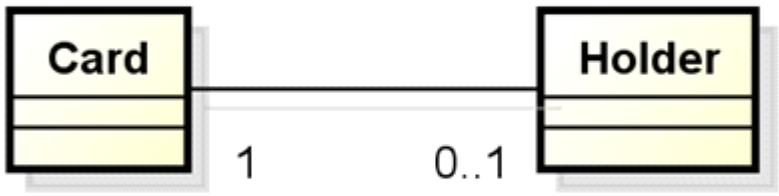


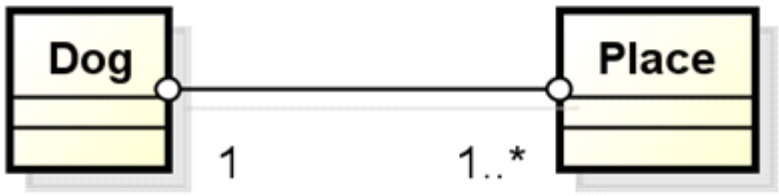
- Зависимости и ассоциации



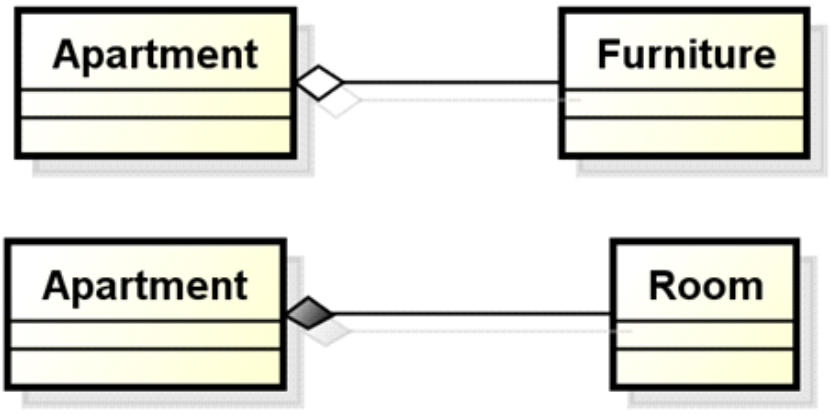
- Мощность

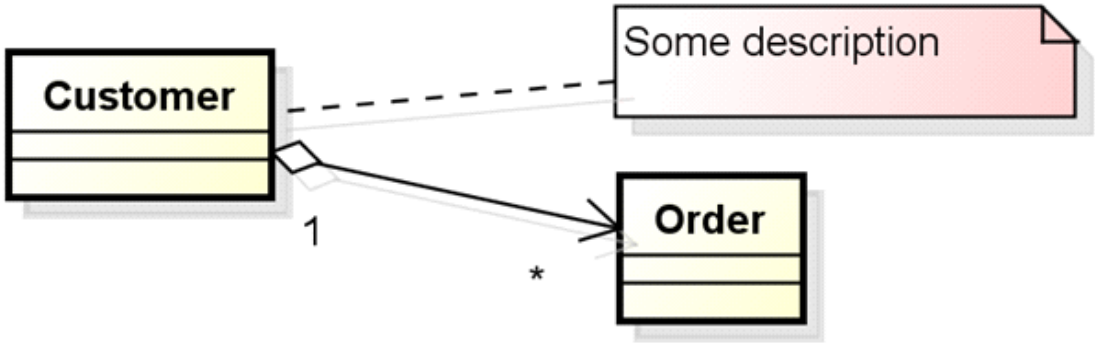






- Агрегация и композиция

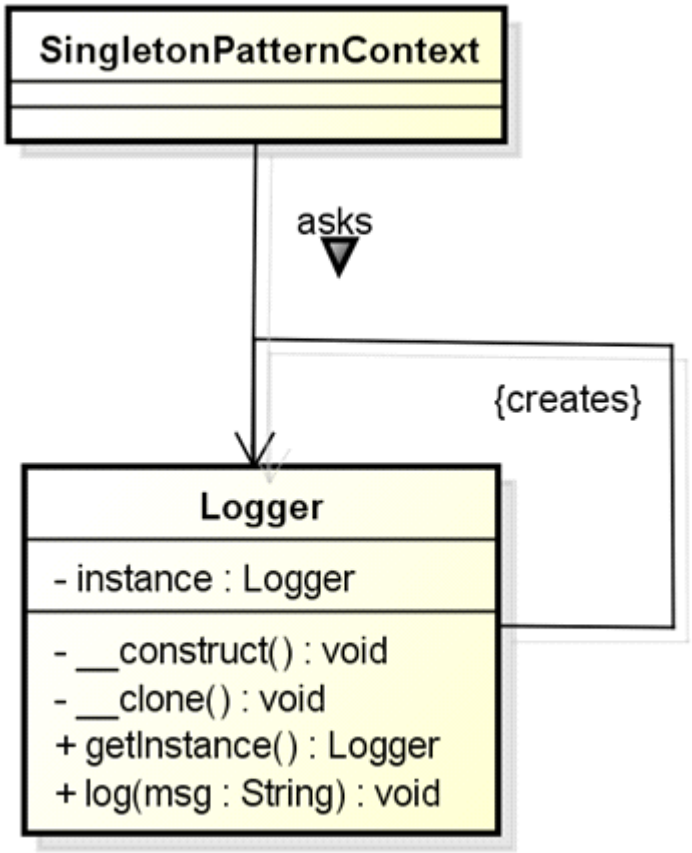




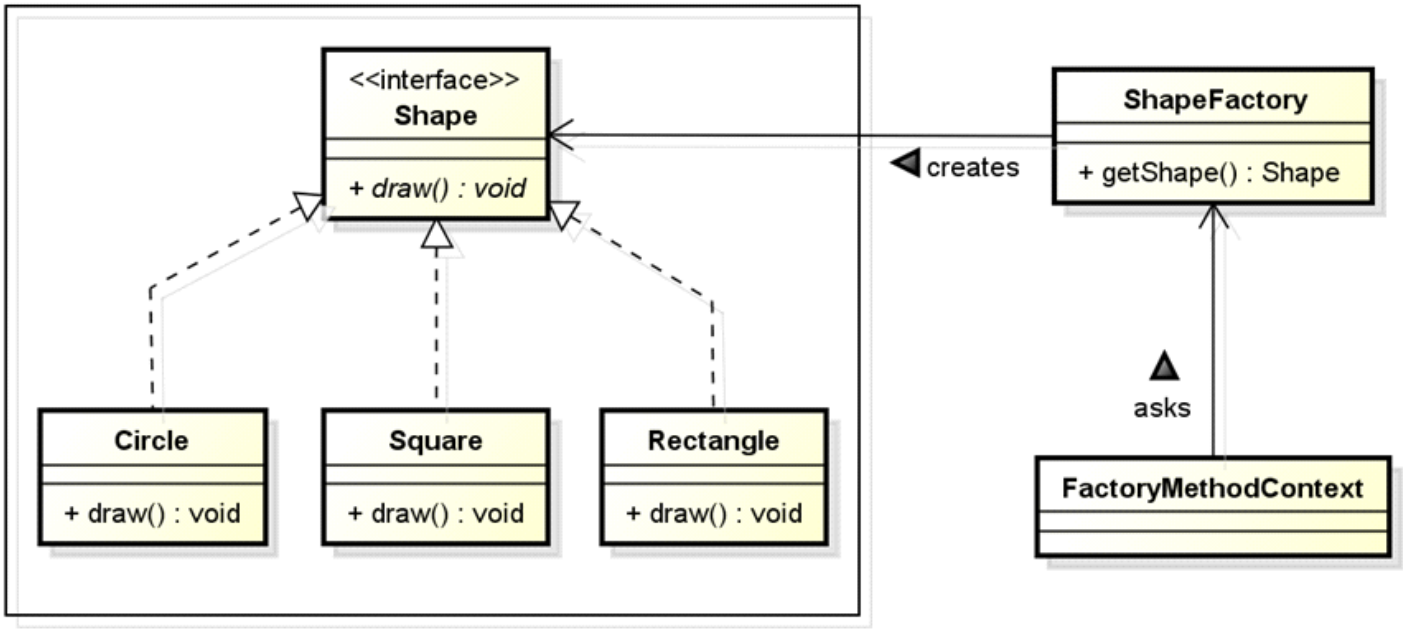
Шаблоны проектирования

- Порождающие шаблоны проектирования
  - Factory Method/Фабричный метод
  - Abstract Factory/Абстрактная фабрика
  - Singleton/Одиночка
- Структурные шаблоны
  - Adapter/Адаптер
  - Decorator/Декоратор
- Поведенческие шаблоны
  - Observer/Наблюдатель
  - Strategy/Стратегия

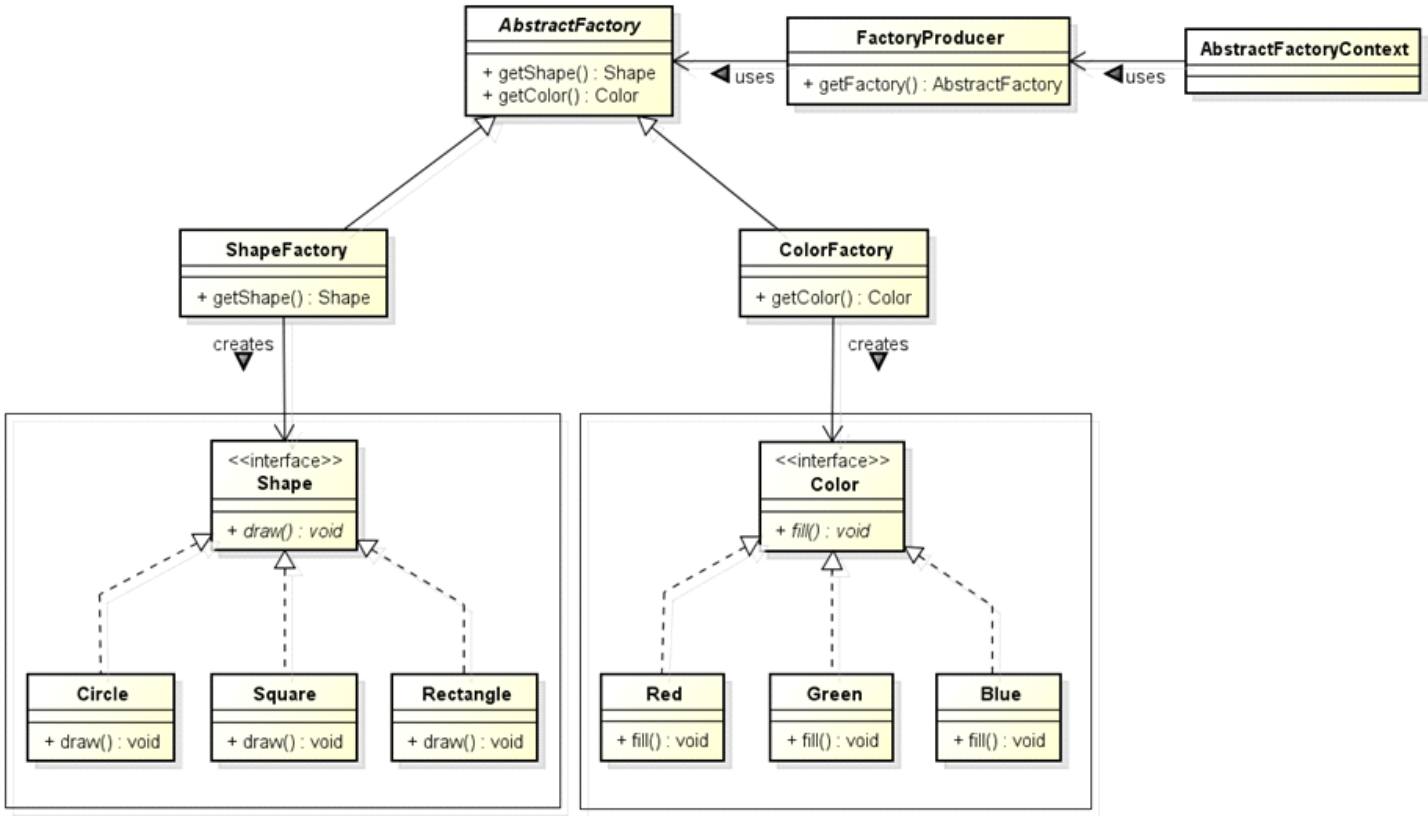
Singleton


SingletonPatternContext
asks
{creates}
Logger
- instance : Logger
- __construct() : void
- __clone() : void
+ getInstance() : Logger
+ log(msg : String) : void

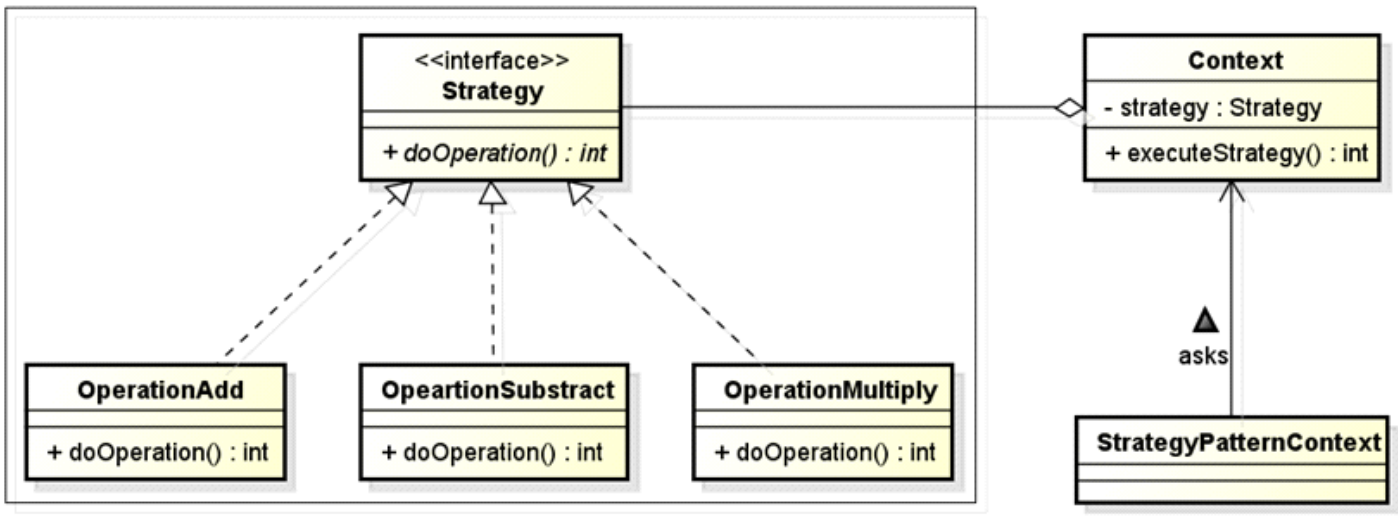
Factory Method



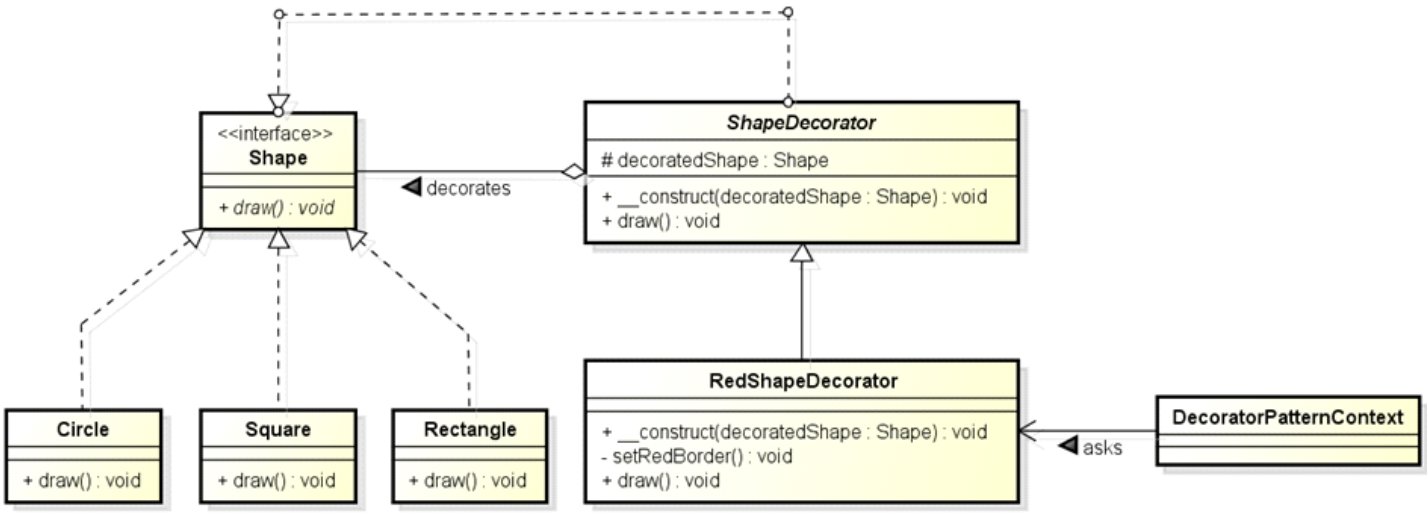
Abstract Factory



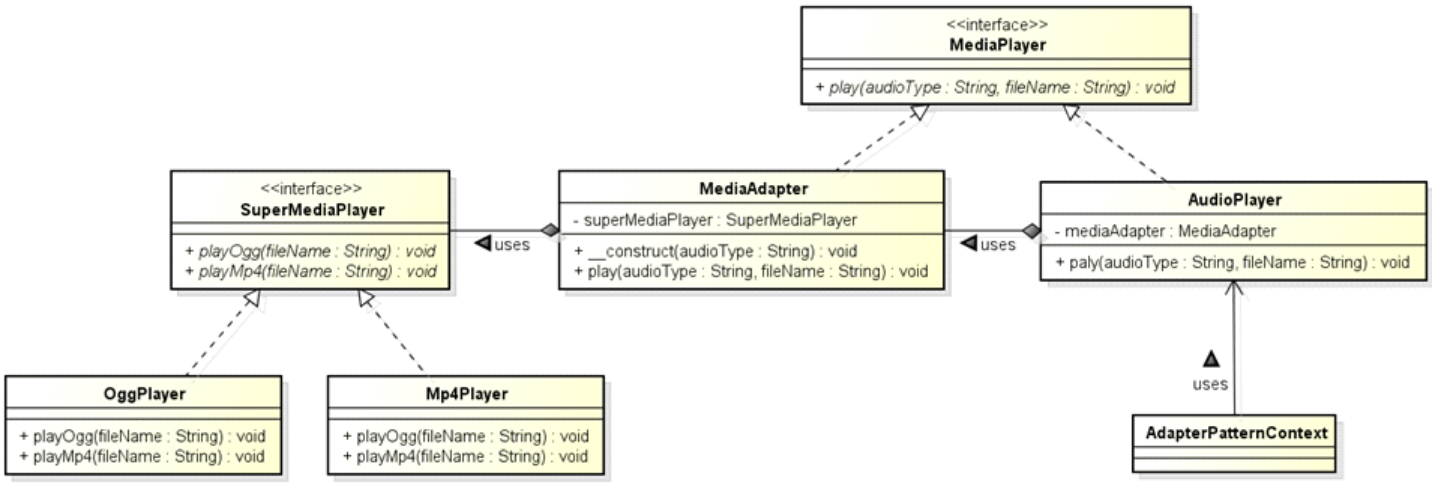
Strategy


<<interface>>
Strategy
+ doOperation() : int
Context
- strategy : Strategy
+ executeStrategy() : int
asks
OperationAdd
+ doOperation() : int
OpeartionSubstract
+ doOperation() : int
OperationMultiply
+ doOperation() : int
StrategyPatternContext

Decorator


<<interface>>
Shape
+ draw() : void
decorates
ShapeDecorator
decoratedShape : Shape
+ __construct(decoratedShape : Shape) : void
+ draw() : void
Circle
+ draw() : void
Square
+ draw() : void
Rectangle
+ draw() : void
RedShapeDecorator
+ __construct(decoratedShape : Shape) : void
- setRedBorder() : void
+ draw() : void
asks
DecoratorPatternContext

Adapter


<<interface>>
MediaPlayer
+ play(audioType : String, fileName : String) : void
<<interface>>
SuperMediaPlayer
+ playOgg(fileName : String) : void
+ playMp4(fileName : String) : void
uses
MediaAdapter
- superMediaPlayer : SuperMediaPlayer
+ __construct(audioType : String) : void
+ play(audioType : String, fileName : String) : void
uses
AudioPlayer
- mediaAdapter : MediaAdapter
+ paly(audioType : String, fileName : String) : void
OggPlayer
+ playOgg(fileName : String) : void
+ playMp4(fileName : String) : void
Mp4Player
+ playOgg(fileName : String) : void
+ playMp4(fileName : String) : void
uses
AdapterPatternContext

Что мы изучили?

- Познакомились с UML
- Рассмотрели наиболее известные шаблоны проектирования и их реализацию на PHP

Модуль 2

PHP. Уровень 4
Standard PHP Library

Темы модуля

- Встроенные классы и интерфейсы
- SPL – Standard PHP Library
- Общие принципы
- Интерфейсы
- Итераторы
- Классы
- Структуры данных
- Функции

Встроенные классы и интерфейсы

- Классы
  - Closure
  - Generator
- Интерфейсы
  - Traversable
  - Iterator
  - IteratorAggregate
  - ArrayAccess
  - Serializable

Класс Closure

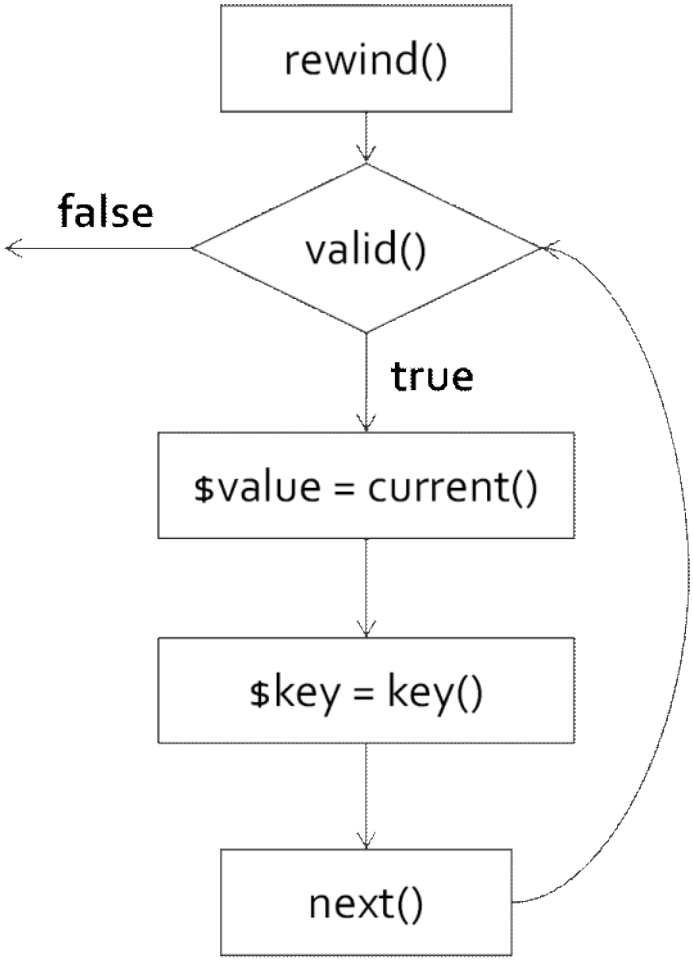
```php
<?
// Обращение к функции через переменную
function Hello($name){
  echo "Привет, $name";
}

$func = "Hello";
$func("Мир!");

// Анонимная функция
$func = function($name){
          echo "Привет, $name\n";
        };
$func("Мир!");



// Использование
$arr = [1, 2, 3, 4, 5];

// Стандартный вариант
function foo($v){
  return $v * 10;
}
$new_arr = array_map(foo, $arr);

// Хак
$new_arr = array_map(create_function('$v', '$v *
10;'), $arr);

// Самый удобный вариант
$new_arr = array_map(function($v){
                        return $v * 10;
                      }, $x);


// Closure (замыкание)
$string = "Hello, world!";
$closure = function() use ($string) {
```

```
            echo $string
        };
$closure();

// Переопределение значения внешней переменной
$x1 = 1;
$closure = function() use (&$x) {
            ++$x;
        };
echo $x; // 1
closure();
echo $x; // 2

// Использование
$mult = function($num){
        return function($x) use ($num){
                return $x * $num;
            };
    };
$mult_by_2 = $mult(2);
$mult_by_3 = $mult(3);
echo $mult_by_2(2); // 4
echo $mult_by_2(5); // 10
echo $mult_by_3(2); // 6
echo $mult_by_3(5); // 15

// Использование в классах
class User{
  private $_name;

  public function __construct($n){ $this->_name = $n;}

  public function greet($greeting){
          return function() use ($greeting) {
      return "$greeting {$this->_name}!";
  };
  }
}

$user = new User("John");
$en = $user->greet("Hello");
echo $en();
```

```php
$ru = $user->greet("Привет");
echo $ru();
```

Интерфейс Iterator

```
Iterator extends Traversable

mixed   current ( void )
scalar  key     ( void )
void    next    ( void )
void    rewind  ( void )
boolean valid   ( void )
```



Лабораторная работа 2.1

Использование интерфейса Iterator

Содержание лабораторной работы 2.1

Использование интерфейса Iterator

Задание

- Создать класс **NumbersSquared** для объекта-итератора, который принимает в конструкторе последовательность чисел и возводит их в степень числа **2**
- Создание объекта должно выглядить как:
```
$obj = new NumbersSquared(3, 7);
```
- Теперь, при итерировании объекта, числа с **3** до **7** должны быть возведены в степень числа **2**, например так:
```
foreach($obj as $num => $square){
  echo "Квадрат числа $num = $square\n";
}
```

Упражнение 1: Создание класса реализующего итератор

- В текстовом редакторе откройте файл **iterator\NumbersSquared.php**
- Создайте класс **NumbersSquared**, который реализует интерфейс **Iterator**
- Добавьте закрытое (private) свойство **start** для хранения первого числа
- Добавьте закрытое (private) свойство **end** для хранения последнего числа
- Добавьте закрытое (private) свойство **current** для хранения текущего числа
- Опишите конструктор класса, который принимает два параметра и присваивает их значения свойствам **start** и **end** соответственно
- Опишите метод **rewind()**, который устанавливает значение свойства **current** равное значениию свойства **start**
- Опишите метод **valid()**, который возвращает значение **false** в случае, если значение свойства **current** больше значения свойства **end** и значение **true** в обратном случае
- Опишите метод **next()**, который увеличивает значение свойства **current** на единицу

- Опишите метод **key()**, который возвращает значение свойства **current**
- Опишите метод **current()**, который возвращает квадрат значения свойства **current**
- Сохраните файл **iterator\NumbersSquared.php**

Упражнение 2: Итерирование объекта

- В файле **iterator\NumbersSquared.php** создайте объект, экземпляр класса **NumbersSquared**, передав ему в конструктор первое и последнее число любой последовательности чисел
- В цикле **foreach** проитерируйте созданный объект по образцу, указанному в задании к данной Лабораторной работе
- Сохраните файл **iterator\NumbersSquared.php**
- Запустите скрипт
- Убедитесь, что скрипт выполняется корректно и без ошибок
- Если есть ошибки, найдите и исправьте их

Интерфейс IteratorAggregate

```
IteratorAggregate extends Traversable
public Traversable getIterator ( void )
```

```php
<?php
class MyIterator implements Iterator{
  private $var = [];
  private $count = 0;

  public function __construct(array $array){
    $this->var = $array;
    $this->count = count($array);
  }
  public function rewind() {
    reset($this->var);
  }
  public function current() {
    $this->count--;
    return date("d-m-Y", current($this->var));
  }
  public function key() {
    return key($this->var);
  }
  public function next() {
    return next($this->var);
  }
  public function valid() {
    return $this->count !== 0;
  }
}

class MySchedule implements IteratorAggregate{
  private $items = [];

  public function getIterator() {
    asort($this->items);
    return new MyIterator($this->items);
  }

  public function add($key, $value) {
```

```
    $this->items[$key] = $value;
  }
}

$schedule = new MySchedule();
$schedule->add('PHP', mktime(0, 0, 0, 3, 20, 2015));
$schedule->add('HTML', mktime(0, 0, 0, 2, 18, 2015));
$schedule->add('XML', mktime(0, 0, 0, 2, 12, 2015));
$schedule->add('AJAX', mktime(0, 0, 0, 4, 26, 2015));

foreach ($schedule as $key => $val) {
  echo "$key : $val\n";
}
```

Интерфейс ArrayAccess

```
ArrayAccess
boolean offsetExists (  mixed $offest  )
mixed   offsetGet    (  mixed $offest  )
void    offsetSet    (  mixed $offest, mixed $value)
void    offsetUnset  (  mixed $offest  )
```

```php
<?php
class Registry implements ArrayAccess{
  private $props = [];

  public function offsetSet($name, $value){
    $this->props[$name] = $value;
    return true;
  }

  public function offsetExists($name){
    return isset($this->props[$name]);
  }

  public function offsetUnset($name){
    unset($this->props[$name]);
    return true;
  }

  public function offsetGet($name){
    if(!isset($this->props[$name])){
      return null;
    }
      return $this->props[$offset];
  }
}
$obj = new Registry();
$obj["login"] = 'root';
$obj["password"] = '1234';
if(isset($obj["login"]))
  echo $obj["login"] . ':' . $obj["password"];
unset($obj["password"]);
```

Перегрузка сериализации

```
Serializable
string serialize   ( void )
void   unserialize ( string $serialized )
```

```php
<?php
class MyData implements Serializable{
  public $data;
  public function __construct(){
    $this->data = "Some data";
  }
  public function getData(){
    return $this->data;
  }
  public function serialize(){
    return serialize($this->data);
  }
  public function unserialize($data){
    $this->data = unserialize($data);
  }
}

$before = new MyData();
$serializedString = serialize($before);
unset($before);
$after = unserialize($serializedString);
echo $after->getData();
```

Класс Generator

```php
function numbers() {
  echo "START\n";
  for ($i = 0; $i < 5; ++$i) {
    yield $i;
  }
  echo "FINISH\n";
}

foreach (numbers() as $value){
  echo "VALUE: $value\n";
}


// Возврат ключей?
function gen() {
  yield 'a';
  yield 'b';
  yield 'name' => 'John';
  yield 'd';
  yield 10 => 'e';
  yield 'f';
}
foreach (gen() as $key => $value)
  echo "$key : $value\n";


// Co-routines: принимаем значение!
function echoLogger() {
  while (true) {
    echo 'Log: ' . yield . "<br>";
  }
}
$logger = echoLogger();
$logger->send('Foo');
$logger->send('Bar');


// Комбинируем возврат и приём значений
function numbers() {
```

```php
  $i = 0;
  while (true) {
    $cmd = (yield $i);
    ++$i;
    if ($cmd == 'stop')
      return; // Выход из цикла
  }
}

$gen = numbers();
foreach ($gen as $v) {
  if ($v == 3)
    $gen->send('stop');
  echo $v;
}
```

SPL

- Standard PHP Library
  - Стандартная библиотека PHP — коллекция классов и интерфейсов для решения стандартных проблем в PHP
  - Библиотека была введена в PHP 5 и доступна по умолчанию, начиная с PHP 5.3

- Итераторы
  - ArrayIterator
  - RecursiveArrayIterator
  - RecursiveIteratorIterator
  - FilterIterator
  - LimitIterator
  - AppendIterator
  - InfiniteIterator
  - DirectoryIterator
  - RecursiveDirectoryIterator
  - RecursiveTreeIterator
  - GlobIterator
  - …
- Интерфейсы
  - Countable
  - RecursiveIterator
  - OuterIterator
  - SeekableIterator
- Обработка файлов
  - SplFileInfo
  - SplFileObject

- ...
- Различные классы и интерфейсы
  - ArrayObject
  - ...
- Структуры данных
  - SplStack
  - SplQueue
  - SplHeap
  - SplMaxHeap
  - SplMinHeap
  - SplPriorityQueue
  - SplFixedArray
  - SplObjectStorage
- Функции
  - spl_classes
  - spl_object_hash
  - spl_autoload
  - iterator_to_array
  - ...

Создание итератора из массива

```
ArrayIterator implements Iterator,
                         Traversable,
                         ArrayAccess,
                         SeekableIterator,
                         Countable,
                         Serializable
```

```php
<?php
$it = new ArrayIterator( [3, 2, 1) ];

foreach ($it as $value)
  echo $value . " "; // 3 2 1

$it -> rewind();
$it -> asort();

foreach ($it as $value)
  echo $value . " "; // 1 2 3

$it -> count(); // 3

// Получаем массив из итератора
$array = iterator_to_array($it);
```

Лабораторная работа 2.2

Использовние класса ArrayIterator и интерфейса IteratorAggregate

Содержание лабораторной работы 2.2

Использование класса ArrayIterator и интерфейса IteratorAggregate

Упражнение 1: Изменение основного класса проекта

- В текстовом редакторе откройте файл **news\NewsDB.class.php**
- Добавьте в качестве "родителя" класса **NewsDB** интерфейс **IteratorAggregate**
- Добавьте закрытое (private) свойство **items** и присвойте ему значение по умолчанию пустой массив
- Создайте закрытый (private) метод **getCategories()**
- Опишите метод **getCategories()**:
    - получите значение полей **id** и **name** из таблицы **category** базы данных **news.db**
    - заполнить свойство класса **items** как:
        - **ключи** элементов массива соответствуют значениям поля **id**
        - **значения** элементов массива соответствуют значениям поля **name**
- В конструкторе класса вызовите метод **getCategories()**
- Создайте и опишите метод **getIterator()**
- Метод **getIterator()** должен возвращать экземпляр класса **ArrayIterator**, в конструктор которого необходимо передать в качестве аргумента свойство класса **items**
- Сохраните файл **news\NewsDB.class.php**

Упражнение 2: Использование класса в качестве итератора

- В текстовом редакторе откройте файл **news\news.php**
- В HTML-коде удалите все между тэгами **<select></select>**
- В цикле **foreach** отрисуйте HTML-список используя объект **$news** в качестве итератора
- Сохраните файл **news\news.php**

- Запустите браузер и в адресной строке наберите **http://mysite.local/news/news.php**
- Вы не должны увидеть каких-либо изменений по сравнению с предыдущими запусками
- Убедитесь, что список категорий отобразился корректно

Рекурсивная итерация

```
RecursiveArrayIterator extends ArrayIterator
                       implements RecursiveIterator

RecursiveIteratorIterator implements OuterIterator,
                                     Traversable,
                                     Iterator
```

```php
<?php
$arr = [1, [2, [3]], [4]];

$rit = new RecursiveArrayIterator($arr);

$rii = new RecursiveIteratorIterator($rit);

foreach ($rii as  $key => $value) {
  $depth = $rit -> getDepth();
  echo "depth=$depth key=$key value=$value\n";
}
// depth=0 key=0 value=1
// depth=1 key=0 value=2
// depth=2 key=0 value=3
// depth=1 key=0 value=4
```

Фильтрация элементов

```
abstract FilterIterator extends IteratorIterator
                        implements OuterIterator,
                                   Traversable,
                                   Iterator
```

```php
<?php
class MyClass extends FilterIterator{
  public function accept() {
    return $this->getInnerIterator()->current() > 5;
  }
}
$arr = [5, 2, 7, 1, 9, 3, 6, 8];
$it = new ArrayIterator($arr);
$fit = new MyClass($it);
foreach ($fit as $value)
  echo  $value . " ";
// 7 9 6 8
```

Ограничение итераций

```
LimitIterator extends IteratorIterator
              implements OuterIterator,
                         Traversable,
                         Iterator
```

```php
<?php
$arr = [1, 2, 3, 4, 5, 6, 7, 8];

$it = new LimitIterator($arr, 2, 4);

foreach ($it as $value)
  echo  "$value ";
// 3 4 5 6
```

Бесконечная итерация с объединением итераторов

```
AppendIterator extends IteratorIterator
               implements OuterIterator,
                          Traversable,
                          Iterator
```

```
<?php
class MyObject {
  public function action() {
    // Что-то делаем
    return $boolean;
  }
}

$object1 = new MyObject();
$object2 = new MyObject();
$arrayIterator1 = new ArrayIterator( [$object1,
$object2] );

$object3 = new MyObject();
$object4 = new MyObject();
$arrayIterator2 = new ArrayIterator( [$object3,
$object4] );

// Объединение итераторов
$arrayIterator = new AppendIterator();
$arrayIterator->append($arrayIterator1);
$arrayIterator->append($arrayIterator2);
?>
```

```
InfiniteIterator extends IteratorIterator
                 implements OuterIterator,
                            Traversable,
                            Iterator
```

```
<?php
// Бесконечная итерация
$it = new InfiniteIterator($arrayIterator);
foreach ($it as $object){
  $r = $object -> action();
```

```
  if(!$r) break;
}
```

Работа с файлами

```
SplFileInfo
SplFileObject extends SplFileInfo
               implements RecursiveIterator,
                          Traversable,
                          Iterator,
                          SeekableIterator
```

```
<?php
$fileInfo = new SPLFileInfo('data.txt');
$fileProps = [];

$fileProps['filename'] = $fileInfo->getFilename();
$fileProps['pathname'] = $fileInfo->getPathname();
$fileProps['size'] = $fileInfo->getSize();
$fileProps['mtime'] = $fileInfo->getMTime();
$fileProps['type'] = $fileInfo->getType();
$fileProps['isWritable'] = $fileInfo->isWritable();
$fileProps['isReadable'] = $fileInfo->isReadable();
$fileProps['isExecutable'] = $fileInfo->
isExecutable();
$fileProps['isFile'] = $fileInfo->isFile();
$fileProps['isDir'] = $fileInfo->isDir();

var_export($fileProps);

echo "============\n";

$file = new SplFileObject('data.txt');
foreach($file as $line) {
  echo "$line\n";
}

echo "============\n";

$file->rewind();
while($file->valid()){
  echo $file->current()."\n";
  $file->next();
}
```

```php
echo "============\n";

$file->seek(3);
echo $file->current();
```

Работа с директориями

```
DirectoryIterator extends SplFileInfo
                  implements SeekableIterator,
                             Traversable,
                             Iterator
```

```
<?php
// Итерация директорий
foreach(new DirectoryIterator('.') as $fileInfo) {
echo $fileInfo->getFileName() . "\n";
}
?>
```

```
FilesystemIterator extends DirectoryIterator
                  implements SeekableIterator
RecursiveDirectoryIterator extends FilesystemIterator
                           implements SeekableIterator,
                                      RecursiveIterator
```

```
<?php
// Рекурсивная итерация директорий
function callback($objectName){
  if($objectName->isDir())
    echo "[$objectName]\n";
  else
    echo "$objectName\n";
}
$rdi = new RecursiveDirectoryIterator('.');
$rii = new RecursiveIteratorIterator($rid);
array_map('callback', iterator_to_array($rii));
?>
```

```
RecursiveTreeIterator extends
RecursiveIteratorIterator
                      implements OuterIterator
```

```
<?php
//Строим дерево
$rdi = new RecursiveDirectoryIterator('.');
$tree = new RecursiveTreeIterator($rdi);
$tree->
```

```
setPrefixPart(RecursiveTreeIterator::PREFIX_LEFT,
"//");
$tree->
setPrefixPart(RecursiveTreeIterator::PREFIX_MID_HAS_NE
XT, ":");
$tree->
setPrefixPart(RecursiveTreeIterator::PREFIX_END_HAS_NE
XT, "[]");
$tree->
setPrefixPart(RecursiveTreeIterator::PREFIX_RIGHT,
"||");

foreach ($tree as $item) {
  echo $item . "\n";
}
?>
```

```
GlobIterator extends FilesystemIterator
            implements SeekableIterator,
                       Countable
```

```
<?php
// Отбираем нужное
$gi = new GlobIterator(__DIR__ .'/*.php');
while($gi->valid()){
  echo $gi->key() . "\n";
  $gi->next();
}
```

Массив как объект

```
ArrayObject  implements IteratorAggregate,
                        ArrayAccess,
                        Traversable,
                        Iterator,
                        Countable
```

```php
<?php
$usersArr = [
    'Вася', 'Петя', 'Иван', 'Маша', 'Джон',
    'Майк', 'Даша', 'Наташа', 'Света'
];

$usersObj = new ArrayObject($usersArr);

// Добавляем новое значение
$usersObj->append('Ира');

//Получаем копию массива
$usersArrCopy = $usersObj->getArrayCopy();

// Проверяем, существует ли пятый элемент массива
if ($usersObj->offsetExists(4)){
  // Меняем значение пятого элемента массива
  $usersObj->offsetSet(4, "Игорь");
}

// Удаляем шестой элемент массива
$usersObj->offsetUnset(5);

// Получаем количество элементов массива
echo $usersObj->count();

// Сортируем по алфавиту
$usersObj->natcasesort();

// Выводим данные массива
for($it = $usersObj->getIterator(); $it->valid();
$it->next()){
  echo $it->key() . ': ' . $it->current() . "\n";
```

```
}

// Копия массива
$usersObjCopy = new ArrayObject($usersArrCopy);
// Выводим данные из копии массива
for($it = $usersObjCopy->getIterator(); $it->valid();
$it->next()){
  echo $it->key() . ': ' . $it->current() . "\n";
}
```

Структуры данных: Хранилище

```
SplObjectStorage  implements ArrayAccess,
                             Serializable,
                             Traversable,
                             Iterator,
                             Countable
```

```php
<?php
$storage = new SplObjectStorage();

$object1 = (object) ['param' => 'name'];
$object2 = (object) ['param' => 'numbers'];

$storage[$object1] = "John";
$storage[$object2] = [1, 2, 3];

foreach($storage as $i => $key){
  echo "Item $i:\n";
  var_dump($key, $storage[$key]);
  echo "\n";
}
```

Структуры данных: Стек

```
SplDoublyLinkedList implements Iterator,
                               Countable,
                               ArrayAccess
SplStack extends SplDoublyLinkedList
```

```php
<?php
$stack = new SplStack();

$stack -> push("John");
$stack -> push("Mike");
$stack -> pop(); // Mike
$stack -> pop(); // John

$stack -> push("John");
$stack -> push("Mike");
$stack -> top(); // Mike
$stack -> pop(); // Mike
$stack -> bottom(); // John
$stack -> pop(); // John
```

Структуры данных: Очередь

```
SplDoublyLinkedList implements Iterator,
                               Countable,
                               ArrayAccess
SplQueue extends SplDoublyLinkedList
```

```php
<?php
class Work {
  public function __construct($title) {
    $this->title = $title;
  }
  public function doIt(){
    return $this->title;
  }
}

$work1 = new Work("Сходить в магазин");
$work2 = new Work("Прочитать книгу");
$work3 = new Work("Тупить в телевизор");

$queue = new SplQueue();

$queue -> enqueue($work1);
$queue -> enqueue($work2);
$queue -> enqueue($work3);

while ($queue -> count() > 0){
  echo $queue -> dequeue() -> doIt();
}
```

Структуры данных: Очередь с приоритетом

```
SplPriorityQueue  implements Iterator, Countable
```

```
<?php
class Work {
  public function __construct($title) {
    $this->title = $title;
  }
  public function doIt(){
    return $this->title;
  }
}

$work1 = new Work("Сходить в магазин");
$work2 = new Work("Прочитать книгу");
$work3 = new Work("Тупить в телевизор");

$queue = new SplPriorityQueue();

$queue -> insert($work1, 1);
$queue -> insert($work2, 3);
$queue -> insert($work3, 2);

foreach ($queue as $work){
  echo $work -> doIt();
}
```

Структуры данных: Куча

```
abstract SplHeap implements Iterator,
                            Countable
SplMinHeap extends SplHeap
          implements Iterator,
                     Countable
SplMaxHeap extends SplHeap
          implements Iterator,
                     Countable
```

```
<?php
$minHeap = new SplMinHeap();

$minHeap -> insert(2);
$minHeap -> insert(3);
$minHeap -> insert(1);

foreach ($minHeap as $value)
  echo $value . " "; // 1 2 3


$maxHeap = new SplMaxHeap();

$maxHeap -> insert(2);
$maxHeap -> insert(3);
$maxHeap -> insert(1);

foreach ($maxHeap as $value)
  echo $value . " "; // 3 2 1
```

Массив фиксированной длины

```
SplFixedArray implements Iterator,
                         ArrayAccess,
                         Countable
```

```php
<?php

// Создаём псевдо-массив
$splArray = new SplFixedArray(5);

$splArray[1] = 2;
$splArray[4] = "foo";
$splArray[5] = "bar"; // Ошибка!

echo $array->getSize(); // 5

// Увеличиваем псевдо-массив
$array->setSize(10);

$array[8] = "bar";
```

Автозагрузка классов

```php
<?php
function loadClass ($class_name) {
  require_once "classes/$class_name.class.php";
}
function loadInterface ($class_name) {
  require_once "classes/$class_name.interface.php";
}
function loadSomething ($class_name) {
  // ...
}
class Main{
  public static function autoload(){
    require_once "classes/$class_name.class.php";
  }
}

// Регистрация функций
spl_autoload_register('loadClass');
spl_autoload_register('loadSomething');
spl_autoload_register('loadInterface', true, 1);

// Список зарегистрированных функций
var_dump(spl_autoload_functions());

// Удаление функции из списка зарегистрированных
spl_autoload_unregister('loadSomething');

// Регистрация статического метода класса
spl_autoload_register(['Main', 'autoload']);
```

Что мы изучили?

- Познакомились со встроенными классами и интерфейсами
- Рассмотрели наиболее интересные итераторы, структуры данных, интерфейсы, классы и функции из набора SPL, которые позволяют решать стандартные задачи

Модуль 3

PHP. Уровень 4
PHP Data Objects

Темы модуля

- Введение в PDO
- Поддерживаемые базы данных
- Соединения
- Запросы
- Фильтрация значений
- Обработка ошибок
- Подготовленные запросы
- Использование хранимых процедур
- Транзакции

PDO: что внутри?

- Драйверы
- Предопределённые константы
- Классы
  - PDO
  - PDOStatement
  - PDOException

- Поддерживаемые базы данных
  - PDO_CUBRID
    - Cubrid
  - PDO_DBLIB
    - FreeTDS / Microsoft SQL Server / Sybase
  - PDO_FIREBIRD
    - Firebird/Interbase 6
  - PDO_IBM
    - IBM DB2
  - PDO_INFORMIX
    - IBM Informix Dynamic Server
  - PDO_MYSQL
    - MySQL 3.x/4.x/5.x
  - PDO_OCI
    - Oracle Call Interface
  - PDO_ODBC
    - ODBC v3 (IBM DB2, unixODBC and win32 ODBC)
  - PDO_PGSQL
    - PostgreSQL
  - PDO_SQLITE

        - SQLite 3 and SQLite 2
    - PDO_4D
        - 4D

- Необходимые расширения
    - php_pdo_firebird
    - php_pdo_informix
    - php_pdo_mssql
    - php_pdo_mysql
    - php_pdo_oci
    - php_pdo_oci8
    - php_pdo_odbc
    - php_pdo_pgsql
    - php_pdo_sqlite

    - php_pdo
        - до PHP 5.3

Соединение с базой данных

```php
/* БЫЛО */

// MySQL
mysql_connect('host', 'user', 'password');
mysql_select_db('db');
// MySQLi
$db = mysqli_connect('host', 'user', 'password', 'db');
// MySQLi ООП
$db = new mysqli('host', 'user', 'password', 'db');

// PostgreSQL
$db = pg_connect("host=host dbname=db user=user password=password");

// SQLite2
$db = sqlite_open("db");
// SQLite2 ООП
$db = new SQLiteDatabase("db");
// SQLite3
$db = new SQLite3("db");

/* СТАЛО */

// MySQL
$pdo = new PDO('mysql:host=host;dbname=db', $user, $pass);
// PostgreSQL
$pdo = new PDO('pgsql:host=host;dbname=db', $user, $pass);
// SQLite
$pdo = new PDO('sqlite:db');


// Постоянное соединение
$pdo = new PDO('mysql:host=host;dbname=test', $user, $pass, [PDO::ATTR_PE
RSISTENT => true]);
```

- Строки для соединения с базами данных
  - MySQL
    - ("mysql:host=hostname;dbname=mysql", "username", "password")
  - SQLite
    - ("sqlite:/path/to/database.db") или ("sqlite::memory:")
  - PostgreSQL
    - ("pgsql:dbname=pdo;host=hostname", "username",

"password")
    - Oracle
        - ("OCI:dbname=mydatabase;charset=UTF-8", "username", "password")
    - ODBC
        - ("odbc:Driver={Microsoft Access Driver (*.mdb)};Dbq=C:\database.mdb;Uid=Admin")
    - Firebird
        - ("firebird:dbname=hostname:C:\path\to\database.fdb", "username", "password")
    - Informix
        - ("informix:DSN=InformixDB", "username", "password")
    - DBLIB
        - ("dblib:host=hostname:port;dbname=mydb","username","password")
-

Выполнение запроса без выборки

```php
$sql = "INSERT INTO users(name, email)
              VALUES('john', 'john@smith.com')";

/* БЫЛО */

// MySQL
$result = mysql_query($sql);
// MySQLi
$result = mysqli_query($cnn, $sql);
// MySQLi
$result = $db->query($sql);

// PostgreSQL
$result = pg_query($sql);

// SQLite2
$result = sqlite_exec($sql, $db);
// SQLite2 ООП
$result = $db->queryExec($sql);
// SQLite3
$result = $db->exec($sql);


/* СТАЛО */

$result =  $pdo->exec($sql);

// Проверка ошибок
if($result === false)
  echo "Ошибка в запросе";
```

Экранирование строки

```php
$name = $_POST["user"];

// MySQL
$name = mysql_real_escape_string($name);
// MySQLi
$name = mysqli_real_escape_string($cnn, $name);
// MySQLi ООП
$name = $db->real_escape_string($name);

// PostgreSQL
$name = pg_escape_string($name);

// SQLite2
$name = sqlite_escape_string($name);
// SQLite3
$name = $db->escapeString($name);

$sql = "INSERT INTO users(name) VALUES('$name')";


/* СТАЛО */

$name =  $pdo->quote($name);
$sql = "INSERT INTO users(name) VALUES($name)";
```

Выборка данных

```php
$sql = "SELECT id, name FROM users";

/* БЫЛО */

// MySQL
$result = mysql_query($sql);
// MySQLi
$result = mysqli_query($cnn, $sql);
// MySQLi ООП
$result = $db->query($sql);

// PostgreSQL
$result = pg_query($sql);

// SQLite2
$result = sqlite_query($sql, $db);
// SQLite2 ООП
$result = $db->query($sql);
// SQLite3
$result = $db->query($sql);

/* СТАЛО */
$stmt =  $pdo->query($sql);

// Обработка ошибок
$pdo->query($sql) or die('Ошибка в запросе!');

// Обработка результата
$row = $stmt->fetch(); // PDO::FETCH_BOTH
$row = $stmt->fetch(PDO::FETCH_NUM);
$row = $stmt->fetch(PDO::FETCH_ASSOC);
```

Обработка ошибок

```php
$pdo = new PDO("sqlite: test.db");

// Используем исключения
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

try{
  // Что-то произошло
}catch(PDOException $e){
  $e -> getCode() . ":" . $e -> getMessage();
}

// Используем предупреждения
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);

// Не выводить никаких сообщений
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT);

echo $pdo -> errorCode();
print_r( $pdo -> errorInfo() );
```

Лабораторная работа 3.1

Базовое использование PDO

Содержание лабораторной работы 3.1

Базовое использование PDO

Упражнение 1: Базовое использование методов PDO

- В текстовом редакторе откройте файл **news\NewsDB.class.php**
- Внесите изменения во все методы класса (включая конструктор и деструктор), используя класс **PDO** и его методы: **exec**, **query**, **fetch**, **quote**
- Попробуйте отследить ошибки используя класс **PDOException** и методы **errorCode**, и **errorInfo** класса **PDO**
- Обратите внимание на SQL-запрос в методе **saveNews**. Не забудьте про кавычки!
- Сохраните файл **news\NewsDB.class.php**
- Запустите браузер
- В адресной строке наберите **http://mysite.local/news/news.php**
- Вы не должны увидеть каких-либо изменений по сравнению с предыдущими запусками
- Если есть ошибки, найдите и исправьте их

Результат в виде объекта

```php
$sql = "SELECT id, name, email FROM users";
$pdo = new PDO("sqlite: users.db");

// Приведение результата к объекту
$stmt = $pdo->query($sql);
$obj = $stmt->fetch(PDO::FETCH_OBJ);

echo $obj -> id . "\n";
echo $obj -> name . "\n";
echo $obj -> email . "\n";


// Ленивое приведение
$stmt = $pdo->query($sql);
$obj = $stmt->fetch(PDO::FETCH_LAZY);

echo $obj[0] . "\n";
echo $obj['name'] . "\n";
echo $obj -> email . "\n";
```

Использование класса

```
/*
Таблица users с полями: id, name, email
  1, First, first@email.ru
  2, Second, second@email.ru
*/

// Поиск названия класса в значении первого поля в выборке

$sql = "SELECT id, name, email FROM users";
$stmt = $pdo->query($sql);

$obj = $stmt->fetch( PDO::FETCH_CLASS|PDO::FETCH_CLASSTYPE );
/*
  Результат:
  stdClass Object ( [name] => First [email] => first@email.ru )
  stdClass Object ( [name] => First [email] => second@email.ru )
*/


class First {
  public $id, $name, $email;
}
$sql = "SELECT name, email FROM users";
$stmt = $pdo->query($sql);

$obj = $stmt->fetch( PDO::FETCH_CLASS|PDO::FETCH_CLASSTYPE );
/*
  Результат:
  First Object ( [id] =>NULL [name] => NULL [email] =>
first@email.ru )
  stdClass Object ( [email] => second@email.ru )
*/

class Second {
  public $id, $name, $email;
}
$sql = "SELECT name, email, id FROM users";
$stmt = $pdo->query($sql);

$obj = $stmt->fetch( PDO::FETCH_CLASS|PDO::FETCH_CLASSTYPE );
/*
  Результат:
  First Object ( [id] =>1 [name] => NULL [email] => first@email.ru )
  Second Object ( [id] =>2 [name] => NULL [email] => second@email.ru )
```

```
*/

// Явное указание названия класса для создания объекта
// По-умолчанию stdClass
$sql = "SELECT id, name, email FROM users";
$stmt = $pdo->query($sql);

$obj = $stmt->fetchObject();
/*
  Результат:
  stdClass Object ( [id] =>1 [name] => First [email] =>
first@email.ru )
  stdClass Object ( [id] =>2 [name] => Second [email] =>
second@email.ru )
*/

// Явное указание названия класса User для создания объекта
class User{
  public id, name, email;
}
$sql = "SELECT id, name, email FROM users";
$stmt = $pdo->query($sql);

$obj = $stmt->fetchObject("User");
/*
  Результат:
  User Object ( [id] =>1 [name] => First [email] => first@email.ru )
  User Object ( [id] =>2 [name] => Second [email] => second@email.ru )
*/


/* Установка режимов выборки */

// Явное указание имеющегося объекта
$user = new User();
$stmt->setFetchMode(PDO::FETCH_INTO, $user);
$obj = $stmt->fetch(PDO::FETCH_ASSOC);
/*
  Результат используется один и тот же объект!
  После извлечения последней записи в объекте будет:
  User Object ( [id] =>2 [name] => Second [email] => second@email.ru )
*/

// Явное указание класса User для создания объекта
$stmt -> setFetchMode(PDO::FETCH_CLASS, "User");

// Явное указание класса User для создания объекта
```

```php
// Свойства заполняются значениями после отработки конструктора
$stmt -> setFetchMode(PDO::FETCH_CLASS|PDO::FETCH_PROPS_LATE, "User");
```

Полная выборка

```php
<?
$sql = "SELECT id, name, email FROM users";
$stmt = $pdo->query($sql);

// Получаем массив массивов
$arr = $stmt->fetchAll(PDO::FETCH_ASSOC); // по-
умолчанию PDO::FETCH_BOTH


class User {
  public $id, $name, $email;
}
// Получаем массив объектов класса User
$arr = $stmt->fetchAll(PDO::FETCH_CLASS, 'User');


$sql = "SELECT city, name FROM users";
$stmt = $pdo->query($sql);

// Выбираем данные только из первого поля
$arr = $stmt->fetchAll(PDO::FETCH_COLUMN, 0);

// Группируем значения второго поля по значению
первого поля
$arr = $stmt->
fetchAll(PDO::FETCH_COLUMN|PDO::FETCH_GROUP);

// Выбираем уникальные значения из первого поля
$arr = $stmt->
fetchAll(PDO::FETCH_COLUMN|PDO::FETCH_UNIQUE);

// Используем функцию обратного вызова
function foo($name, $email){
  return $name . ': '. $email . "\n");
}
$arr = $stmt->fetchAll(PDO::FETCH_FUNC, 'foo');
```

Подготовленные запросы

```php
// Именованные псевдопеременные

$sql = 'SELECT email FROM users
          WHERE id = :id AND name = :name';

$stmt = $pdo -> prepare($sql);

$stmt -> execute( [':id' => 5, ':name' => 'John'] );
$john = $stmt->fetchAll();

$stmt -> execute( ['id' => 6, 'name' => 'Mike'] );
$mike = $stmt->fetchAll();


// Неименованные псевдопеременные

$sql = 'SELECT email FROM users
          WHERE id = ? AND name = ?';

$stmt = $pdo -> prepare($sql);

$stmt -> execute( [5, 'John'] );
$john = $stmt->fetchAll();

$stmt -> execute( [6, 'Mike'] );
$mike = $stmt->fetchAll();



/* Привязка параметров */

$id = 5;
$name = 'John';

// Для именованные псевдопеременных
$sql = 'SELECT email FROM users
          WHERE id = :id AND name = :name';
$stmt = $pdo -> prepare($sql);
$stmt -> bindParam(':id', $id, PDO::PARAM_INT);
$stmt -> bindParam(':name', $name, PDO::PARAM_STR);
```

```php
$stmt -> execute();

// Для неименованные псевдопеременных
$sql = 'SELECT email FROM users
          WHERE id = ? AND name = ?';
$stmt = $pdo -> prepare($sql);
$stmt -> bindParam(1, $id, PDO::PARAM_INT);
$stmt -> bindParam(2, $name, PDO::PARAM_STR);
$stmt -> execute();


/* Привязка значений */

$id = 5;
$name = 'John';

// Для именованные псевдопеременных
$sql = 'SELECT email FROM users
          WHERE id = :id AND name = :name';
$stmt = $pdo -> prepare($sql);
$stmt -> bindValue(':id', $id, PDO::PARAM_INT);
$stmt -> bindValue(':name', $name, PDO::PARAM_STR);
$stmt -> execute();

// Для неименованные псевдопеременных
$sql = 'SELECT email FROM users
          WHERE id = ? AND name = ?';
$stmt = $pdo -> prepare($sql);
$stmt -> bindValue(1, $id, PDO::PARAM_INT);
$stmt -> bindValue(2, $name, PDO::PARAM_STR);
$stmt -> execute();



/* Привязка возвращаемых полей к именам переменных */

$sql = 'SELECT id, name, email FROM users';
$stmt = $pdo -> prepare($sql);
$stmt -> execute();

$stmt -> bindColumn(1, $id);
$stmt -> bindColumn(2, $name);
```

```php
$stmt -> bindColumn('email', $email);
// имя поля регистрозависимо!

while($stmt -> fetch(PDO::FETCH_BOUND))
  echo "$id : $name : $email\n";
```

Использование хранимых процедур

```php
$id = 5;
$name = 'John';

$stmt = $db -> prepare('CALL getEmail(?, ?, ?)');

// Параметр IN
$stmt -> bindParam(1, $id, PDO::PARAM_INT);
// Параметр INOUT
$stmt -> bindParam(2, $name, PDO::PARAM_STR|PDO::PARAM_INPUT_OUTPUT);
// Параметр OUT
$stmt -> bindParam(3, $email, PDO::PARAM_STR);

$stmt -> execute();
```

Использование транзакций

```
try {
  $pdo -> beginTransaction();

  // Исполняем много запросов
  // Если все запросы исполнены успешно, то фиксируем
это

  $pdo->commit();

}catch (PDOException $e) {

  // Если хотя бы в одном запросе произошла ошибка,
откатываем всё назад
  $pdo -> rollBack();

}
```

Лабораторная работа 3.2

Использование транзакций

Содержание лабораторной работы 3.2

Использование транзакций

Упражнение 1: Использование транзакции в конструкторе класса

- В текстовом редакторе откройте файл **news\NewsDB.class.php**
- В конструкторе класса создайте транзакцию при создании и заполнении таблиц базы данных
- Транзакция должна обеспечивать создание базы данных, создание таблиц и заполнение таблицы **category**
- В браузер должно выводится сообщение о невозможности создания базы данных
- Сохраните файл **news\NewsDB.class.php**
- Сделайте копию базы данных **news.db** и удалите оригинал
- Запустите браузер
- В адресной строке наберите **http://mysite.local/news/news.php**
- Вы должны увидеть страницу с HTML-формой без новостей
- Если есть ошибки, найдите и исправьте их
- Удалите вновь созданную базу данных **news.db** (не копию!)
- В текстовом редакторе откройте файл **news\NewsDB.php**
- Сделайте ошибку в запросе при заполнении таблицы **category**
- Сохраните файл **news\NewsDB.php**
- Запустите браузер
- В адресной строке наберите **http://mysite.local/news/news.php**
- Вы должны получить сообщение о невозможности создания базы данных
- Обратите внимание на то, что файл **news.db** создается, но нулевой длины
- В конструкторе класса **NewsDB** внесите изменение в условие, которое учитывает не только наличие файла базы данных, но и его размер

Что мы изучили?

- Познакомились с расширением "Объекты данных PHP" - простым и согласованным интерфейсом для доступа к базам данных

Модуль 4

PHP. Уровень 4
Reflection API

Темы модуля

- Введение в Reflection API
- Использование классов и интерфейсов
- Практическое использование механизма отражений
- Обзор PHP repository: PEAR vs Composer
- Обзор Composer
- Обзор phpDocumentor

Reflection: что внутри?

- interface Reflector
- class Reflection

- class ReflectionFunctionAbstract
  - implements Reflector
- class ReflectionFunction
  - extends ReflectionFunctionAbstract
- class ReflectionParameter
  - implements Reflector
- class ReflectionClass
  - implements Reflector
- class ReflectionObject
  - extends ReflectionClass
- class ReflectionMethod
  - extends ReflectionFunction
- class ReflectionProperty
  - implements Reflector
- class ReflectionExtension
  - implements Reflector
- class ReflectionException
  - extends Exception

Класс ReflectionFunctionAbstract

- Свойства
  - public $name
- Методы
  - int getStartLine ( )
  - int getEndLine ( )
  - string getExtensionName ( )
  - string getFileName ( )
  - string getName ( )
  - int getNumberOfParameters ( )
  - int getNumberOfRequiredParameters ( )
  - array getParameters ( )
  - array getStaticVariables ( )
  - bool isInternal ( )
  - bool isUserDefined ( )
  - ReflectionExtension getExtension ( )
  - string getNamespaceName ( )
  - bool isClosure ( )
  - bool isDeprecated ( )
  - string getDocComment ( )
  - abstract void __toString ( )

Класс ReflectionFunction

- Константы
  - int IS_DEPRECATE
- Свойства
  - public $name
- Методы
  - abstract void __toString ( )
  - static string export (string $name [, string $return])
  - mixed invoke ([mixed $parameter [, mixed $...]])
  - string __toString ( )

Класс ReflectionParameter

- Свойства
  - public $name
- Методы
  - bool allowsNull ( )
  - __construct (string $function , string $parameter)
  - ReflectionFunction getDeclaringFunction ( )
  - mixed getDefaultValue ( )
  - string getName ( )
  - int getPosition ( )
  - bool isArray ( )
  - bool isDefaultValueAvailable ( )
  - bool isOptional ( )
  - bool isPassedByReference ( )

Класс ReflectionClass

- Свойства
  - public $name
- Методы
  - __construct (mixed $argument)
  - mixed getConstant (string $name)
  - array getConstants ( )
  - array getDefaultProperties ( )
  - string getDocComment ( )
  - int getStartLine ( )
  - int getEndLine ( )
  - ReflectionExtension getExtension ( )
  - string getExtensionName ( )
  - string getFileName ( )
  - array getInterfaceNames ( )
  - array getInterfaces ( )
  - object getMethod (string $name)
  - array getMethods ( )
  - string getName ( )
  - string getNamespaceName ( )
  - object getParentClass ( )
  - array getProperties ( )
  - ReflectionProperty getProperty (string $name)
  - array getStaticProperties ( )
  - mixed getStaticPropertyValue (string $name [, string $default])
  - bool hasConstant (string $name)
  - bool hasMethod (string $name)
  - bool hasProperty (string $name)

- bool implementsInterface (string $interface)
- bool isAbstract ( )
- bool isFinal ( )
- bool isInstance (object $object)
- bool isInstantiable ( )
- bool isInterface ( )
- bool isInternal ( )
- bool isUserDefined ( )
- object newInstance (mixed $args [, mixed $...])
- void setStaticPropertyValue (string $name , string $value)
- string __toString ( )

Класс ReflectionMethod

- Константы
  - int IS_STATIC
  - int IS_PUBLIC
  - int IS_PROTECTED
  - int IS_PRIVATE
  - int IS_ABSTRACT
  - int IS_FINAL
- Свойства
  - public $name
  - public $class
- Методы
  - ReflectionClass getDeclaringClass ( )
  - mixed invoke (object $object [, mixed $parameter [, mixed $...]])
  - bool isAbstract ( )
  - bool isConstructor ( )
  - bool isDestructor ( )
  - bool isFinal ( )
  - bool isPrivate ( )
  - bool isProtected ( )
  - bool isPublic ( )
  - bool isStatic ( )

Класс ReflectionExtension

- Свойства
  - public $name
- Методы
  - __construct (string $name)
  - array getClasses ( )
  - array getClassNames ( )
  - array getConstants ( )
  - array getDependencies ( )
  - array getFunctions ( )
  - array getINIEntries ( )
  - string getName ( )
  - string getVersion ( )
  - string __toString ( )

Reflection API: примеры

```php
// Получаем экземпляр класса ReflectionClass
$rc = new ReflectionClass('Имя_класса');

// Наследует ли класс тот или иной интерфейс?
$rc->implementsInterface('Имя_интерфейса');

// Имеет ли класс тот или иной метод?
$rc->hasMethod('Имя_метода');

// Получаем экземпляр класса ReflectionMethod
$rm = $rc->getMethod('Имя_метода');

// Является ли метод статическим?
$rm->isStatic();

// Выполнение статического метода
$result = $rm->invoke(null);

// Выполнение обычного метода
$instance = $rc->newInstance();
$result = $rm->invoke($instance);
```

Лабораторная работа 4

Использование Reflection API в системе плагинов

Содержание лабораторной работы 4

Использование Reflection API в системе плагинов

Упражнение 1: Создание класса для манипуляций с плагинами

- В текстовом редакторе откройте файл **favorites\classes \Favorites.class.php**
- Опишите конструктор класса, в котором производится подключение (через include или require) всех классов из папки **classes**. Это можно сделать с помощью функций **opendir()**, **glob()** или с помощью итераторов **DirectoryIterator** и **RecursiveDirectoryIterator**. Возможностей много - выбирайте любую
- Опишите метод **findPlugins()**, который из всех доступных приложению классов выбирает только те классы, которые наследуют интерфейс **IPlugin**
- Для каждого выбранного класса создайте экземпляр класса **ReflectionClass** и добавьте его в массив **$plugins** - закрытое свойство класса
- Вызовите метод **findPlugins()** из конструктора класса
- Опишите метод **getFavorites($methodName)**. Метод принимает в качестве аргумента имя метода (например, 'getLinksItems'), который может присутствовать в классах, наследующих интерфейс **IPlugin**
- В цикле обойдите классы из массива **$plugins** и если нужный метод найден, то необходимо его вызвать и результат добавить в массив **$list**. Не забудьте, что искомый метод может быть статическим!
- Метод **getFavorites()** должен вернуть массив **$list** с результатами отработанных методов
- Сохраните файл **favorites\classes\Favorites.class.php**

Упражнение 2: Подключение плагинов и их использование

- В текстовом редакторе откройте файл **favorites\index.php**
- Создайте объект, экземпляр класса **Favorites**
- Вызовите метод **getFavorites()** последовательно передавая ему в качестве аргумента имена методов: **getLinksItems**, **getArticlesItems** и **getAppsItems** и полученные результаты сохраните в переменных

- Используя полученные результаты заполните HTML-списки **ul**
- Сохраните файл **favorites\index.php**
- Запустите браузер и в адресной строке наберите **http://mysite.local/favorites/**
- Убедитесь, что данные на странице отображаются корректно. Если есть ошибки, найдите их и исправьте

Обзор PHP Repository

- PEAR
  - Пакеты с исходными кодами на языке PHP
  - Возможность установки пакетов
  - Стандарт оформления исходного кода, включая контроль версий
- Composer
  - http://getcomposer.org/download/
  - Настроить системную переменную PATH
    - C:\...\composer\vendor\bin
  - CLI-версия PHP
    - php -v
  - Хранилище пакетов
    - https://packagist.org/
  - Файл composer.json
  - Команды
    - composer install
    - composer update

Обзор phpDocumentor

- Где живёт?
  - http://www.phpdoc.org

- Установка (через Composer)

```
{
   "require":
   {
      "php": ">=5.3.3",
      "phpdocumentor/phpdocumentor":
      "v2.0.1"
   }
}
```

- Запуск в консоли
  - phpdoc -d . -t output

- Основные тэги
  - @access (public|protected|private)
  - @method return name(type $arg1[= val1],[...]) [description]
  - @property[-read|-write] data_type $var_name [description]
  - @return data_type
  - @static
  - @var data_type [description]

Что мы изучили?

- Научились использовать Reflection API
- Произвели обзор PHP репозиториев
- Рассмотрели базовое использование phpDocumentor

Модуль 5

PHP. Уровень 4
Расширение cURL
Регулярные выражения
Пространства имён
Обзор модульного тестирования

Темы модуля

- cURL
  - Введение
  - Основные функции
  - Основные опции
  - Опции для заголовков
- Регулярные выражения
  - Введение
  - Метасимволы
  - Специальные последовательности символов
  - Квантификаторы
  - Модификаторы
  - Функции используемые с регулярными выражениями
- Пространства имён
  - Введение
  - Объявление пространства имён
  - Иерархия пространств имён
  - Правила доступа
  - Импорт и псевдонимы
- Обзор модульного тестирования
  - Тестирование кода
  - Модульное тестирование
  - Использование phpUnit

Модуль 5.1 (cURL)

Расширение cURL

Введение в cURL

- Свободная, кроссплатформенная служебная программа командной строки для передачи файлов по различным протоколам с синтаксисом URL
- Libcurl — это библиотека интерфейса API для передачи, которую разработчики могут встроить в свои программы
- cURL действует как автономная обёртка для библиотеки libcurl
- Для libcurl имеется более 30 различных привязок к языкам программирования: **php_curl**

- Основные функции
  - resource curl_init ([string $url = NULL])
  - bool curl_setopt (resource $ch , int $option , mixed $value)
  - mixed curl_exec (resource $ch)
  - void curl_close (resource $ch)

Использование cURL

```php
<?php
// Создание
$ch = curl_init();

// Установка опций
curl_setopt($ch, CURLOPT_URL, "http://site.ru");

// Выполнение
curl_exec($ch);

// Закрытие
curl_close($ch);
?>
```

- Опции для функции curl_setopt
  - CURLOPT_RETURNTRANSFER: boolean
  - CURLOPT_HEADER: boolean
  - CURLOPT_NOBODY: boolean
  - CURLOPT_FILE: stream resource
  - CURLOPT_WRITEHEADER: stream resource
  - CURLOPT_POST: boolean
  - CURLOPT_POSTFIELDS: mixed
  - CURLOPT_PUT: boolean
  - CURLOPT_INFILE: stream resource
  - CURLOPT_INFILESIZE: integer
  - CURLOPT_HEADERFUNCTION: string
  - CURLOPT_BINARYTRANSFER: boolean

- Опции для заголовков
  - CURLOPT_COOKIE: string
  - CURLOPT_ENCODING: string
  - CURLOPT_REFERER: string
  - CURLOPT_USERAGENT: string
  - CURLOPT_USERPWD: string
  - CURLOPT_HTTPHEADER: array
  - CURLOPT_HTTP200ALIASES: array

- Получение информации
  - mixed curl_getinfo (resource $ch [, int $opt = 0])
    - CURLINFO_HTTP_CODE
    - CURLINFO_FILETIME
    - CURLINFO_PRIMARY_IP
    - CURLINFO_LOCAL_IP
    - CURLINFO_HEADER_SIZE
    - CURLINFO_SIZE_DOWNLOAD
    - CURLINFO_CONTENT_TYPE
    - CURLINFO_CONTENT_LENGTH_DOWNLOAD
    - ...

Модуль 5.2 (RegExp)

Регулярные выражения

Введение в Регулярные выражения

- Типы регулярных выражений в PHP
  - PCRE
  - ~~POSIX~~
- Формат определения шаблонов
  - <разделитель><шаблон><разделитель>[<модификаторы>]
- Разделители
  - /
  - |
  - @
  - #
- Основные термины
  - Метасимволы
  - Квантификаторы
  - Классы символов
  - Ссылки
- Функции поиска, замены, разделения
  - `preg_match($pattern, $subject [,$matches]);`

Базовое использование

```
// . Любой символом, кроме символа перевода строки.

preg_match('/./', 'PHP 5', $matches);
echo $matches[0]; // P

preg_match('/PHP.5/', 'PHP 5', $matches);
echo $matches[0]; // PHP 5

preg_match('/PHP.5/', 'PHP-5', $matches);
echo $matches[0]; // PHP-5

preg_match('/PHP.5/', 'PHP5', $matches);
echo $matches[0]; //


// \ Экранирование метасимволов и разделителей

preg_match('/.com/', 'site.com', $matches);
echo $matches[0]; // .com

preg_match('/.com/', 'site-com', $matches);
echo $matches[0]; // -com

preg_match('/\.com/', 'site-com', $matches);
echo $matches[0]; //

preg_match('/\.com/', 'site.com', $matches);
echo $matches[0]; // .com
```

Повторения

```
/*
 {m} точное вхождение
 {m,n} минимум и максимум
 {m,} минимум
*/

preg_match('/tre{1,2}f/', 'trf', $matches);
echo $matches[0]; //

preg_match('/tre{1,2}f/', 'tref', $matches);
echo $matches[0]; // tref

preg_match('/tre{1,2}f/', 'treef', $matches);
echo $matches[0]; // treef

preg_match('/tre{1,2}f/', 'treeef', $matches);
echo $matches[0]; //

preg_match('/fo{2,}ba{2}r/', 'foobaar', $matches);
echo $matches[0]; // foobaar

preg_match('/fo{2,}ba{2}r/', 'fooooooobaar',
$matches);
echo $matches[0]; // fooooooobaar

preg_match('/fo{2,}ba{2}r/', 'fobaar', $matches);
echo $matches[0]; //


/* Квантификаторы
 ? что и {0,1}
 + что и {1,}
 * что и {0,}
*/

preg_match('/PHP.?5/', 'PHP 5', $matches);
echo $matches[0]; // PHP 5

preg_match('/PHP.?5/', 'PHP5', $matches);
```

```php
echo $matches[0]; // PHP5

preg_match('/a+b/', 'caaabc', $matches);
echo $matches[0]; // aaab

preg_match('/a+b/', 'cabc', $matches);
echo $matches[0]; // ab

preg_match('/a+b/', 'cbc', $matches);
echo $matches[0]; //

preg_match('/a*b/', 'caaaabc', $matches);
echo $matches[0]; // aaaab

preg_match('/a*b/', 'cbc', $matches);
echo $matches[0]; // b
```

Метасимволы

```php
// ^ Ограничение начала строки
preg_match('/^abc/', 'abcd', $matches);
echo $matches[0]; // abc

preg_match('/^abc/', 'xabcd', $matches);
echo $matches[0]; //

// $ Ограничение конца строки
preg_match('/xyz$/', 'abcdxyz', $matches);
echo $matches[0]; // xyz

preg_match('/xyz$/', 'xyza', $matches);
echo $matches[0]; //

// [...] Класс искомых символов.
preg_match('/[0-9]+/', 'PHP is released in 1995',
$matches);
echo $matches[0]; // 1995

preg_match('/[^0-9]+/', 'PHP is released in 1995',
$matches);
echo $matches[0]; // PHP is released in

preg_match('/[a-zA-Z ]+/', 'PHP is released in 1995',
$matches);
echo $matches[0]; // PHP is released in

preg_match('/[^a-zA-Z ]+/', 'PHP is released in 1995',
$matches);
echo $matches[0]; // 1995


// (...) Группировка элементов.
$subject = 'PHP is released in 1995';
$pattern = '/PHP [a-zA-Z ]+([12][0-9])([0-9]{2})/';
preg_match($pattern, $subject, $matches);
print_r($matches);
// [0]=>PHP is released in 1995, [1]=>19, [2]=>05
```

Специальные последовательности

```php
// \? \+ \* \[ \] \{ \} \\ Экранирование
$subject = '4**';
$pattern_in_apos = '/^4\*\*$/';
$pattern_in_quot = "/^4\\*\\*$/";

$subject = 'PHP\5';
$pattern_in_apos = '/^PHP\\\5$/';
$pattern_in_quot = "/^PHP\\\\5$/";

/*
 \t \n \f \r (ASCII 9, 10, 12, 13)
 \d ( [0-9] )
 \D ( [^0-9] )
 \s ( [\t\n\f\r ] )
 \S ( [^\t\n\f\r ] )
 \w ( Любая буква, цифра, символ подчеркивания )
 \W ( Противоположность \w )
*/

// \b ( Позиция между соседними символами \w и \W )
$string = "##Testing123##";
preg_match('/\b.+\b/', $string, $matches);
echo $matches[0]; // Testing123
//\B ( Противоположность \b )


// Жадные квантификаторы: * и +
$subject = '<b>I am bold.</b> <i>I am italic.</i> <b>I
am also bold.</b>';
preg_match('#<b>(.+)</b>#', $subject, $matches);
echo $matches[1];
// I am bold.</b> <i>I am italic.</i> <b>I am also
bold.

// Таблетка от жадности: ?
preg_match('#<b>(.+?)</b>#', $subject, $matches);
echo $matches[1]; // I am bold.
```

Модификаторы

```
// i ( [a-zA-Z] )

// m Многострочный поиск
$subject = "ABC\nDEF\nGHI";
preg_match('/^DEF/', $subject, $matches);
echo $matches[0]; //

preg_match('/^DEF/m', $subject, $matches);
echo $matches[0]; // DEF

// S Однострочный поиск: "." = . + перевод строки
$subject = "ABC\nDEF\nGHI";
preg_match('/BC.DE/', $subject, $matches);
echo $matches[0]; //

preg_match('/BC.DE/S', $subject, $matches);
echo $matches[0]; // BC\nDE

// x Пропуск пробелов и комментариев(#) в тексте
шаблона
$subject = "ABC\nDEF\nGHI";
preg_match('/A B C/', $subject, $matches);
echo $matches[0]; //

preg_match('/A B C/x', $subject, $matches);
echo $matches[0]; // ABC

// D Что и $, если строка не заканчивается \n
preg_match('/BC$/', 'ABC\n', $matches);
echo $matches[0]; // BC

preg_match('/BC$/D', 'ABC\n', $matches);
echo $matches[0]; //

// A Что и ^ (начало строки)
preg_match('/[a-c]{3}/i', '123ABC', $matches);
echo $matches[0]; // ABC

preg_match('/[a-c]{3}/iA', '123ABC', $matches);
```

```php
echo $matches[0]; //

// U Ленивость по-умолчанию
$subject = '<b>I am bold.</b> <i>I am italic.</i> <b>I
am also bold.</b>';
preg_match('#<b>(.+)</b>#U', $subject, $matches);
echo $matches[1]; // I am bold.
```

Функции

```
// Функции поиска

$subject = '<b>I am bold.</b> <i>I am italic.</i>';
$pattern = '#<[^>]+>(.*)</[^>]+>#U';

preg_match($pattern, $subject, $matches);
print_r($matches); // [0]=><b>I am bold.</b>, [1]=>I
am bold.

preg_match_all($pattern, $subject, $matches,
PREG_PATTERN_ORDER);
print_r($matches);
// [0][0] => <b>I am bold.</b>, [0][1] => <i>I am
italic.</i>
// [1][0] => I am bold., [1][1] => I am italic.

preg_match_all($pattern, $subject, $matches,
PREG_SET_ORDER);
print_r($matches);
// [0][0] => <b>I am bold.</b>, [0][1] => I am bold.
// [1][0] => <i>I am italic.</i>, [1][1] => I am
italic.


// Функция замены
$subject = 'April 15, 2003';
$pattern = '/(\w+) (\d+), (\d+)/i';
$replace = '$2 $1, $3'; // "\$2 \$1, \$3"
echo preg_replace($pattern, $replace, $subject);


// Функция разделения
$subject = 'hypertext language, programming';
$pattern = '/[\s,]+/';
$words = preg_split($pattern, $subject);
print_r($words);
// [0]=>hypertext, [1]=>language, [2]=>programming

$chars = preg_split('//', 'PHP', 0,
```

```
PREG_SPLIT_NO_EMPTY);
print_r($chars); // [0] => P, [1] => H, [2] => P
```

Модуль 5.3 (Namespaces)

Пространства имён

Namespaces - что это?

- Решаемые проблемы
  - Пересечение имен функций, констант и классов
  - Использование псевдонимов для удобства использования
- Что объявляем?
  - Классы
  - Функции
  - Константы
- Разделитель
  - \
- Псевдоконстанта
  - __NAMESPACE__

Объявление пространства имён

- Объявление

```php
<?php
  namespace MyModule; // самая первая строка!

  const CONNECT_OK = 1;
  class Connection { /* ... */ }
  function connect() { /* ... */  }
?>
```

- Иерархия пространств имён

```php
<?php
  namespace MyModule\Sub\Level;

  const CONNECT_OK = 1;
  class Connection { /* ... */ }
  function connect() { /* ... */  }
?>
```

- Объявления в одном файле (не рекомендуется)

```php
<?php
  namespace MyModule{
    const CONNECT_OK = 1;
    class Connection { /* ... */ }
    function connect() { /* ... */  }
  }
  /* Никакого пространства между блоками! */
  namespace AnotherModule{
    const CONNECT_OK = 1;
    class Connection { /* ... */ }
    function connect() { /* ... */  }
  }
?>
```

- Лучше, но не надо

```php
<?php
  namespace MyModule{
    const CONNECT_OK = 1;
```

```
    class Connection { /* ... */ }
    function connect() { /* ... */  }
  }
  namespace{
    /*
    Глобальное пространство имён
    Здесь основной код
    */
  }
?>
```

- Надо так
  - Файл MyModule.php

```
<?php
  namespace MyModule;
  /* ... */
?>
```

  - Файл AnotherModule.php

```
<?php
  namespace AnotherModule;
  /* ... */
?>
```

  - Файл index.php

```
<?php
  require_once "MyModule.php";
  require_once "AnotherModule.php";
  /*
    А это глобальное пространство имён
    Здесь наш код
  */
?>
```

- Объединение пространств имён
  - Файл MyModule.php

```
<?php
  namespace MyModule;
  /* ... */
?>
```

  - Файл AnotherModule.php

```
<?php
```

```php
  namespace AnotherModule;
  require_once "MyModule.php";
  /* ... */
?>
```

Правила доступа

- Unqualified name

```php
<?php
namespace MyModule
const E_ALL = 100;
echo E_ALL; // Локальная константа
echo \E_ALL; // Глобальная константа
?>
```

- Qualified name

```php
<?php
require_once "MyModule.php";
echo MyModule\E_ALL; // Константа из MyModule
echo \E_ALL; // Глобальная константа
?>
```

- Fully qualified name

```php
<?php
require_once "MyModule.php";
echo \MyModule\E_ALL; // Константа из MyModule
echo \E_ALL; // Глобальная константа
?>
```

Импорт и псевдонимы

```php
<?php
namespace Foo;
// Здесь подключаются все необходимые файлы
class Bar { /*...*/ }

use My\Full\Classname as Bar;
use My\Full\NSname; // My\Full\NSname as NSname
use \ArrayObject; // импорт глобального класса

$obj = new Bar; // объект класса My\Full\Classname

$obj = new namespace\Bar; // объект класса Foo\Bar

NSname\subns\func(); // функция My\Full\NSname\subns
\func

$obj = new ArrayObject(array(1)); // глобальный ArrayO
bject
```

Модуль 5.3 (Unit testing)

Обзор модульного тестирования

Тестирование кода

- Тесты можно считать хорошими, если их:
  - Легко научиться писать
  - Легко написать
  - Легко прочитать
  - Легко выполнить
  - Быстро выполняются
  - Поддерживается изолированность
  - Поддерживается комбинируемость
- Ограничения:
  - Легкость обучения написанию тестов противоречит легкости написания
  - Изолированность противоречит скорости выполнения

Модульное тестирование

- Процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы
  - Поощрение изменений
  - Упрощение интеграции
  - Документирование кода
  - Отделение интерфейса от реализации

- phpUnit
  - http://www.phpunit.de
  - Установка (через Composer)

```
{
  "require":
  {
    "php": ">=5.3.3",
    "phpunit/phpunit": "3.7.24"
  }
}
```

  - Запуск в консоли
    - phpunit test-php-file.php

Использование phpUnit

```php
<?php
// Использование утверждений
class MyClassTest extends PHPUnit_Framework_TestCase{

  public function testMyFunction(){
    echo $this->assertEquals(10, 10); // true
    echo $this->assertEquals(10, 9); // false

    echo $this->assertTrue(1); // true
    echo $this->assertTrue(0); // false

    echo $this->assertFalse(1); // false
    echo $this->assertFalse(0); // true
  }
}
?>
```

```php
<?php
// Использование зависимостей
class MyClassTest extends PHPUnit_Framework_TestCase{

  public function testOne(){
    return 1;
  }
  /**
  * @depends testOne
  */
  public function testTwo($num){
    echo $this->assertEquals($num, 1);
  }
}
?>
```

```php
<?php
// Использование источников данных
class MyClassTest extends PHPUnit_Framework_TestCase{

  /**
  * @dataProvider provider
```

```
  */
  public function testTwo($num1, $num2){
    echo $this->assertEquals($num, $num2);
  }
  public function provider(){
    return [
      [1, 1], [2, 2], [3, 4], [4, 4]
    ];
  }
}
?>

<?php
// Тестовые окружения
class MyClassTest extends PHPUnit_Framework_TestCase{

  public static function setUpBeforeClass(){
    echo "Start\n";
  }
  protected function setUp(){
    echo "Test start\n";
  }
  protected function assertPreConditions(){
    echo "Assertion start\n";
  }
  public function testOne(){
    echo $this->assertTrue(true);
  }
  protected function assertPostConditions(){
    echo "Assertion finish\n";
  }
  protected function tearDown(){
    echo "Test finish\n";
  }
  public static function tearDownAfterClass(){
    echo "Finish\n";
  }
}
?>
```

Использование покрытия

- phpunit --coverage-html "куда" "файл с тестами"

Project

Current directory: C:\Users\Public\Zend\Apache2\htdocs\phpunit

Legend: Low: 0% to 35% | Medium: 35% to 70% | High: 70% to 100%

	Coverage								
	Lines			Functions / Methods			Classes		
Total		44.44%	4 / 9		66.67%	4 / 6		33.33%	1 / 3
tests		80.00%	4 / 5		80.00%	4 / 5		50.00%	1 / 2
all.php		0.00%	0 / 4		0.00%	0 / 1		0.00%	0 / 1

Generated by PHPUnit 3.4.14 and Xdebug 2.1.2 using PHP 5.3.2 at Wed Sep 14 12:33:07 MSD 2011.

Project

Current file: C:\Users\Public\Zend\Apache2\htdocs\phpunit\tests\classes\demo.php

Legend: executed | not executed | dead code

	Coverage								
	Classes			Functions / Methods			Lines		
Total		0.00%	0 / 1		66.67%	2 / 3		66.67%	2 / 3
Demo		0.00%	0 / 1		66.67%	2 / 3		66.67%	2 / 3
public function sum($a,$b)					100.00%	1 / 1		100.00%	1 / 1
public function dummy($a,$b)					0.00%	0 / 1		0.00%	0 / 1
public function subtract($a,$b)					100.00%	1 / 1		100.00%	1 / 1

```
  1                 : <?php
  2                 : class Demo {
  3                 :
  4                 :     public function sum($a,$b) {
  5               1 :         return $a+$b;
  6                 :     }
  7                 :     public function dummy($a,$b) {
  8               0 :         return "nothing";
  9                 :         if($a==100) $b=0;
 10                 :     }
 11                 :     public function subtract($a,$b) {
 12               1 :         return $a-$b;
```

Что мы изучили?

- Рассмотрели основные возможности расширения cURL
- Познакомились с правилами регулярных выражений
- Уяснили важность пространств имён для больших проектов
- Произвели обзор модульного тестирования

Модуль 6

PHP. Уровень 4
Шаблон проектирования MVC

Темы модуля

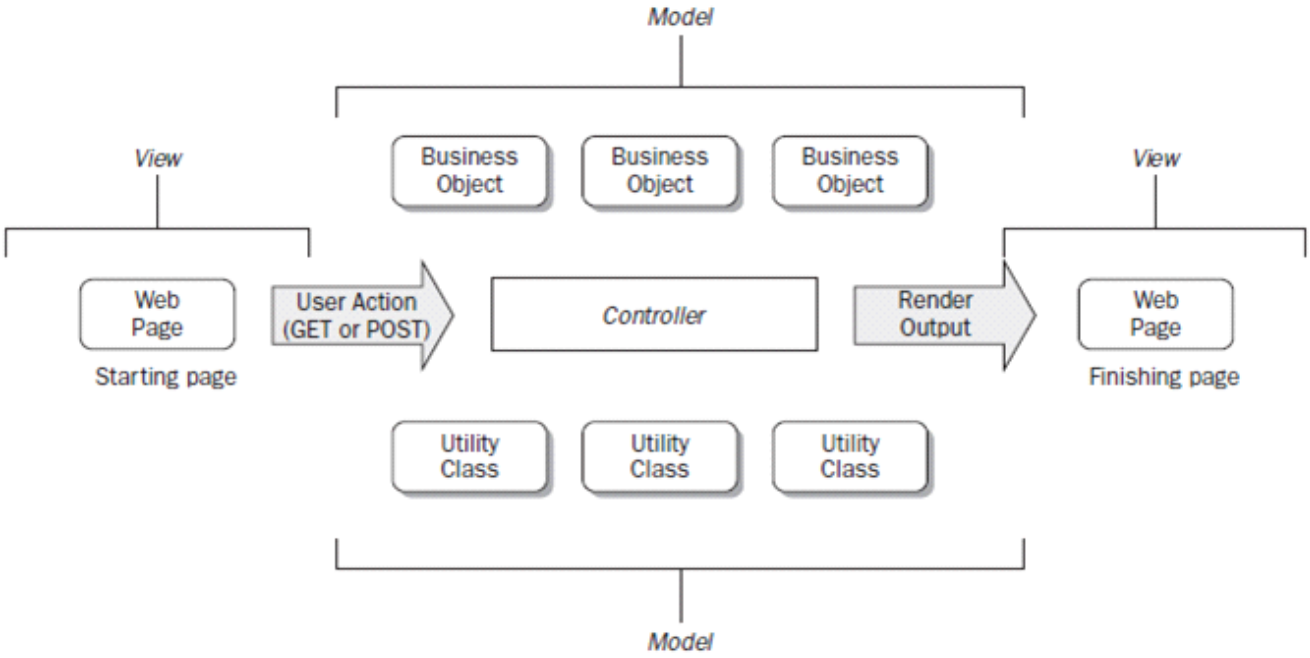
- Введение в MVC
- MVC и ООП
- Обзор PHP фреймворков

Шаблон проектирования MVC

- Знакомый код?

```
<html>
  <head>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <?php
      session_start();
      if (!isset($_SESSION['username'])) {
        echo "Вы не авторизованы на сайте";
      } else {
        echo "Добро пожаловать, " .
$_SESSION['username'];
      }

      $pdo = new PDO('sqlite:site.db');
      $stmt = $pdo->prepare("SELECT * FROM
products");
      $stmt->execute();
      $dbresult = $stmt->
fetchAll(PDO::FETCH_ASSOC);

      foreach ($dbresult as $record) {
        foreach ($record as $k => $v) {
          echo $k . ": " . $v . "<br>";
        }
      }
    ?>
  </body>
</html>
```

- Model-View-Controller



- View

```php
<?php
  include 'controller.php';
?>
<html>
  <head>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <?php
      echo $output;
      if(is_string($dbresult)){
        echo $dbresult;
      }else{
        foreach ($dbresult as $record) {
          foreach ($record as $k => $v) {
            echo $k . ": " . $v . "<br>";
          }
        }
      }
    ?>
  </body>
</html>
```

- Controller

```php
<?php
  include 'model.php';

  session_start();
  if (isset($_SESSION['username'])) {
    $output = "Добро пожаловать, ".
$_SESSION['username'];
  } else {
    $output = "Вы не авторизованы не сайте";
  }
  $dbresult = fetchAllProducts();
?>
```

- Model

```php
<?php
  function getDBConnection() {
    try{
      $pdo = new PDO('sqlite:site.db');
      return $pdo;
    }catch(PDOException $e) {
      return "Извините, " . $e->getMessage());
    }
  }

  function fetchAllProducts() {
    $pdo = getDBConnection();
    if(!is_object($pdo))
      return $pdo;
    try{
      $stmt = $pdo->prepare("SELECT * FROM
products");
      $stmt->execute();
      return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }catch(Exception $e) {
      return "Извините, " . $e->getMessage());
    }
  }
?>
```

MVC и ООП

- Controllers
  - FrontController (как правило, встроенный)
    - интерпритация переменных запроса и направление исполняемого кода
    - выполняет всю черновую работу: создание моделей, парсинг шаблонов, вывод результата
  - ActionControllers (пользовательские)
    - action == метод
- Models
  - классы-утилиты
- Views
  - шаблоны

MVC приложение

- Маршрутизация
  - /controller/action[/key 1][/key 2]...
  - /controller/action[/key 1][/value 1]...
  - Примеры:
    - /
    - /book/show
    - /book/show/id/25
    - /book/show/category/php/year/2012
    - /book/get/format/xml

- mod_rewrite => bootstrap file (index.php)

- Структура
  - application
    - contrlollers
    - models
    - views
  - images
  - styles
  - index.php (bootstrap)
  - .htaccess
    - RewriteCond %{REQUEST_FILENAME} !-f
    - RewriteRule !\.(js|gif|jpg|png|css)$ index.php

- Bootstrap

- Инициализация
  - путей по-умолчанию для поиска файлов
  - необходимых конфигурационных данных
  - автозагрузки классов
- Инициализация FrontController
- Роутинг
- Вывод данных

- FrontController
  - Конструктор
    - определение необходимого контроллера
    - определение необходимого действия
    - парсинг параметров
  - Роутинг
    - инициализация текущего контроллера
    - инициализация текущего действия
  - Вспомогательные методы

- ActionController
  - Действия (actions)
    - инициализация экземпляра класса FrontController
    - инициализация необходимой модели
    - обработка необходимых данных

        - передача данных модели
        - получение результата от модели
        - передача результата во FrontController

- Модель
    - Обработка данных
    - Заполнение представления
    - Возврат результата в контроллер

- Представление
    - Заполнение шаблона

Лабораторная работа 6

Создание и использование MVC фреймворка

Содержание лабораторной работы 6

Создание и использование MVC фреймворка

Упражнение 1: Создание базового поведения

- Перенесите содержимое папки **C:\Users\Public\OpenServer\domains \mysite.local\mvc** на уровень выше, то есть в папку **mysite.local**
- В папке **application\controllers** создайте файл **UserController.php**
- В файле **UserController.php** создайте класс **UserController**, который наследует интерфейс **IController**
- В классе **UserController** создайте действие (action) **hello**, то есть метод **helloAction**
- Метод должен быть отработан при получении URL вида **/user/hello/name/john**
- Получите необходимые параметры и передайте их модели **FileModel**
- Представление для данного действия находится в папке **views**, имя файла - константа **USER_DEFAULT_FILE**
- Сохраните файл **application\controllers\UserController.php**
- Запустите браузер
- В адресной строке наберите **http://mysite.local/user/hello/name/john**
- Вы должны получить строку **Hello, john!**
- Проверьте других пользователей

Упражнение 2: Создание других действий и представлений (выполняется по желанию и при наличие времени)

- В классе **UserController** попробуйте создать действия (actions) для
    - вывода списка пользователей, отрабатывающий URL вида **/user/list/**
    - вывода роли пользователя, отрабатывающий URL вида **/user/get/role/john/**
- Представление для данных действий находятся в папке **views**, имена файлов - константы **USER_LIST_FILE** и **USER_ROLE_FILE**
- Сохраните файл **application\controllers\UserController.php**
- Протестируйте новый функционал приложения

PHP Frameworks

- Выбор фреймворка
  - Архитектура
  - Документация
  - Коммьюнити
  - Поддержка
  - Гибкость

Что мы изучили?

- Познакомились шаблоном проектирования MVC
- Написали базовый MVC фреймворк
- Совершили обзор PHP фреймворков

Модуль 7

PHP. Уровень 4
Создание REST-сервиса

Темы модуля

- Введение в REST
- Знакомство с микро-фреймворком Slim
- Знакомство с библиотекой NotORM
- Базовые операции REST-сервера
- Базовые операции REST-клиента

Что такое REST?

- Representational State Transfe
- Передача состояния представления
- Стиль построения архитектуры распределенного приложения
  - Данные должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON)
  - Сетевой протокол должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ»
  - Использование методов HTTP для указания необходимых операций

Микро-фреймворк Slim

```php
<?php
//http://slimframework.com

// Автозагрузчик
\Slim\Slim::registerAutoloader();

// Создание экземпляра класса
$app = new \Slim\Slim();

// Запуск приложения
$app -> run();

// Роутинг и базовые операции
$app->get('/', function(){
  echo 'Привет, гость!';
});

$app->post('/:id', function($id) use ($app){
  // Получаем значение параметра name
  $name = $app->request()->post('name');
  // Получаем массив всех параметров
  $params = $app->request()->post();
  // Посылаем ответ
  $res = $app->response();
  // Посылаем HTTP заголовок с нужным значением
  $res['Content-Type'] = 'text/xml';
});
```

Библиотека NotORM

```php
<?
//http://www.notorm.com

//Инициализация
$pdo = new PDO('sqlite:db');
$db = new NotORM($pdo);

// Выборка всех записей
foreach($db->users() as $user){}

// Выборка с условием
$user = $db->users()->where('id', $id);
$row = $user->fetch();

// Изменение записи
$user->update($array);

// Вставка записи
$db->users()->insert($array);
```

Что мы изучили?

- Познакомились с популярным подходом к созданию веб-приложения REST
- Получили представление о функционировании REST-сервера на базе микро-фреймворка Slim в связке с библиотекой NotORM

Что почитать?

- Документация PHP

Что дальше?

- XML И XSLT. Современные технологии обработки данных для ВЕБ
- JavaScript. Уровень 3б. AJAX. Разработка веб - приложений для Web 2.0