Игорь Борисов

PHP. Уровень 3. Профессиональная разработка на PHP

---

http://igor-borisov.ru

Темы курса

- Объектно-ориентированное программирование
- Практическое использование ООП с базой данных SQLite
- PHP и XML
- PHP и веб-службы
- Сокеты и сетевые функции
- Работа с графикой

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 3 Объектно-ориентированное программирование

Темы модуля

- ООП и PHP
- Классы и объекты
- Свойства и методы объектов
- Конструкторы и деструкторы
- Клонирование объектов
- Наследование классов
- Перегрузка методов
- Обработка исключений
- Абстрактные классы и методы
- Интерфейсы
- Константы класса
- Статические свойства и методы класса
- Автозагрузка классов
- Модификаторы доступа к свойствам и методам
- "Магические методы"
- Финальные классы и методы
- Типажи
- Уточнение типов
- Полезные функции для классов и объектов

Классы и объекты

```php
// Описание класса
class Pet{
    // тело класса
}


// Создание объекта, экземпляра класса
$cat = new Pet();
$dog = new Pet;


// Проверим
echo gettype($cat); // object
echo is_object($dog); // 1
```

Свойства объектов

```php
// Описание класса
class Pet{
    // Описание свойств
    public $type = "unknown";
    public $name;
}


// Создание объектов, экземпляров класса
$cat = new Pet();
$cat = new Pet();

// Изменяем значения свойств
$cat->type = "cat";
$cat->name = "Murzik";

$dog->type = "dog";
$dog->name = "Tuzik";


// Проверим, какого типа $cat?
echo $cat->type; // cat
// Проверим, как зовут собачку?
echo $dog->name; // Tuzik
```

Методы объектов

```php
// Описание класса
class Pet{
    // Описание свойств
    public $type = "unknown";
    public $name;
    // Описание методов
    function say($word){
      echo "Object said $word";
    }
}

// Создание объектов, экземпляров класса
$cat = new Pet();
$cat = new Pet();
// Изменяем значения свойств
$cat->type = "cat";
$cat->name = "Murzik";
$dog->type = "dog";
$dog->name = "Tuzik";

// Вызываем метод объекта
$cat->say("Myau"); // Object said Myau
$dog->say("Gav"); // Object said Gav
```

Обращение к свойствам и методам внутри класса

```php
// Описание класса
class Pet{
    // Описание свойств
    public $type = "unknown";
    public $name;

    // Описание методов
    function say($word){
      // Доступ к свойству
      echo $this->name . " said $word";
      // Доступ к методу
      $this->drawLine();
    }

    function drawLine(){
      echo "<hr>";
    }

}

// Создание объектов, экземпляров класса
$cat = new Pet();
$cat = new Pet();
// Изменяем значения свойств
$cat->type = "cat";
$cat->name = "Murzik";
$dog->type = "dog";
$dog->name = "Tuzik";

// Вызываем метод объекта
$cat->say("Myau"); // Murzik said Myau
$dog->say("Gav"); // Tuzik said Gav
```

Использование псевдоконсант

```php
// Описание класса
class SuperClass{

    function functionName(){
      echo "<p>Вызвана функция " . __FUNCTION__;
    }

    function className(){
      echo "<p>Используем класс " . __CLASS__;
    }

    function methodName(){
      echo "<p>Вызван метод " . __METHOD__;
    }



}

// Создание объекта
$obj = new SuperClass();

// Используем псевдоконстаты
$obj->functionName(); // functionName
$obj->className(); // SuperClass
$obj->methodName(); // SuperClass::methodName
```

Лабораторная работа 1.1

Создание класса и его экземпляров

Конструкторы и деструкторы

```php
// Описание класса
class Pet{
    // Описание свойств
    public $type = "unknown";
    public $name;

    // Конструктор класса
    function __construct($type, $name){
      $this->type = $type;
      $this->name = $name;
    }

    // Описание методов
    function say($word){
      // Доступ к свойству
      echo $this->name . " said $word";
      // Доступ к методу
      $this->drawLine();
    }

    function drawLine(){
      echo "<hr>";
    }

    // Деструктор класса
    function __destruct(){
      echo $this->name . " removed";
    }
}

// Создание объектов, экземпляров класса
$cat = new Pet("cat", "Murzik");
$cat = new Pet("dog", "Tuzik");

// Вызываем метод объекта
$cat->say("Myau");
$dog->say("Gav");
```

Лабораторная работа 1.2

Использование конструктора и деструктора

Клонирование объектов

```php
// Копирование значений переменных всех типов, кроме объектов
$x = 10;
$y = $x; // $y - копия $x
$y = 20;
echo $y; // 20
echo $x; // 10

// Создание ссылок для всех типов, кроме объектов
$x = 10;
$y = &$x; // $y - ссылка на $x
$y = 20;
echo $y; // 20
echo $x; // 20


class MyClass{

    public $param;

    // Конструктор класса
    function __construct($param){
      $this->param = $param;
    }

    // Перегружаем оператор clone
    function __clone(){
      echo "Object cloned";
    }

}

// Создание объекта
$objX = new MyClass(10); // $objX - ссылка на объект в памяти
$objY = $objX; // $objY - ссылка на $objX

$objY->param = 20;
echo $objX->param; // 20


$objZ = clone $objX; // $objZ копия $objX
$objZ->param = 30;
echo $objX->param; // 20
```

Наследование классов

```php
// Создание супер-класса
class SimpleHouse{

    public $model = "";
    public $square = 0;
    public $floors = 0;
    public $color = "none";

    // Конструктор класса
    function __construct($model, $square = 0, $floors = 1){
      $this->model = $model;
      $this->square = $square;
      $this->floors = $floors;
    }

    function startProject(){
      echo "Start. Model: {$this->model}\n";
    }

    function stopProject(){
      echo "Stop. Model: {$this->model}\n\n";
    }

    function build(){
      echo "Build. House: {$this->square}x{$this->floors}\n";
    }

    function paint(){
      echo "Paint. Color: {$this->color}\n";
    }

}

// Создание простого дома
$simple = new SimpleHouse("A-100-123", 120, 2);
$simple->color = "red";
$simple->startProject();
$simple->build();
$simple->paint();
$simple->stopProject();

// Создание класса-наследника
class SuperHouse extends SimpleHouse{

    public $fireplace = true;
    public $patio = true;
```

```
    function fire(){
      if ($this->fireplace)
        echo "Fueled fireplace\n";
    }
}
$super = new SuperHouse("A-100-125", 320, 3);
$super->color = "green";
$super->startProject();
$super->build();
$super->paint();
$super->fire();
$super->stopProject();

// Создание класса-наследника
class FabricHouse extends SimpleHouse{

    // Перегрузка метода
    function build(){
      echo "Build. Fabric: {$this->square}x{$this->floors}\n";
    }

}

$fabric = new FabricHouse("B-200-007", 3250, 5);
$fabric->color = "white";
$fabric->startProject();
$fabric->build();
$fabric->paint();
$fabric->stopProject();

// Создание класса-наследника
class SuperFabricHouse extends FabricHouse{

    // Перегрузка метода
    function build(){
      echo "=============================================\n";
      // Вызов родительского метода
      echo parent::build();
      echo "=============================================\n";
    }

}

$super_fabric = new SuperFabricHouse("C-201-034", 5150, 7);
$super_fabric->color = "black";
$super_fabric->startProject();
$super_fabric->build();
$super_fabric->paint();
```

```php
$super_fabric->stopProject();
```

Лабораторная работа 1.3

Реализация наследования классов

Обработка исключений

```php
function test($var = false){
  try {
    echo "Start\n";
    if(!$var)
      throw new Exception('$var is false!');
    echo "Continue\n";
  }catch(Exception $e){
    echo "Exception: " . $e->getMessage() . "\n";
    echo "in file: " . $e->getFile() . "\n";
    echo "on line: " . $e->getLine() . "\n";
  }
  echo "The end\n";
}

var_dump(test(), test(1));

// Надо получить ответ от последней функции в цепочке в случае ошибки
// foo() -> bar() -> … -> baz()
// foo() <- bar() <- … <- baz()

function foo(){
  $x = bar();
  if(!$x)
    echo "Плохо";
  else
    echo "Хорошо";
}
function bar(){
  return baz();
}
// ...
function baz(){
  // Что-то делаем
  if(!$param)
    return false;
  return true;
}

// Проброс исключения
function foo(){
  try {
    bar();
    echo "Хорошо";
  }catch(Exception $e){
    echo $e->getMessage();
  }
```

```
}
function bar(){
  baz();
}
// ...
function baz(){
  // Что-то делаем
  if(!$param)
    throw new Exception("Плохо!");
}

// Создание пользовательских исключений
class MathException extends Exception{
  function __construct($msg){
    parent::__construct($msg);
  }

  function someMethod(){
    return __CLASS__;
  }
}

try {
  $x = rand(5, 15);
  $y = rand(0, 1);
  if($y == 0) // Генерируем собственное исключение
    throw new MathException("На 0 делить нельзя!");
  if($y < 0) // Генерируем встроенное исключение
    throw new Exception("Что-то случилось!");
  echo $x / $y;
}catch(MathException $e){
  echo $e->someMethod() . ":" . $e->getMessage();
}catch(Exception $e){
  echo $e->getMessage();
}

// Внимание!
try {
  $x = rand(5, 15);
  $y = rand(0, 1);
  if($y == 0) // Генерируем собственное исключение
    throw new MathException("На 0 делить нельзя!");
  if($y < 0) // Генерируем встроенное исключение
    throw new Exception("Что-то случилось!");
  echo $x / $y;

}catch(Exception $e){
  echo $e->getMessage(); // Попадём сюда!
}catch(MathException $e){
```

```
  echo $e->someMethod() . ":" . $e->getMessage();
}

// Финализация
function test($var = false){
  try {
    echo "TRY\n";
    if(!$var)
      throw new Exception("Error");
  }catch(Exception $e){
    echo "CATCH\n";
  }finally{
    echo "FINALLY\n";
  }
}

var_dump(test(), test(1));

// Возвращаем значения
function test($var = false){
  try {
    echo "TRY\n";
    if(!$var)
      throw new Exception("Error");
    return 1;
  }catch(Exception $e){
    echo "CATCH\n";
    return 2;
  }finally{
    echo "FINALLY\n";
    return 3; // Можно закомментировать эту строку
  }
}

var_dump(test(), test(1));
```

Абтрактные классы и методы

```php
// Создание абстрактного класса
class HouseAbstract{
    public $model = "";
    public $square;
    public $floors;

    function __construct($model, $square = 0, $floors = 1){
        if(!$model)
            throw new Exception('Ошибка! Укажите модель!');
        $this->model = $model;
        $this->square = $square;
        $this->floors = $floors;
    }

    function startProject(){
      echo "Start. Model: {$this->model}\n";
    }

    function stopProject(){
      echo "Stop. Model: {$this->model}\n\n";
    }

    // Абстрактный метод
    abstract function build();
}
// Создание супер-класса
class SimpleHouse extends HouseAbstract{

    // Свойства абстрактного класса +
    public $color = "none";

    // Обязательная реализация абстрактного метода
    function build(){
      echo "Build. House: {$this->square}x{$this->floors}\n";
    }

    // Свой метод
    function paint(){
      echo "Paint. Color: {$this->color}\n";
    }

}

// Создание простого дома
$simple = new SimpleHouse("A-100-123", 120, 2);
$simple->color = "red";
```

```php
$simple->startProject();
$simple->build();
$simple->paint();
$simple->stopProject();

// Создание класса-наследника
class SuperHouse extends SimpleHouse{

    public $fireplace = true;
    public $patio = true;

    function fire(){
      if ($this->fireplace)
        echo "Fueled fireplace\n";
    }
}
$super = new SuperHouse("A-100-125", 320, 3);
$super->color = "green";
$super->startProject();
$super->build();
$super->paint();
$super->fire();
$super->stopProject();

// Создание супер-класса
class FabricHouse extends HouseAbstract{

    // Обязательная реализация абстрактного метода
    function build(){
      echo "Build. Fabric: {$this->square}x{$this->floors}\n";
    }

}

$fabric = new FabricHouse("B-200-007", 3250, 5);
$fabric->startProject();
$fabric->build();
$fabric->stopProject();

// Создание класса-наследника
class SuperFabricHouse extends FabricHouse{

    // Перегрузка метода
    function build(){
      echo "=============================================\n";
      // Вызов родительского метода
      echo parent::build();
      echo "=============================================\n";
    }
```

```
}

$super_fabric = new SuperFabricHouse("C-201-034", 5150, 7);
$super_fabric->startProject();
$super_fabric->build();
$super_fabric->stopProject();
```

Интерфейсы

```php
// Создание интерфейса
interface Paintable{
    // Абстрактный метод
    function paint();
}
interface Brick{}
interface Panel{}

// Создание абстрактного класса
class HouseAbstract{
    public $model = "";
    public $square;
    public $floors;

    function __construct($model, $square = 0, $floors = 1){
        if(!$model)
            throw new Exception('Ошибка! Укажите модель!');
        $this->model = $model;
        $this->square = $square;
        $this->floors = $floors;
    }

    function startProject(){
      echo "Start. Model: {$this->model}\n";
    }

    function stopProject(){
      echo "Stop. Model: {$this->model}\n\n";
    }

    // Абстрактный метод
    abstract function build();
}
// Создание супер-класса
class SimpleHouse extends HouseAbstract implements Paintable, Brick{

    // Свойства абстрактного класса +
    public $color = "none";

    // Обязательная реализация абстрактного метода
    function build(){
      echo "Build. House: {$this->square}x{$this->floors}\n";
    }

    // Обязательная реализация абстрактного метода
    function paint(){
```

```
      echo "Paint. Color: {$this->color}\n";
    }

}

// Создание простого дома
$simple = new SimpleHouse("A-100-123", 120, 2);
$simple->color = "red";
$simple->startProject();
$simple->build();
// Проверка класса в цепочке предков
if($simple instanceOf Paintable)
    $simple->paint();
$simple->stopProject();

// Создание класса-наследника
class SuperHouse extends SimpleHouse{

    public $fireplace = true;
    public $patio = true;

    function fire(){
      if ($this->fireplace)
        echo "Fueled fireplace\n";
    }
}
$super = new SuperHouse("A-100-125", 320, 3);
$super->color = "green";
$super->startProject();
$super->build();
// Проверка класса в цепочке предков
if($super instanceOf Paintable)
    $super->paint();
$super->fire();
$super->stopProject();

// Создание супер-класса
class FabricHouse extends HouseAbstract implements Panel{

    // Обязательная реализация абстрактного метода
    function build(){
      echo "Build. Fabric: {$this->square}x{$this->floors}\n";
    }

}

$fabric = new FabricHouse("B-200-007", 3250, 5);
$fabric->startProject();
$fabric->build();
```

```php
// Проверка класса в цепочке предков
if($fabric instanceOf Paintable)
    $fabric->paint();
$fabric->stopProject();

// Создание класса-наследника
class SuperFabricHouse extends FabricHouse{

    // Перегрузка метода
    function build(){
      echo "================================================\n";
      // Вызов родительского метода
      echo parent::build();
      echo "================================================\n";
    }

}

$super_fabric = new SuperFabricHouse("C-201-034", 5150, 7);
$super_fabric->startProject();
$super_fabric->build();
// Проверка класса в цепочке предков
if($super_fabric instanceOf Paintable)
    $super_fabric->paint();
$super_fabric->stopProject();
```

Лабораторная работа 1.4

Использование абстрактных классов и интерфейсов

Константы и статические члены класса

```php
class ConstructionCompany{
    const NAME = "Рога и копыта";

    function printName(){
        // Обращение к константе из метода класса
        echo self::NAME;
    }
}

// Обращение к константе без создания экземпляра класса
echo ConstructionCompany::NAME; // Рога и копыта

$company = new ConstructionCompany();
$company->printName();  // Рога и копыта


class Worker{
    public name;
    // Статическое свойство класса
    public static workerCount = 0;

    function __construct($name){
        if(!$name)
            throw new Exception('Ошибка! Укажите имя рабочего!');
        $this->name = $name;
        // Изменение статического свойства класса
        ++self::$workerCount;
    }

    // Статический метод класса
    static function welcome(){
        // Никаких $this в статическом методе класса!
        echo 'Добро пожаловать на стройплощадку! Нас уже ' . self::$workerCount . "\n";
    }
}
Worker::welcome();
$w1 = new Worker('Вася Пупкин');
$w2 = new Worker('Федя Сумкин');
echo 'Текущее количество рабочих: ' . Worker::$workerCount . "\n";

// Позднее статическое связывание (с PHP 5.4)
// Проблема
class A{
    static function whoAmI(){
        echo __CLASS__;
    }
    static function identity(){
        self::whoAmI();
    }
}
class B extends A{
    static function whoAmI(){
        echo __CLASS__;
    }
```

```
}
B::identity(); // A

// Позднее статическое связывание (с PHP 5.3)
class A{
    static function whoAmI(){
        echo __CLASS__;
    }
    static function identity(){
        static::whoAmI();
    }
}
class B extends A{
    static function whoAmI(){
        echo __CLASS__;
    }
}
B::identity(); // B
```

Лабораторная работа 1.5

Использование статических членов класса

Автоматическая загрузка класса

```php
$myClass = new MyClass(); // Ошибка! Описание класса не найдено


function __autoload($class){
    echo $class;
}
$myClass = new MyClass(); // MyClass. Далее: Ошибка! Описание класса не найдено

// Решение.
// Описываем класс в одноимённом файле, то есть MyClass.class.php
class MyClass{
    function __construct(){
        echo __CLASS__;
    }
}

// В текущем файле подключаем файл с описанием нужного класса
function __autoload($class){
    include $class.'.class.php';
}
$myClass = new MyClass(); // MyClass
```

Лабораторная работа 1.6

Использование автозагрузки классов

Модификаторы доступа к свойствам и методам

```
class ParenClass{

  public $public = 1;
  protected $protected = 2;
  private $private = 3;


  function getProtected(){
    return $this->protected;
  }
  function getPrivate(){
    return $this->private;
  }
}

$parent = new ParentClass();
echo $parent->public;
echo $parent->protected; // нельзя
echo $parent->private; // нельзя

echo $parent->getProtected();
echo $parent->getPrivate();


class ChildClass{
  function getParentProtected(){
    return $this->protected;
  }
  function getParentPrivate(){
    return $this->private;
  }

  function getRealParentPrivate(){
    return $this->getPrivate();
  }
}
$child = new ChildClass();
echo $child->getParentProtected();
echo $child->getParentPrivate();  // нельзя

echo $child->getRealParentPrivate();

// Использование модификаторов
class HouseAbstract{
    private $model = "";
```

```
    private $square;
    private $floors;

    function __construct($model, $square = 0, $floors = 1){
        if(!$model)
            throw new Exception('Ошибка! Укажите модель!');
        $this->model = $model;
        $this->square = $square;
        $this->floors = $floors;
    }

    function getModel(){
      return $this->model;
    }

    function getSquare(){
      return $this->square;
    }

    function getFloors(){
      return $this->floors;
    }
    // Описание других методов
}

class SimpleHouse{

    private $color = "none";

    function getColor(){
      return $this->color;
    }

    function setColor(){
      return $this->color;
    }
    // Описание других методов
}

$simple = new SimpleHouse("A-100-123", 120, 2);
$simple->setColor("red");
```

Доступ к недоступным свойствам и методам

```
class MyClass{}

$obj = new MyClass();
obj->param = 100;
echo obj->param; // 100
obj->func(10, 20);  // Ошибка!

class MyClass{

  function __set($name, $value){
    echo "Set property '$name' to value $value";
  }

  function __get($name){
    return "Access to property '$name'";
  }

  function __call($name, $args){
    echo "Call method '$name' with arguments: " . implode(', ', $args);
  }

  static function __callStatic($name, $args){
    echo "Call static method '$name' with arguments: " . implode(', ', $args);
  }
}

$obj = new MyClass();
obj->param = 100; // Set property 'param' to value 100
echo obj->param; // Accsess to property 'param'
obj->func(10, 20);  // Call method 'func' width arguments: 10, 20
MyClass::staticFunc(10, 20);  // Call static method 'staticFunc' width arguments: 10, 20



// Использование "магических" сеттеров и геттеров
class HouseAbstract{
    private $model = "";
    private $square;
    private $floors;

    function __construct($model, $square = 0, $floors = 1){
        if(!$model)
            throw new Exception('Ошибка! Укажите модель!');
        $this->model = $model;
        $this->square = $square;
        $this->floors = $floors;
    }

    function __get($name){
      switch($name){
        case 'model': return $this->model;
        case 'square': return $this->square;
        case 'floors': return $this->floors;
        default: throw new Exception('Неизвестное свойство!');
      }
```

```
    }
    // Описание других методов
}

class SimpleHouse{

    private $color = "none";

    function __get($name){
      switch($name){
        case 'color': return $this->color;
        default: throw new Exception('Неизвестное свойство!');
      }
    }

    function __set($name, $value){
      switch($name){
        case 'color': $this->color = $value; break;
        default: throw new Exception('Неизвестное свойство!');
      }
    }
    // Описание других методов
}

$simple = new SimpleHouse("A-100-123", 120, 2);
$simple->color("red");
echo $simple->color;
```

Ещё немного магии

```php
// Преобразование объекта в строку

class MyClass{}

$obj = new MyClass();
echo obj; // Ошибка!

class MyClass{

  function __toString(){
    return 'Это объект, экземпляр класса ' . __CLASS__;
  }

}

$obj = new MyClass();
echo obj; // Это объект, экземпляр класса MyCLASS


// Обращение к объекту как к функции

class Math{

  function __invoke($num, $action){

    switch($action){
      case '+': return $num + $num;
      case '*': return $num - $num;
      default: throw new Exception('Неизвестное свойство!');
    }
  }

}

$obj = new Math();
echo obj(5, '+'); // 10
echo obj(5, '*'); // 25


// Сериализация объекта

class User{
    private $login;
    private $password;
    private $params = [];
```

```
    function __construct($login, $password){
        $this->login = $login;
        $this->password = $password;
        $this->params = $this->getUser();
    }

    function getUser(){
     // Что-то делаем
     // Например, возвращаем массив данных пользователя
    }

    // Вызывается перед сериализацией
    function __sleep(){
      return ['login', 'password'];
    }

    // Вызывается после сериализации
    function __wakeup(){
      $this->params = $this->getUser();
    }
}

$obj = new User("john", "1234");
$str = serialize($obj);
unset($obj);
$obj = unserialize($str);
```

Финальные классы и методы

```php
// Финальный класс
final class Breakfast{
  function eatFood($food){
    echo "Съели $food";
  }
}
class McBreakfast extends Breakfast{}

$obj = new McBreakfast(); // Ошибка!


class Math{
  // Финальный метод
  final function sum($num1, $num2){
    echo 'Сумма: ' . $num1 + $num2;
  }
}

class Algebra extends Math{
  function sum($num1, $num2){
    $result = $num1 + $num2;
    echo "Сумма: $num1 и $num2 = $result";
  }
}

$obj = new Algebra(); // Ошибка!
```

Типажи (traits)

```php
// Базовое использование

trait Hello{
  function getGreet(){
    return "Hello";
  }
}

trait User{
  function getUser($name){
    return ucfirst($name);
  }
}

class Welcome{
  use Hello, User;
}

$obj = new Welcome();
echo $obj->getGreet(), ', ', $obj->getUser('john'), '!';
// Hello, John!


// Наследование
trait Greeting{
  use Hello, User;

  function sayHello($name){
    echo $obj->getGreet(), ', ', $obj->getUser($name), '!';
  }

}

class Welcome{
  use Greeting;
}

(new Welcome())->sayHello('john');


// Изменение модификаторов доступа
trait Hello{
  private function sayHello($name){
    return "Hello, $name!";
  }
}
```

```php
class Welcome{
  use Hello{
    sayHello as public;
  }
}

(new Welcome())->sayHello('John');



// Разрешение конфликтов имён
trait Hello{
  private function sayHello(){
    return "Hello";
  }
}

trait User{
  public function sayHello($name){
    return $name;
  }
}

class Welcome{
  use User, Hello{
    Hello::sayHello as public word;
    User::sayHello insteadof Hello;
  }
}

$obj = new Welcome();
echo $obj->word(), ', ', $obj->sayHello('John'), '!';
```

Уточнение типа и полезные функции

```php
// Базовое использование
class MyClass{}
$my = new MyClass();
$std = new stdClass();

// Ожидается передача объекта, экземпляра класса MyClass
function foo(MyClass $obj){}

foo($my); // Отработает успешно
foo($std); // Ошибка!


class MyClass{

  function func(){
    return __METHOD__;
  }

  static function staticFunc(){
    return __METHOD__;
  }

  function __invoke(){
    return __METHOD__;
  }
}

$obj = new MyClass();

// Ожидается то, что можно вызвать
function foo(callable $x){
  if(func_num_args() == 2){
    $m = func_get_arg(1);
    return $x->$m();
  }elseif(is_array($x)){
    return $x[0]::$m[1]();
  }else{
    return $x();
  }
}

echo foo($obj, "func"); // MyClass::func
echo foo(["MyClass", "staticFunc"]); // MyClass::staticFunc
echo foo($obj); // MyClass::__invoke
```

```
// Полезные функции для классов и объектов
http://php.net/manual/ru/ref.classobj.php
```

Что мы изучили?

- Познакомились с парадигмой ООП
- Уяснили специфику реализации ОО парадигмы в PHP

Лабораторные работы

- Лабораторная работа 1.1
- Лабораторная работа 1.2
- Лабораторная работа 1.3
- Лабораторная работа 1.4
- Лабораторная работа 1.5
- Лабораторная работа 1.6

Лабораторная работа 1.1

Создание класса и его экземпляров

Упражнение 1: Создание класса

- Создайте класс **User**
- В текстовом редакторе откройте файл **oop\users.php**
- В классе создайте свойства **name**, **login** и **password**
- В классе создайте и опишите метод **`showInfo()`**, который выводит информацию о пользователе в произвольной форме
- Создайте три объекта, экземпляра класса **User**: **$user1**, **$user2** и **$user3**
- Задайте произвольные значения свойств **name**, **login** и **password** для каждого из объектов
- Вызовите метод **`showInfo()`** для каждого объекта
- Сохраните файл **oop\users.php**

Упражнение 2: Вывод данных в браузер

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/oop/users.php**
- Проверьте работу скрипта. Если есть ошибки, найдите их и исправьте

Лабораторная работа 1.2

Использование конструктора и деструктора

Упражнение 1: Создание конструктора

- В текстовом редакторе откройте файл **oop\users.php**
- В классе **User** создайте и опишите конструктор, который принимает в качестве аргументов **имя**, **логин** и **пароль** пользователя
- Конструктор должен инициализировать свойства **name**, **login** и **password**
- Измените код, который инициализирует объекты, передавая нужные параметры в конструктор
- Удалите те строки кода, в которых задаются значения свойств объектов

Упражнение 2: Создание деструктора

- В классе **User** создайте и опишите деструктор
- Деструктор должен выводить строку **Пользователь [логин_пользователя] удален**
- Подставьте вместо подстроки **[логин_пользователя]** значение свойства **login**
- Сохраните файл **oop\users.php**

Упражнение 3: Проверка работы скрипта

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/oop/users.php**
- Проверьте работу скрипта. Если есть ошибки, найдите их и исправьте

Лабораторная работа 1.3

Реализация наследования классов

Упражнение 1: Создание класса-наследника

- В текстовом редакторе откройте файл **oop\users.php**
- Создайте и опишите класс **SuperUser**, наследованный от класса **User**
- В классе **SuperUser** создайте свойство **role**
- Перегрузите конструктор супер-класса так, чтобы он принимал четвёртым параметром значение для свойства **role**
- Вызовите из конструктора родительский конструктор и передайте в него первые три параметра
- Перегрузите метод супер-класса **showInfo()** так, чтобы выводилось и значение свойства **role**
- Создайте объект **$user**, экземпляр класса **SuperUser**
- Вызовите метод **showInfo()** объекта **$user**
- Сохраните файл **oop\users.php**

Упражнение 2: Проверка работы скрипта

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/oop/users.php**
- Проверьте работу скрипта. Если есть ошибки, найдите их и исправьте

Лабораторная работа 1.4

Использование абстрактных классов и интерфейсов

Упражнение 1: Создание и использование абстрактного класса

- В текстовом редакторе откройте файл **oop\users.php**
- Создайте и опишите абстрактный класс **UserAbstract**
- В классе **UserAbstract** объявите абстрактный метод **showInfo()**
- Обновите класс **User**, унаследовав его от абстрактного класса **UserAbstract**
- Если требуется, внесите в класс **User** необходимые изменения
- Сохраните файл **oop\users.php**
- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/oop/users.php**
- Проверьте работу скрипта. Если есть ошибки, найдите их и исправьте

Упражнение 2: Создание и использование интерфейса

- В текстовом редакторе откройте файл **oop\users.php**
- Создайте и опишите интерфейс **ISuperUser**
- В интерфейсе **ISuperUser** объявите метод **getInfo()**
- Обновите класс **SuperUser**, унаследовав его от интерфейса **ISuperUser**
- Создайте и опишите метод **getInfo()** в классе **SuperUser**
- Метод **getInfo()** должен возвращать ассоциативный массив, в котором именами элементов массива являются имена свойств объекта, а значениями элементов - значения свойств объекта
- Вызовите метод **getInfo()** для экземпляра класса **SuperUser**
- Используйте функцию **print_r()** для просмотра данных, полученных с помощью метода **getInfo()**
- Сохраните файл **oop\users.php**
- Запустите браузер

- Наберите в адресной строке браузера **http://mysite.local/oop/users.php**
- Проверьте работу скрипта. Если есть ошибки, найдите их и исправьте

Лабораторная работа 1.5

Использование статических членов класса

Упражнение 1: Уяснение задачи

- Посчитать количество созданных экземпляров класса **User**
- Посчитать количество созданных экземпляров класса **SuperUser**
- Пользователи считаются отдельно для каждого класса

Упражнение 2: Создание статических свойств классов

- В текстовом редакторе откройте файл **oop\users.php**
- Создайте в классах **User** и **SuperUser** статические свойства для подсчета количества созданных объектов
- Присвойте этим свойствам начальное значение **0**
- В конструкторах классов инкрементируйте значения данных свойств
- В нижней части кода, после создания экземпляров классов, выведите в браузер количество тех и других объектов примерно так:
  `Всего обычных пользователей:`
  `[количество_экземпляров_класса_User]`
  `Всего супер-пользователей:`
  `[количество_экземпляров_класса_SuperUser]`
- Сохраните файл **oop\users.php**

Упражнение 3: Проверка работы скрипта

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/oop/users.php**
- Проверьте работу скрипта. Если есть ошибки, найдите их и исправьте

Лабораторная работа 1.6

Использование автозагрузки классов

Упражнение 1: Создание классов в отдельных файлах

- В текстовом редакторе откройте файл **oop\users.php**
- В текстовом редакторе создайте новый файл
- Перенесите описание абстрактного класса **UserAbstract** из файла **oop\users.php** в новый файл
- Сохраните новый файл как **oop\classes\UserAbstract.class.php**
- В текстовом редакторе создайте новый файл
- Перенесите описание класса **User** из файла **oop\users.php** в новый файл
- Сохраните новый файл как **oop\classes\User.class.php**
- В текстовом редакторе создайте новый файл
- Перенесите описание интерфейса **ISuperUser** из файла **oop\users.php** в новый файл
- Сохраните новый файл как **oop\classes\ISuperUser.class.php**
- В текстовом редакторе создайте новый файл
- Перенесите описание класса **SuperUser** из файла **oop\users.php** в новый файл
- Сохраните новый файл как **oop\classes\SuperUser.class.php**
- Сохраните файл **oop\users.php**

Упражнение 2: Использование автозагрузки файлов

- В файле **oop\users.php** (основной код) создайте и опишите функцию **__autoload()**, которая производит автозагрузку нужного класса при создании его экземпляра
- Сохраните файл **oop\users.php**
- Запустите браузер
- Наберите в адресной строке браузера

http://mysite.local/oop/users.php

- Проверьте работу скрипта. Вы не должны увидеть никаких изменений. Если есть ошибки, найдите их и исправьте

Модуль 2

PHP. Уровень 3
Практическое использование ООП с базой данных SQLite

Темы модуля

- Обзор базы данных SQLite
- Преимущества и ограничения SQLite перед другими базами данных
- Особенности SQLite
- Выполнение основных операций с базой данных

Что такое SQLite?

- Библиотека, написанная на языке C
- Осуществляет механизм работы с данными с помощью SQL
- http://sqlite.org

Преимущества и ограничения

- Преимущества
  - Полностью бесплатна
  - Нет необходимости в средствах администрирования
  - Высокая производительность и легкая переносимость
  - Поддержка процедурного и объектноориентированного интерфейсов
  - Хранение больших объемов данных
  - Хранение строк и бинарных данных неограниченной длины

- Ограничения
  - Предназначена для небольших и средних приложений
  - Основной выигрыш в производительности, если преобладают операции вставки и выборки данных
  - При чрезвычайно активном обращении к данным, или в случае частых сортировок, SQLite работает медленнее своих конкурентов

Поддержка SQLite в PHP

- В PHP 5.0 поддержка SQLite версии 2 была встроена в ядро
- Начиная с PHP 5.1 поддержка SQLite вынесена за пределы ядра
  - extension=php_sqlite.dll
- В PHP 5.3 добавлена поддержка SQLite версии 3
  - extension=php_sqlite3.dll
- В PHP 5.4 поддержка SQLite версии 2 удалена
  - extension=~~php_sqlite.dll~~

Особенности SQLite

- Можно так
  - CREATE TABLE users(id INTEGER, name TEXT, age INTEGER)
- Можно так
  - CREATE TABLE users(id, name, age)
- Для задания первичного ключа
  - id INTEGER PRIMARY KEY
  - id INTEGER PRIMARY KEY AUTOINCREMENT
- Экранирование строк через двойной апостроф
  - 'Harry O''Brian'

Создание, открытие и закрытие базы данных

```php
// Создаём или открываем базу данных test.db
$db = new SQLite3("test.db");

// Закрываем базу данных без удаления объекта
$db->close();

// Открываем другую базу данных для работы
$db->open("another.db");

// Удаляем объект
unset($db);
```

Структура приложения "Лента новостей"

- news.php
  - основной файл новостной ленты
- INewsDB.class.php
  - интерфейс INewsDB с декларациями методов для новостной ленты
- NewsDB.class.php
  - класс NewsDB реализующий интерфейс INewsDB
- save_news.php
  - php-код обработки данных для добавления записи в таблицу БД
- delete_news.php
  - php-код обработки данных для удаления записи из таблицы БД
- get_news.php
  - вывод списка записей из таблицы БД

Лабораторная работа 2.1

Создание класса веб-приложения

Выполнение запроса

```php
// Экранирование строк
$name = $db->escapeString($name);

// Для запросов без выборки данных
$sql = "INSERT INTO users (name, age)
               VALUES ('$name', 25)";

// Возвращает значение булева типа
$result = $db->exec($sql);

// Количество изменённых записей
echo $db->changes();

// Отслеживание ошибок
echo $db->lastErrorCode();
echo $db->lastErrorMsg();
```

Подготовленные запросы

```
$sql = "INSERT INTO users (name, age)
               VALUES (:name, :age)";

// Готовим запрос
$stmt = $db->prepare($sql);
// Привязываем параметры
$stmt->bindParam(':name', $name);
$stmt->bindParam(':age', $age);
// Исполняем запрос
$result = $stmt->execute();
// Закрываем при необходимости
$stmt->close();
```

Лабораторная работа 2.2

Создание и заполнение базы данных веб-приложения

Лабораторная работа 2.3

Добавление записей в базу данных

Выборка данных

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

// В случае неудачи возвращает false
$result = $db->querySingle($sql);
// В $result - значение первого поля первой записи

$result = $db->querySingle($sql, true);
// В $result - массив значений первой записи

// Стандартная выборка
$result = $db->query($sql);
// Обработка выборки
$row = $result->fetchArray(); // SQLITE3_BOTH

// Получаем ассоциативный массив
$row = $result->fetchArray(SQLITE3_ASSOC);
// Получаем индесированный массив
$row = $result->fetchArray(SQLITE3_NUM);
```

Лабораторная работа 2.4

Выборка записей из базы данных и их показ

Лабораторная работа 2.5

Удаление записей из базы данных

Что мы изучили?

- Познакомились с базой данных SQLite
- Научились базовому использованию расширения PHP SQLITE3
- Создали небольшое веб-приложение, используя при его разработке объектно-ориентировааный подход

Лабораторные работы

- Лабораторная работа 2.1
- Лабораторная работа 2.2
- Лабораторная работа 2.3
- Лабораторная работа 2.4
- Лабораторная работа 2.5

Лабораторная работа 2.1

Создание класса веб-приложения

Упражнение 1: Создание класса

- В текстовом редакторе откройте файл **news\INewsDB.class.php**
- Внимательно ознакомьтесь с его содержимым
- В текстовом редакторе откройте файл **news\NewsDB.class.php**
- Создайте и опишите класс **NewsDB** реализующий интерфейс **INewsDB**
- Создайте константу класса **DB_NAME** и присвойте ей значение **news.db** - имя базы данных. Файл должен создаваться в **корневой** директории сайта!
- Создайте закрытое (private) свойство **$_db** для хранения экземпляра класса **SQLite3**.
- Подумайте, что если данный класс будет наследоваться? Надо обеспечить доступ на чтение значения свойства **$_db** классам-наследникам

Упражнение 2: Создание конструктора и деструктора класса

- Создайте и опишите конструктор класса, в котором выполняется подключение к базе данных **SQLite**
- Присвойте свойству **$_db** значение, которое является экземпляром класса **SQLite3**
- Создайте и опишите деструктор класса, в котором выполняется удаление экземпляра класса **SQLite3**

Упражнение 3: Проверка работы скрипта

- В глобальной области кода (вне класса) создайте временный объект **$news**, экземпляр класса **NewsDB**
- Сохраните файл **news\NewsDB.class.php**
- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/news/NewsDB.class.php**

- Откройте проводник Windows и перейдите в **корневую** директорию сайта mysite.local
  (C:\Пользователи\Общие\OpenServer\domains\mysite.local\ )
- В директории вы должны увидеть созданный пустой файл **news.db**
- Если файла нет или код выдает ошибки, найдите их и исправьте
- Удалите файл **news.db**

Лабораторная работа 2.2

Создание и заполнение базы данных веб-приложения

Упражнение 1: Изменение конструктора класса

- В текстовом редакторе откройте файл **news\news.txt**
- Ознакомьтесь со структурой базы данных **news**
- В текстовом редакторе откройте файл **news\NewsDB.class.php**
- Измените конструктор класса так, чтобы в нём выполнялась проверка, существует ли база данных на следующих условиях:
    - Если базы данных не существует, создайте ее и выполните SQL запросы для добавления таблиц. Готовые запросы находятся в файле **news\news.txt**
    - В противном случае, просто откройте существующую базу данных для работы
- Сохраните файл **news\NewsDB.class.php**

Упражнение 2: Проверка работы скрипта

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/news/NewsDB.class.php**
- В **корневой** директории вы должны увидеть созданный **заполненный** файл **news.db**
- Если файла нет или он есть, но пустой, или код выдает ошибки, найдите их и исправьте
- В файле **news\NewsDB.class.php** удалите строку создающую объект **$news** и сохраните данный файл

Лабораторная работа 2.3

Добавление записей в базу данных

Упражнение 1: Настройка главного файла

- В текстовом редакторе откройте файл **news\news.php**
- В начале файла подключите файл с описанием класса **NewsDB**
- Создайте объект **$news**, экземпляр класса **NewsDB**
- Создайте переменную **$errMsg** со строковым значением **""** (пустая строка)
- Ниже по коду после строки **<h1>Последние новости</h1>** проверьте, не является ли переменная **$errMsg** пустой строкой? Если **НЕТ**, то выведите значение переменной **$errMsg** в произвольной форме
- В верхней части файла (после объявления переменной **$errMsg**) проверьте, была ли отправлена HTML-форма? Если **ДА**, то подключите файл с кодом для обработки HTML-формы **save_news.inc.php**
- Сохраните файл **news\news.php**

Упражнение 2: Описание метода добавления записи в базу данных

- В текстовом редакторе откройте файл **news\NewsDB.class.php**
- Опишите метод **saveNews()**. Описание параметров и возвращаемого значеня метода можно посмотреть в интерфейсе **INewsDB**
- Создайте переменную **$dt** и присвойте ей значение текущих даты и времени в формате временной метки (timestamp)
- Сформируйте строку запроса на добавление новой записи в таблицу **msgs**, используя переменную **$dt** и переданные параметры
- Выполните запрос
- Возвратите нужное значение
- Сохраните файл **news\NewsDB.class.php**

Упражнение 3: Обработка параметров HTML-формы

- В текстовом редакторе откройте файл **news\save_news.inc.php**

- Проверьте, была ли корректным образом отправлена HTML-форма?
    - Если **НЕТ**, то присвойте переменной **$errMsg** строковое значение "**Заполните все поля формы!**"
    - Если **ДА**, то отфильтруйте полученные данные и вызовите метод **saveNews()**, передав ему необходимые параметры
- С помощью возвращаемого методом значения проверьте, был ли запрос успешным?
    - Если **ДА**, выполните перезапрос страницы **news.php**
    - Если **НЕТ**, то присвойте переменной **$errMsg** строковое значение "**Произошла ошибка при добавлении новости**"
- Сохраните файл **news\save_news.inc.php**

Упражнение 4: Проверка работы скрипта

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/news/news.php**
- Добавьте новость заполнив поля HTML-формы
- Откройте файл базы данных **news.db** в тестовом редакторе и убедитесь, что запись добавлена
- Добавьте еще несколько новостей
- Попробуйте послать форму, оставив некоторые поля пустыми
- Если есть ошибки, найдите их и исправьте

Лабораторная работа 2.4

Выборка записей из базы данных и их показ

Упражнение 1: Описание метода выборки записей из базы данных

- В текстовом редакторе откройте файл **news\NewsDB.class.php**
- Опишите метод **getNews()**. Описание параметров и возвращаемого значеня метода можно посмотреть в интерфейсе **INewsDB**
- Сформируйте строку SQL-запроса на выборку всех данных из таблицы **msgs** в обратном порядке. Обратите внимание, что в запросе необходимо сделать объединение двух таблиц. Запрос должен выглядеть примерно так:

```
SELECT msgs.id as id, title, category.name as category,
       description, source, datetime
  FROM msgs, category
  WHERE category.id = msgs.category
  ORDER BY msgs.id DESC
```

- Выполните запрос
- Возвратите **нужное** значение
- Сохраните файл **news\NewsDB.class.php**

Упражнение 2: Подключение файла обработки данных

- В текстовом редакторе откройте файл **news\news.php**
- Внизу файла перед закрывающим тэгом подключите файл **news \get_news.inc.php** с кодом для обработки полученных записей
- Сохраните файл **news\news.php**

Упражнение 3: Вывод записей

- В текстовом редакторе откройте файл **news\get_news.inc.php**
- Вызовите метод **getNews()**
- С помощью возвращаемого методом значения проверьте, был ли запрос успешным?
    - Если **НЕТ**, то присвойте переменной **$errMsg** строковое значение

"**Произошла ошибка при выводе новостной ленты**"

- Если **ДА**, то получите количество записей и выведите его в браузер в произвольной форме

- Используя цикл, выведите в браузер все новости со ссылкой на конкретную новость в произвольной форме
- После каждого сообщения сформируйте ссылку для удаления этой записи. Информацию об идентификаторе удаляемого сообщения передавайте методом **GET**
- Сохраните файл **news\get_news.inc.php**

Упражнение 4: Проверка работы скрипта

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/news/news.php**
- Убедитесь, что данные выводятся корректно. Если есть ошибки, найдите их и исправьте

Лабораторная работа 2.5

Удаление записей из базы данных

Упражнение 1: Описание метода удаления записи из базы данных

- В текстовом редакторе откройте файл **news\NewsDB.class.php**
- Опишите метод **deleteNews()**. Описание параметров и возвращаемого значеня метода можно посмотреть в интерфейсе **INewsDB**
- Сформируйте строку SQL-запроса на удаление записи
- Выполните запрос и возвратите нужное значение
- Сохраните файл **news\NewsDB.class.php**

Упражнение 2: Подключение файла обработки данных для удаления

- В текстовом редакторе откройте файл **news\news.php**
- Вверху файла перед началом HTML-кода подключите файл **news \delete_news.inc.php** с кодом для обработки данных для удаления записи
- Перед подключением убедитесь в наличие параметра, который указывает на удаление записи
- Сохраните файл **news\news.php**

Упражнение 3: Удаление записи

- В текстовом редакторе откройте файл **news\delete_news.inc.php**
- Примите и отфильтруйте полученные данные
- Проверьте, корректны ли полученные данные?
    - Если **НЕТ**, то просто выполните перезапрос страницы
    - Если **ДА**, то вызовите метод **deleteNews()**
- С помощью возвращаемого методом значения проверьте, был ли запрос успешным?
    - Если **НЕТ**, то присвойте переменной **$errMsg** строковое значение "**Произошла ошибка при удалении новости**"

    - Если **ДА**, то выполните перезапрос страницы, чтобы избавиться от информации, переданной через адресную строку

- Сохраните файл **news\delete_news.inc.php**

Упражнение 4: Проверка работы скрипта

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/news/news.php**
- Попробуйте удалить одну или несколько записей
- Убедитесь, что записи удаляются корректно
- Попробуйте удалить несуществующую запись
- Попробуйте передать GET-параметром произвольную строку
- Убедитесь, что показывается информация об ошибках
- Если есть ошибки, найдите их и исправьте

Модуль 3

PHP. Уровень 3
PHP и XML

Темы модуля

- Введение в XML-технологии
- XML-технологий в PHP
  - SAX
  - DOM
  - SimpleXML
  - XMLReader и XMLWriter
- Обзор XSL/T
- Преобразование данных на сервере

Введение в XML

- XML (Extensible Markup Language)
  - Расширяемый язык разметки
- Предназначен для:
  - хранения структурированных данных
  - обмена информацией между программами
  - создания на его основе других, более специализированных, языков разметки (OFX, OTP, WSDL, SOAP, VML, XSL, ebXML, CML, XLANG)
- Цель создания:
  - обеспечение совместимости при передаче структурированных данных между разными системами обработки информации

- Различия XML и HTML
  - HTML описывает ИЗ ЧЕГО СОСТОИТ и КАК отображать документ
  - XML определяет ЗНАЧЕНИЕ и ОТНОШЕНИЕ данных

Правила XML

- Если документ содержит символы, выходящие за рамки ASCII, необходимо указать кодировку
- XML-документ состоит из вложенных элементов
- Элемент состоит из открывающего и закрывающего тегов, а также содержимого
- XML чувствителен к регистру символов
- Элементы могут вкладывться друг в друга
- Теги должны быть правильно вложены друг в друга
- Все парные теги должны быть закрыты
- Возможно формирование пустых элементов
- Должен существовать только один корневой элемент, который содержит все остальные элементы. Пустой документ (без корневого элемента) недопустим!
- Элементы могут иметь атрибуты
- Значения атрибутов заключаются в одинарные или двойные кавычки
- У каждого конкретного элемента не должно быть повторяющихся атрибутов

Корректность и валидность XML-документов

- Корректные XML-документы (well-formed)
  - Документы, полностью соответствующие правилам оформления XML
  - Корректность проверяется XML-парсер
- Валидные XML-документы (valid)
  - Корректные XML-документы, которые соответствуют заранее определенному набору правил
  - Валидность проверяется валидатором
- Описание структуры документа
  - DTD – Document Type Definition
  - XML Схемы

Средства PHP для работы с XML-документом

- SAX (Simple API for XML)
  - чтение XML-документа
- DOM (Document Object Model)
  - чтение, модификация и создание новых XML-документов
- SimpleXML
  - чтение и модификация XML-документов
- XMLReader и XMLWriter
  - чтение и модификация XML-документов

- XSL/T (Extensible StylesheetLanguage Transformations)
  - преобразование XML-документов в другие форматы

Simple API for XML (SAX)

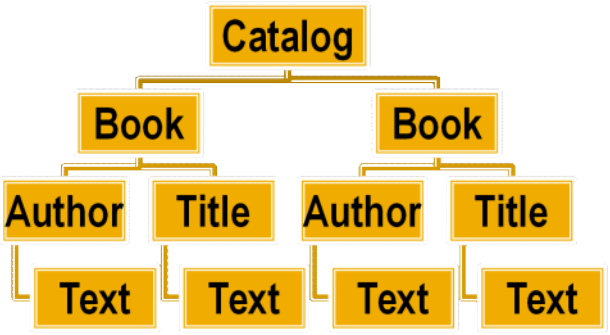
- Официальный сайт: http://www.saxproject.org/
- Описывает метод парсинга XML-документов для получения данных из них
- Создавать и изменять XML-документы с помощью SAX невозможно
- Основан на событиях
- XML-парсеру предоставляется набор собственных функций для обработки различных типов XML-данных
- Парсер автоматически вызывает эти функции в процессе последовательной обработки XML-документа

```
// Использование SAX

// Создание парсера
$sax = xml_parser_create("utf-8");

// Декларация функций обработки событий
function onStart($parser, $tag, $attributes){}
function onEnd($parser, $tag){}
function onText($parser, $text){}

// Регистрация функций как обработчиков событий
xml_set_element_handler($sax, "onStart", "onEnd");
xml_set_character_data_handler($sax, "onText");

// Запуск парсера
xml_parse($sax, "XML строка!");

// Обработка ошибок
echo xml_error_string( xml_get_error_code($sax) );
```

Document Object Model (DOM)

- Интерфейс, позволяющий программам управлять содержимым документов XML, а также изменять их структуру
- Существует спецификация DOM (W3C)
- Представляет XML-документ в виде дерева узлов

```
<catalog>
   <book>
       <author>Хьюз</author>
       <title>PHP</title>
   </book>

   <book>
       <author>Григин</author>
       <title>Справочник</title>
   </book>
</catalog>
```

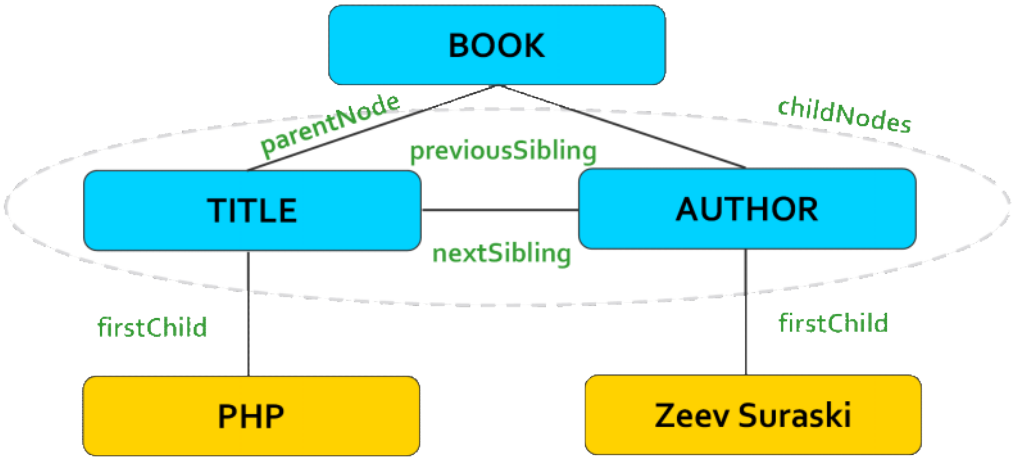


- Типы узлов документа

Код типа	Тип узла	Описание	Пример
1	ELEMENT	Элемент	<book>...</book>
2	ATTRIBUTE	Атрибут элемента	lang="ru"
3	TEXT	Текстовый узел	Это текст
8	COMMENT	Комментарий	<!– Комментарий -->
10	DocumentType	Декларация типа документа	<!DOCTYPE html >

- Связи между узлами

```
<book>
  <title>PHP</title>
  <author>Zeev Suraski</author>
</book>
```



Использование DOM

```php
// Чтение XML-документа

// Создание объекта, экземпляра класса DomDocument
$dom = new DomDocument();

// Загрузка документа
$dom->load("catalog.xml");

// Получение коневого элемента
$root = $dom->documentElement;

// Получение типа узла
echo $root->nodeType; // 1

// Получение коллекции дочерних узлов (экземпляр класса DomNodeList)
$children = $root->childNodes;

// Получение текстового содержимого узла
$content = $root->textContent;

// Получение коллекции элементов с определённым именем
$books = $dom->getElementsByTagName("book");


// Создание/изменение XML-документа

// Создание объекта, экземпляра класса DomDocument
$dom = new DomDocument("1.0", "utf-8");

// Получение коневого элемента
$root = $dom->documentElement;

// Создание новых элементов
$book = $dom->createElement("book");
$title = $dom->createElement("title");

// Создание текстового узла
$text = $dom->createTextNode("Название книги");

// Добавление узлов к узлам
$title->appendChild($text);
$book->appendChild($title);
$root->appendChild($book);

// Другой вариант создания нового элемента
$author = $dom->createElement("author", "Автор книги");
```

```php
// Добавляем узел к узлу перед другим узлом
$book->insertBefore($author, $title);

// Создаём секцию CDATA
$description = $dom->createElement("description");
$cdata = $dom->createCDATASection("...описание книги...");
$description->appendChild($cdata);
$book->appendChild($description);

// Сохраняем документ
$dom->save("catalog.xml");
```

Лабораторная работа 3.1

Создание RSS с помощью DOM

Использование SimpleXML

```php
// Загружаем документ и преобразуем его в объект
$sxml = simplexml_load_file("catalog.xml");
// Загружаем XML-строку и преобразуем его в объект
$sxml = simplexml_load_string("XML строка");

// Получение текста нужного элемента (название второй книги)
echo $sxml->book[1]->title;

// Получение атрибута элемента
echo $sxml->book[1]->title["lang"];

// Изменение текста нужного элемента (название первой книги)
$sxml->book[0]->title = "Новое название";

// Преобразование объекта в строку
$xml = $sxml->asXML();
// Запись строки в файл
file_put_contents("catalog.xml", $xml);
```

Лабораторная работа 3.2

Чтение RSS с помощью SimpleXML

Обзор XMLReader и XMLWriter

```php
// Использование XMLReader

// Создание объекта
$xml = new XMLReader("catalog.xml");
// Перемещение курсора
$xml->read();
$xml->next();
// Получение свойств элемента
echo $xml->nodeType;
echo $xml->depth;
echo $xml->name;
echo $xml->value;
// Получение объекта DomNode
$domNode = $xml->expand();


// Использование XMLWriter

// Создание объекта
$writer = new XMLWriter();
// Выделение памяти под запись
$writer->openMemory();
// Создавать отступы
$writer->setIndent = true;

// Создание документа и узлов
$writer->startDocument("1.0", "utf-8");
  $writer->startElement("catalog");
    $writer->startElement("book");
      $writer->startElement("title");
        $writer->text("Название книги");
      $writer->endElement();
    $writer->endElement();
  $writer->endElement();
$writer->endDocument();

// Получаем XML-строку
$xml = $writer->outputMemory();
// Запись строки в файл
file_put_contents("catalog.xml", $xml);
```

Преобразование XML с XSL/T

- Extensible Stylesheet Language /Transformations
- Стилевая технология, предназначенная для трансформации XML-документов в другие форматы
- Таблицы стилей XSL создаются по правилам XML-документов
- Таблицы стилей XSL состоят из набора шаблонов

```php
// Загрузка исходного XML-документа
$xml = new DomDocument();
$xml->load("catalog.xml");

// Загрузка таблицы стилей XSL
$xsl = new DomDocument();
$xsl->load("catalog.xsl");

// Создание XSLT процессора
$processor = new XSLTProcessor();

// Загрузка XSL в процессор
$processor->importStylesheet($xsl);
// Выполнение преобразования
echo $processor->transformToXML($xml);
```

Что мы изучили?

- Познакомились с XML технологиями
- Изучили основные средства для работы с XML-документом: DOM и SimpleXML
- Рассмотрели другие средства для работы с XML-документом
- Научились преобразовывать XML-документ в другие форматы

Лабораторные работы

- Лабораторная работа 3.1
- Лабораторная работа 3.2

Лабораторная работа 3.1

Создание RSS с помощью DOM

Упражнение 1: Знакомство со структурой RSS-документа

- В текстовом редакторе откройте файл **news\rss.txt** и ознакомьтесь со структурой RSS-документа
- В текстовом редакторе откройте файл **news\NewsDB.class.php**
- Добавьте константу класса **RSS_NAME** для хранения имени RSS-файла, например, **rss.xml**
- Добавьте константу класса **RSS_TITLE** для хранения заголовка новостной ленты, например, **Последние новости**
- Добавьте константу класса **RSS_LINK** для хранения ссылки на саму новостную ленту - **http://mysite.local/news/news.php**

Упражнение 2: Создание метода для формирования RSS-документа

- Создайте и опишите метод **createRss()**, который будет формироавть RSS-документ
- Создайте объект **$dom**, экземпляр класса DOMDocument
- Напишите следующие строки для правильного форматирования документа:
```
$dom->formatOutput = true;
$dom->preserveWhiteSpace = false;
```
- Создайте корневой элемент **rss** и привяжите его к объекту **$dom**
- Напишите следующие строки для создания атрибута **version** корневого элемента:
```
$version = $dom->createAttribute("version");
$version->value = '2.0';
$rss->appendChild($version);
```
- Создайте элемент **channel** и привяжите его к корневому элементу
- Создайте элементы **title** и **link**, и привяжите их к элементу **channel**. Содержимое элементов находится в константах **RSS_TITLE** и **RSS_LINK**
- Получите данные в виде массива из базы данных и дальнейшие действия производите в цикле

- Создайте новый XML-элемент **item** для очередной новости
- Создайте XML-элементы для всех данных новостной ленты (вместе с текстовыми узлами): **title**, **link**, **description**, **pubDate**, **category**. Не забудьте обернуть текст для элемента **description** секцией **CDATA**
- Привяжите созданные XML-элементы с данными к XML-элементу **item**
- Привяжите XML-элемент **item** к элементу **channel**
- Сохраните файл. Имя файла - константа **RSS_NAME**
- Вызовите метод **createRss()** после добавления новости в методе **saveNews()**
- Сохраните файл **news\NewsDB.class.php**

Упражнение 3: Создание RSS-документа

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/news/news.php**
- Добавьте запись в новостную ленту
- Убедитесь, что в папке **news** появился файл **rss.xml**
- Откройте файл **news\rss.xml** и убедитесь, что данные записаны корректно
- Попробуйте добавить еще несколько записей в новостную ленту
- Если есть ошибки, найдите их и исправьте

Лабораторная работа 3.2

Чтение RSS с помощью SimpleXML

Упражнение 1: Создание файла и кода для чтения RSS-документа

- В текстовом редакторе откройте файл **news\rss_reader.php**
- Пересохраните этот файл как **C:\Users\Public\OpenServer\domains \localhost\rss_reader.php**
- Создайте константу **RSS_URL** для хранения адреса RSS-потока со значением **http://mysite.local/news/rss.xml**
- Создайте константу **FILE_NAME** для хранения RSS-документа на локальном сервере со значением **news.xml**
- Обычно новости обновляются с какой-либо периодичностью. Поэтому нет необходимости каждую минуту (и даже секунду) дёргать файл с удалённого сервера. Лучше закешировать данные у себя на локальном сервере и обновлять их через определённый период. Для этого создайте и опишите кеширующую функцию **download()**, которая закачивает RSS-документ с адреса **RSS_URL** и сохраняет его на локальном сервере под именем **FILE_NAME**
- Проверьте, существует ли файл на на локальном сервере? Если **НЕТ**, то создайте его с помощью функции **download()**
- После заголовка первого уровня **Последние новости** зачитайте с помощью **SimpleXML** RSS-документ:
    - Создайте объект - экземпляр класса SimpleXML и загрузите документ
    - В цикле выведите в произвольной форме новостную ленту
- Осуществите проверку на необходимость загрузки свежего RSS-файла на локальный сервер с помощью функции **download()**
- Сохраните файл **rss_reader.php**

Упражнение 2: Чтение RSS-документа

- Запустите браузер
- Наберите в адресной строке браузера **http://localhost/rss_reader.php**
- Убедитесь, что в лента новостей выводится корректно
- Если есть ошибки, найдите их и исправьте

Модуль 4

PHP. Уровень 3
PHP и XML Web services

Темы модуля

- Введение в XML Web services
- История появления веб-служб
- Использование расширения SOAP
- Использование WSDL
- Использование расширения XML-RPC
- Использование контекста потока

XML Web services

- Программы, доступ к которым осуществляется по протоколу HTTP
- Обмен данными происходит в формате XML
- Независимы от платформы
- Простоты в разработке и отладке
- Используются открытые протоколы и стандарты
- Есть возможность описать услуги, предоставляемые службой и способы обращения к ним

История появления веб-служб

- Remote Procedure Call
  - подход, позволяющий программе вызывать процедуры из другого адресного пространства
- Реализации RPC
  - XML-RPC
    - текстовый протокол на базе HTTP (RFC-3529)
  - SOAP
    - текстовый протокол на базе HTTP (RFC-4227)
  - JSON-RPC
    - текстовый протокол на базе HTTP (RFC-4627)
  - .NET Remoting
    - бинарный протокол на базе TCP, UDP, HTTP
  - DCOM
    - MSRPC Microsoft Remote Procedure Call
  - Java RMI

Simple Object Access Protocol

- Простой протокол доступа к объектам
- Запросы посылаются HTTP методом POST
- Структура SOAP сообщения:
  - Envelope
  - Header
  - Body

- SOAP запрос

```
<soap:Envelope
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getStock xmlns="http://site.ru/ws">
      <num>12345</num>
    </getStock>
  </soap:Body>
</soap:Envelope>
```

- SOAP ответ

```
<soap:Envelope
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getStockDetailsResponse xmlns="http://site.ru/ws">
      <getStockDetailsResult>
        <id>12345</id>
        <productName>Стакан граненый</productName>
        <description>Стакан граненый. 250 мл.</description>
        <price>9.95</price>
      </getStockDetailsResult>
    </getStockDetailsResponse>
  </soap:Body>
</soap:Envelope>
```

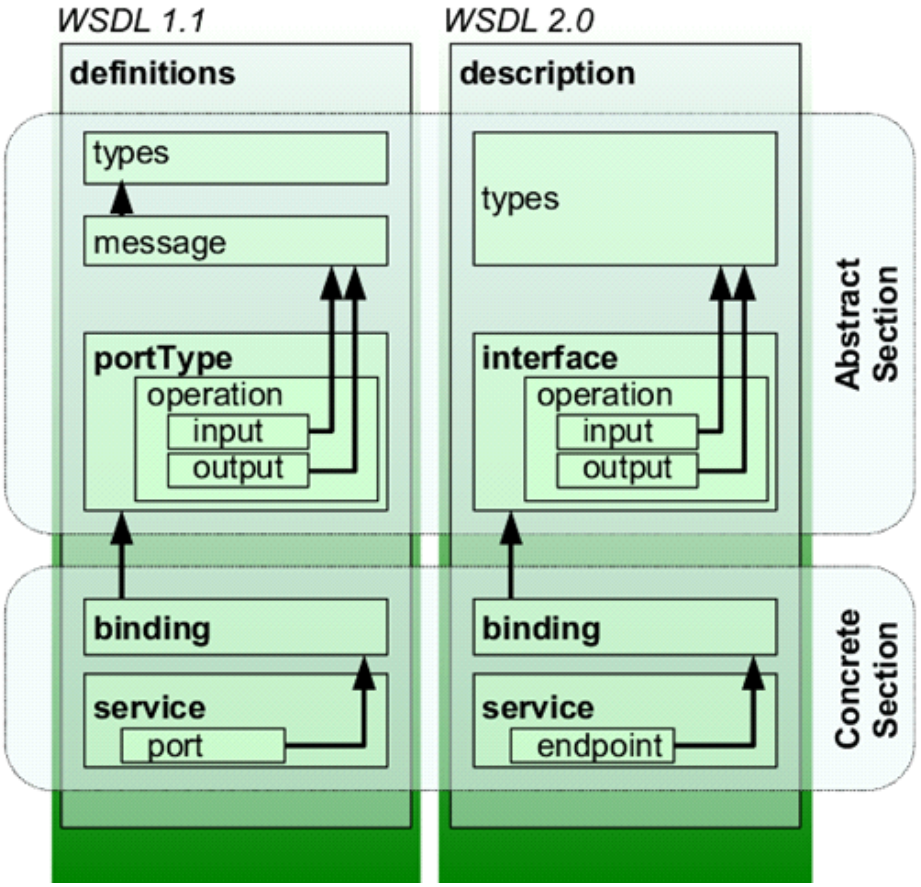
- Для работы необходимо подключить модуль **php_soap.dll**
- Основные SOAP классы:
  - SoapServer
  - SoapClient

Создание SOAP сервера

```php
// Описание службы - процедурный интерфейс
$stock = [
  "a"=>100,
  "b"=>200,
  "c"=>300,
  "d"=>400,
  "e"=>500
];
function getStock($code){
  global $stock;
  if (isset($stock[$code]))
    return $stock[$code];
  return 0;
}

// Создание сервера
$server = new SoapServer("stock.wsdl");
// Добавление функции, которая будет видна клиенту
$server->addFunction("getStock");
// Обработка SOAP-запроса (запуск сервера)
$server->handle();

// Если функций больше, чем одна, то
$funcs = ["getStock", "setStock"];
$server->addFunction($funcs);

// Служба является классом, то
$server->setClass("StockService");
```

Web Services Description Language

- Язык описания веб-служб
- WSDL документ - XML документ
- Описывает:
  - Где находится служба
  - Какие операции (функции) доступны клиенту
  - Количество аргументов функций
  - Типы аргументов функций
  - Типы возвращаемых функциями значений
  - Описание пользовательских типов для параметров функций



Создание SOAP клиента

```php
// Создание сервера
$client = new SoapClient("stock.wsdl");
// Вызов удалённой процедуры
$amount = $server->getStock("b");
echo "Товаров на полке: $amount";

// Посмотреть список доступных операций
print_r( $client->__getFunctions() );
```

Лабораторная работа 4.1

Использование SOAP веб-службы

Использование XML-RPC

- XML-RPC запрос

```
<methodCall>
  <methodName>getStock</methodName>
    <params>
      <param>
        <value><i4>3</i4></value>
      </param>
  </params>
</methodCall>
```

- XML-RPC ответ

```
<methodResponse>
  <params>
    <param>
      <value><i4>300</i4></value>
    </param>
    <param>
      <value><string>Sony Vayo</string></value>
    </param>
  </params>
</methodResponse>
```

- Подключить расширение **php_xmlrpc.dll**

- Создание XML-RPC сервера

```
// Описание службы
$stock = [
  "a"=>100,
  "b"=>200,
  "c"=>300,
  "d"=>400,
  "e"=>500
];

function get_stock($methodName, $arguments, $extra){
```

```
  global $stock;
  $code = $arguments[0];
  if(is_set($stock[$code]))
    return $stock[$code];
  return ["faultCode" => 1, "faultString" => "Нет такой полки"];
}

// Создание сервера
$server = xmlrpc_server_create();
// Добавление функции, которая будет видна клиенту
xmlrpc_server_register_method($server, "getStock", "get_stock");
// Приём запроса
$request = file_get_contents("php://input");
// Обработка запроса
echo xmlrpc_server_call_method($server, $request, null);
```

- Создание XML-RPC клиента

```
// Создание запроса
$server = xmlrpc_encode_request("getStock", "b");
// XML-RPC запрос и получение ответа
$response = запрос_любым_способом_методом_POST("URL");
// Декодирование ответа
$result = xmlrpc_decode("URL");
if( xmlrpc_is_fault($result) )
  echo $result["faultString"];
else
  echo $result;
```

Контекст потока

```php
// Формирование необходимых данных
$options = [
  "http"=> [
    "method" => "GET",
    "header" => "User-Agent: PHPBot\r\n".
                "Cookie: user=John\r\n"
  ]
];

// Создание контекста потока
$context = stream_context_create($options);

// Запрос с использованием созданного контекста потока
echo file_get_contents("http://mysite.local/xml-rpc/server.php", false, $context);


// Получение ответа при использовании fopen()
$f = fopen("http://mysite.local/xml-rpc/server.php", "r", false, $context);
echo stream_get_contents($f);

// Получение заголовков ответа
print_r( stream_get_meta_data($f) );
```

Лабораторная работа 4.2

Использование XML-RPC службы

Что мы изучили?

- Произвели обзор RPC
- Научились создавать SOAP сервер
- Научились использовать WSDL
- Научились создавать SOAP клиента
- Рассмотрели использование XML-RPC

Лабораторные работы

- Лабораторная работа 4.1
- Лабораторная работа 4.2

Лабораторная работа 4.1

Использование SOAP веб-службы

Упражнение 1: Создание SOAP-сервера

- В текстовом редакторе откройте файл **soap\soap-server.php**
- Ознакомьтесь с содержимым службы **NewsService**
- В нижней части файла после описания класса введите следующий текст:

```
// Отключение кеширования wsdl-документа
ini_set("soap.wsdl_cache_enabled", "0");
// Создание SOAP-сервера
$server = new
SoapServer("http://mysite.local/soap/news.wsdl");
// Регистрация класса
$server->setClass("NewsService");
// Запуск сервера $server->handle();
```

- Сохраните файл **soap\soap-server.php**

Упражнение 2: Создание SOAP-клиента

- В текстовом редакторе откройте файл **soap\soap-client.php**
- Пересохраните этот файл как **C:\Users\Public\OpenServer\domains\localhost\soap-client.php**
- В файле введите следующий текст:

```
$client = new
SoapClient("http://mysite.local/soap/news.wsdl");
try{
  // Сколько новостей всего?
  $result = $client->getNewsCount();
  echo "<p>Всего новостей: $result</p>";
  // Сколько новостей в категории Политика?
  $result = $client->getNewsCountByCat(1);
  echo "<p>Всего новостей в категории Политика:
$result</p>";
  // Покажем конкретную новость
  $result = $client->getNewsById(1);
  $news = unserialize(base64_decode($result));
  var_dump($news);
}catch(SoapFault $e){
  echo 'Операция '.$e->faultcode.' вернула ошибку: '.$e->
getMessage();
```

```
}
```

- Сохраните файл **soap-client.php**

Упражнение 3: Тестирование сервиса

- Запустите браузер
- Наберите в адресной строке браузера **http://localhost/soap-client.php**
- Убедитесь, что данные выводятся корректно
- Если есть ошибки, найдите их и исправьте
- Попробуйте допустить намеренную ошибку в файле **soap\soap-server.php**, например, укажите в SQL-запросе несуществующую таблицу. Какие данные при этом будут выведены в браузер?

Лабораторная работа 4.2

Использование XML-RPC службы

Упражнение 1: Создание XML-RPC сервера

- В текстовом редакторе откройте файл **xml-rpc\xml-rpc-server.php**
- Ознакомьтесь с содержимым службы **NewsService**
- В нижней части файла после описания класса введите следующий текст:

```
/* Читаем запрос */
$request = file_get_contents("php://input");
/* Создаем XML-RPC сервер */
$server = xmlrpc_server_create();
/* Регистрируем метод класса */
xmlrpc_server_register_method($server, "getNewsById", [new
NewsService, "xmlRpcGetNewsById"]);
/*Отдаем правильный заголовок*/
header('Content-Type: text/xml;charset=utf-8');
/* Отдаем результат */
print xmlrpc_server_call_method($server, $request, null);
```

- Сохраните файл **xml-rpc\xml-rpc-server.php**

Упражнение 2: Создание XML-RPC-клиента

- В текстовом редакторе откройте файл **xml-rpc\xml-rpc-client.php**
- Пересохраните этот файл как **C:\Users\Public\OpenServer\domains \localhost\xml-rpc-client.php**
- В файле введите следующий текст:

```
header('Content-Type: text/html;charset=utf-8');
/* Сюда приходят данные с сервера */
$output = [];
/* Основная функция */
function make_request($xml, &$output) {
  /* НАЧАЛО ЗАПРОСА */
  $options = [
    'http'=>[
      'method' => "POST",
      'header' => "User-Agent: PHPRPC/1.0\r\n" .
                  "Content-Type: text/xml\r\n" .
                  "Content-length: " . strlen($xml) . "\r
\n",
      'content' => "$xml"
```

```
    ]
  ];
  $context = stream_context_create($options);
  $retval = file_get_contents('http://mysite.local/xml-
rpc/xml-rpc-server.php', false, $context);
  /* КОНЕЦ ЗАПРОСА */
  $data = xmlrpc_decode($retval);
  if (is_array($data) && xmlrpc_is_fault($data)){
    $output = $data;
  }else{
    $output = unserialize(base64_decode($data));
  }
}

/* Идентификатор статьи */
$id = 1;
$request_xml = xmlrpc_encode_request('getNewsById',
array($id));
make_request($request_xml, $output);
/* Вывод результата */
var_dump($output);
```

- Сохраните файл **xml-rpc-client.php**

Упражнение 3: Тестирование сервиса

- Запустите браузер и наберите в адресной строке браузера **http://localhost/xml-rpc-client.php**
- Убедитесь, что данные выводятся корректно. Если есть ошибки, найдите их и исправьте
- Попробуйте передать неверный идентификатор новости
- Попробуйте передать неправильное количество параметров
- Попробуйте допустить намеренную ошибку в файле **xml-rpc\xml-rpc-server.php**, например, укажите в SQL-запросе несуществующую таблицу

Модуль 5

PHP. Уровень 3
Сокеты и сетевые службы

Темы модуля

- Соединение с удалёнными узлами с использованием сокетов
- Сетевые функции

Использование сокетов

- Позволяют осуществить доступ к используемым сетевым протоколам
- Варианты использования:
  - Подключение к службе, для которой отсутствует соответствующая обёртка (file wrapper)
  - Осуществление действий, невозможных при использовании потоков, но возможных при использовании сетевых протоколов

```
// Создание сокета
$socket = fsockopen("mysite.local", $80, $errno, $errmsg, 30);

if(!$socket){
  echo "$errno : $errmsg";
}else{
  // Поготовка запроса
  $output = "HEAD /server.php HTTP/1.1\r\n";
  $output .= "Host: mysite.local\r\n";
  $output .= "Connection: close\r\n\r\n";

  // Посылаем запрос
  fwrite($socket, $output);

  // Читаем ответ
  while( !$feof($socket) ){
    echo $fgets($socket);
  }
  // Закрываем сокет
  $fclose($socket);
}
```

Сетевые функции

```php
// Получаем имя хоста по ip-адресу
$host_name = getHostByAddr("127.0.0.0");

// Получаем ip-адрес по имени хоста
$ip_address = getHostByName("mysite.local");
// Получаем массив ip-адресов по имени хоста
$ip_addresses = getHostByNameL("mysite.local");

// Получаем номер порта по имени службы
$port = getServByName("http", "tcp");

// Получаем имя службы по номеру порта
$service = getServByPort(80, "tcp");

// Получаем DNS запись для указанного хоста
$dns_record = dns_get_record("mysite.local");
// Получаем MX запись для указанного хоста
$dns_record = getmxrr("mysite.local");
// Проверяем имя хоста на существование
$exists = checkdnsrr("mysite.local");
```

Что мы изучили?

- Научились использовать сокеты
- Познакомились с сетевыми функциями

Модуль 6

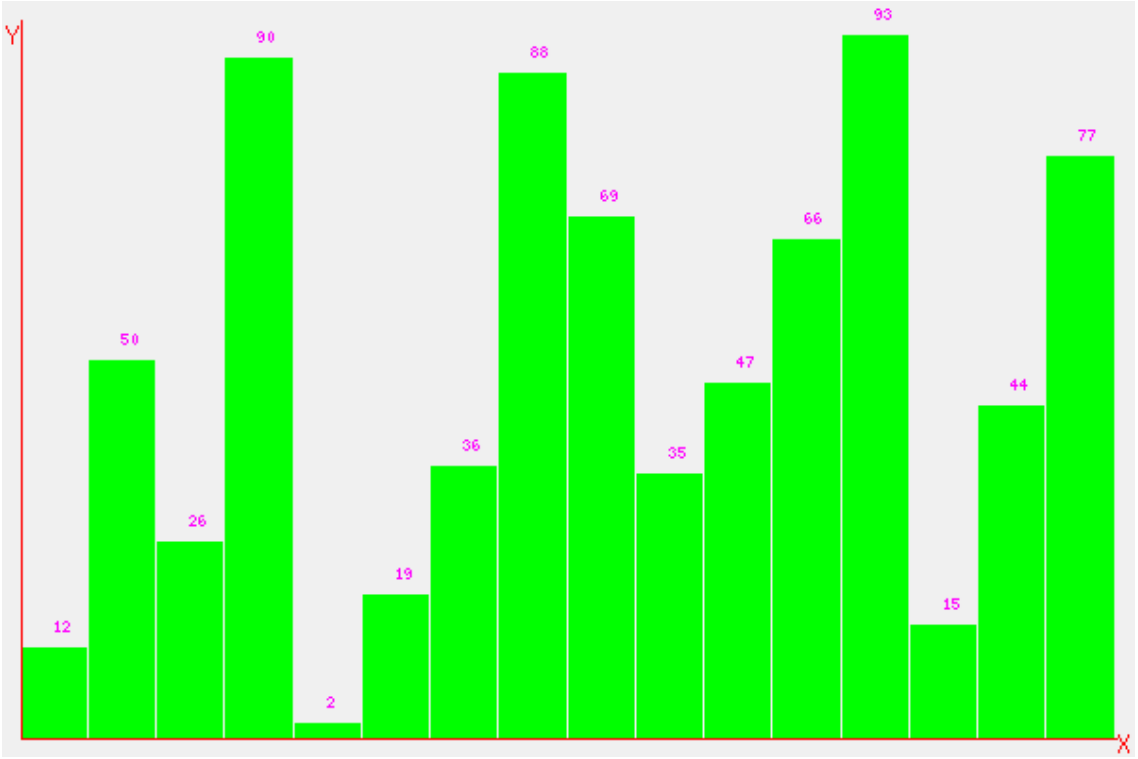
PHP. Уровень 3
Работа с графикой

Темы модуля

- Введение в графические форматы
- Вопросы генерации графики на PHP
- Использование расширения GD2
- Базовые функции для работы с графикой

Работа с изображениями в PHP

- В PHP можно генерировать изображения «на лету»
- Или изменять уже существующие изображения



- Это можно сделать с помощью библиотеки GD2
  - http://www.boutell.com/gd/

- Необходимо подключить расширение **php_gd2.dll**
- Поддерживаемые форматы:
  - GIF (< 1.6 и > 2.0.28)
  - JPEG (все версии)
  - PNG (> 1.6)

- Изображения можно сохранять на сервере
- Изображения можно напрямую отдавать

клиенту:

- <img src="create_image.php">

Базовые операции при работе с изображениями

```php
// Создание изображения (256 цветов)
$img = imageCreate(500, 300);
// Создание полноцветного изображения
$img = imageCreateTrueColor(500, 300);

// Генерация изображения в формате GIF
  // Копируем на выход
  imageGif($img);
  // Сохраняем на диск
  imageGif($img, "logo.gif");

// Генерация изображения в формате PNG
  // Копируем на выход
  imagePng($img);
  // Сохраняем на диск
  imagePng($img, "logo.png");

// Генерация изображения в формате JPEG
  // Копируем на выход
  imageJpeg($img);
  imageJpeg($img, null, 75);
  // Сохраняем на диск
  imageJpeg($img, "logo.jpg", 75);

// Включение сглаживания
  imageAntiAlias($img, true);

// Выбор цвета
$color = imageColorAllocate($img, 255, 0, 0);

// Выбор прозрачного цвета (для формата GIF)
imageColorTransparent($img, $color);

// Заливка фона изображения
imageFill($img, 0, 0, $color);

// Отрисовка пикселя
imageSetPixel($img, 10, 10, $color);

// Отрисовка линии
imageLine($img, 20, 20, 80, 280, $color);

// Отрисовка прямоугольника
imageRectangle($img, 20, 20, 80, 280, $color);
// Отрисовка залитого прямоугольника
```

```php
imageFilledRectangle($img, 20, 20, 80, 280, $color);

// Отрисовка многоугольника
$points = [0, 0, 100, 200, 300, 200];
imagePolygon($img, $points, 3, $color);
// Отрисовка залитого многоугольника
imageFilledPolygon($img, $points, 3, $color);

// Отрисовка эллипса
imageEllipse($img, 200, 150, 300, 200, $color);
// Отрисовка залитого эллипса
imageFilledEllipse($img, 200, 150, 300, 200, $color);

// Отрисовка дуги
imageArc($img, 200, 150, 300, 200, 0, 40, $color);
// Отрисовка сектора
imageFilledArc($img, 200, 150, 300, 200, 0, 40, $color, IMG_ARC_PIE);
imageFilledArc($img, 200, 150, 300, 200, 0, 40, $color, IMG_ARC_CHORD);
imageFilledArc($img, 200, 150, 300, 200, 0, 40, $color,
                                        IMG_ARC_EDGED | IMG_ARC_NOFILL);


// Отрисовка текста
imageString($img, 3, 150, 200, "Текст", $color);

// Отрисовка первого символа текста
imageChar($img, 3, 150, 200, "Текст", $color);
// Отрисовка первого символа текста лежащего на левом боку
imageCharUp($img, 3, 150, 200, "Текст", $color);

// Продвинутая отрисовка текста
imageTtfText($img, 30, 10, 300, 150, $color, "Arial.ttf", "Текст");


// Использование существующего изображения
$img = imageCreateFromGif("picture.gif");
$img = imageCreateFromPng("picture.png");
$img = imageCreateFromJpeg("picture.jpg");
$img = imageCreateFromString($string);


// Установка толщины линии
imageSetThickness($img, 5);

// Использование стилей
$style = [$red, $red, $red, $black, $black, $black];
imageSetStyle($img, $style);
imageLine($img, 20, 20, 80, 280, IMG_COLOR_STYLED);
```

Лабораторная работа 6

Создание и использование CAPTCHA

Что мы изучили?

- Познакомились с базовыми функциями для работы с изображениями расширения GD2

Лабораторные работы

- Лабораторная работа 6

Лабораторная работа 6

Создание и использование CAPTCHA

Уяснение задачи

- При загрузке файла **gd\registration.php** пользователю показывается изображение, которое динамически формируется скриптом **gd\noise-picture.php**
- Пользователь вводит в текстовое поле веб-формы строку изображённую на картинке и отправляет скрипту **gd\registration.php**
- В файле **gd\registration.php** введённые пользователем данные проверяются на совпадение и выдаётся соответствующий ответ
- Необходимо реализовать создание картинки со случайной строкой и проверку введённых пользователем данных
- Хранение случайной строки, сформированной для изображения с целью проверки на соответствие введённых пользователем данных, реализовать через механизм пользовательских сессий

Упражнение 1: Создание изображения

- В текстовом редакторе откройте файл **gd\noise-picture.php**
- Создайте изображение на основе файла **images/noise.jpg**
- Создайте цвет для рисования
- Включите сглаживание. Оно ухудшает распознавание текста на картинке программным способом
- Рекомендуемые значения значения для отрисовки текста на изображении **noise.jpg**:
    - На изображении помещается около **5**-**6** символов размером от **18** до **30** пт. с расстоянием **40** пт. между символами
    - Начальные координаты для отрисовки строки по осям **X** и **Y** где-то **20** и **30** соответсвенно
- Сгенерируйте случайную уникальную строку любым способом, который придёт вам в голову
- Используя цикл отрисуйте строку посимвольно используя шрифты из папки **fonts**
- Для каждого символа задайте случайные значение размера и угла наклона. Можно поиграться шрифтами и цветами
- Отдайте полученный результат как **jpeg**-изображение с **50%** сжатием
- Не забудьте сохранить сгенерированную строку в сессии

- Сохраните файл **gd\noise-picture.php**
- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/gd/noise-picture.php**
- Убедитесь, что изображение с символами выводится в браузер
- Если есть ошибки, найдите их и исправьте

Упражнение 2: Использование созданного изображения

- В тестовом редакторе откройте файл **gd\registration.php**
- Проверьте, была ли отправлена форма
- Проверьте, не отключен ли показ картинок в браузере пользователя. Если отключен, сообщите ему об этом
- Проверьте введённые пользователем данные на соответствие с текстом на изображении и сообщите ему о результате
- Сообщения для пользователя выводите внизу файла после HTML-формы
- Сохраните файл **gd\registration.php**

Упражнение 3: Тестирование CAPTCHA

- Запустите браузер
- Наберите в адресной строке браузера **http://mysite.local/gd/registration.php**
- Введите в тестовое поле символы, изображенные на картинке и отправьте данные на сервер. После перезагрузки страницы вы должны увидеть положительный ответ
- Введите в тестовое поле произволные символы и отправьте данные на сервер. После перезагрузки страницы вы должны увидеть отрицательный ответ
- Попробуйте отключить показ картинок в браузере и перезапустите его. Что произойдет после перезагрузки страницы?

Что почитать?

- Документация PHP

Что дальше?

- PHP. Уровень 4. Расширенные возможности PHP