

Лабораторные работы по курсу РНР: Проектирование и разработка сложных веб – проектов

Лабораторная работа 2.5

Пространства имен (Namespace) – механизм группировки связанных констант, функций и классов, логически связанных между собой.

1. Откройте файл `/labs/namespace/lab.php`
2. Распечатайте текущее пространство имён
`echo __NAMESPACE__;`
3. Убедитесь, что в браузере ничего не вывелось
4. Над строкой с оператором `echo` впишите пространство имён `Specialist:`
`namespace Specialist`
5. Убедитесь, что в браузере отобразилось слово «Specialist». Закомментируйте вывод.
6. Внимательно изучите код

```
const SPEC_URL = "http://specialist.ru";
function spec_func() { return __FUNCTION__; }
class Specialist {
    static function getClass(){
        return __CLASS__;
    }
}
```

7. Вставьте код в lab.php, сохраните файл
8. Изучите результат работы. Должен получиться результат:
«http://specialist.ru
http://specialist.ru
Specialist\spec_func
Specialist\Specialist»
9. Создайте в файле lab.php функцию substr():

```
function substr(){
  echo " <p> test";
}
substr();
```

10. Обратите внимание! Несмотря на то, что `substr()` – существующая функция PHP в нашем пространстве имён `Specialist`, она является самостоятельной функцией.
11. Добавьте перед вызовом функции символ обратного слеша «`\`».
12. Посмотрите результат в браузере. Функция будет требовать два параметра. Удалите слеш перед функцией.

Лабораторная работа 2.6

1. В нижней части скрипта в файле lab.php создайте экземпляр класса User (в текущем файле этот класс не описан, нам нужно просто попытаться создать экземпляр).

- ```
$user = new User();
echo "<p>",$user->name;
```
2. Должна показаться ошибка: «**Fatal error:** Class 'Specialist\User' not found in...». То есть в **текущем** пространстве имён нет такого класса.
  3. Нам нужно подключить класс User из папки vendor. Папка с таким названием обычно содержит сторонние библиотеки/классы, которые мы можем загружать.  
Добавьте обратный слеш перед названием класса при создании объекта:  

```
$user = new \User();
```
  4. Должна показаться ошибка: «**Fatal error:** Class 'User' not found in...». То есть в **глобальном** пространстве имён нет такого класса.
  5. В верхней части файла lab.php подключите autoload.php:  

```
require __DIR__ . '/vendor/autoload.php';
```
  6. Создайте поочередно экземпляры класс Foo  

```
$foo = new \Some\Foo\Foo();
```

 //а потом просто 

```
new \Some\Foo ()
```

```
$foo->getInfo();
```
  7. Сравните работу в случае с 

```
\Some\Foo\Foo()
```

 и 

```
\Some\Foo()
```

. Найдите в папке vendor файлы, соответствующие вызываемым в коде.
  8. Примечание: во многих фреймворках такая организация загрузки классов упрощает построение веб-приложения

## Лабораторная работа 2.7 (для самостоятельной работы)

1. Изучите способы загрузки файлов с классами в Laravel, Yii и Zend Framework 2

## Лабораторная работа 3.1

Работа со слоем PDO-абстракции: вставка и выборка данных из базы

Упражнение 1. Подготовительная работа по созданию структуры базы/таблицы

1. Напишите SQL-запрос на создание базы данных **products** в базе данных MySQL.
2. Напишите SQL-запрос на создание базы данных **products.sql3** в базе данных MySQL.
3. Напишите и выполните запросы на создание в базах таблицы **product** со структурой, содержащей поля **id**, **name**, **price**

Упражнение 2. Работа с базой данных на сервере MySQL

1. Откройте папку сайта **/labs/pdo**
2. Откройте файл **pdo.php**
3. Напишите SQL-запрос на добавление записи в базу **products**
4. Внесите несколько строк данных
5. Напишите запрос на выборку из базы, используя подготовленные запросы
6. Сохраните файл **/labs/pdo/pdo.php**
7. Выведите данные из базы
8. Убедитесь в корректности вывода товаров из базы MySQL

Упражнение 3. Работа с базой данных на сервере SQLite

1. Внесите изменения в файл **/labs/pdo/pdo.php**, чтобы данные выбирались из **products.sql3**
2. Сохраните файл **/labs/pdo/pdo.php**
3. Выведите данные из базы
4. Убедитесь в корректности вывода товаров из базы SQLite
5. Исправьте ошибки

## Лабораторная работа 3.1

Работа со слоем PDO-абстракции: транзакции в PDO

Упражнение 1. Создание транзакции

1. Откройте файл **/labs/pdo/pdo.php**
2. Измените фрагмент кода, отвечающего за вставку данных: вместо вставки нескольких полей отдельными командами, оберните эти команды в транзакцию
3. Выведите данные из базы
4. Убедитесь в корректности вывода товаров из базы SQLite
5. Исправьте ошибки

## Лабораторная работа 4.1

Построение формы с использованием интерфейса Reflection API

Упражнение 1. Изучение класса для использования Reflection API

1. Откройте файл **labs/reflection/ProductForm.php**
2. Познакомьтесь со структурой класса (свойствами, модификаторами, PHPDoc)

Упражнение 2. Получение информации о классе при помощи ReflectionClass

1. Откройте файл **labs/reflection/form.php**
2. Получите информацию о классе **ProductForm** при помощи **ReflectionClass**
  - a. Создайте экземпляр **ReflectionClass** с аргументом **ProductForm**
  - b. Выведите информацию о классе при помощи **Reflection::export()**
  - c. Проверьте вывод в браузере
  - d. Познакомьтесь с описанием класса
  - e. Закомментируйте вывод информации о классе
3. Получите информацию о всех свойствах класса **ProductForm** в переменную **\$properties**
4. Выполните следующие действия в цикле
  - a. Возьмите очередное свойство и прочитайте для него PHPDoc
  - b. Из PHPDoc извлеките информацию о типе элемента, его имени
  - c. Создайте строку с HTML-элементом на основе полученной информации
  - d. Выведите HTML-элемент
5. Сохраните файл **labs/reflection/form.php**

Упражнение 3. Проверка работы

1. Откройте в браузере файл **labs/reflection/form.php**
2. Убедитесь, что форма отображается корректно
3. В случае появления ошибок, исправьте их

## Лабораторная работа 4.2

Получение информации о классах на основе Reflection API

Упражнение 1. Получение списка классов

1. Откройте папку **labs/patterns**
2. Создайте файл **info.php** в этой папке и откройте его на редактирование
3. Пройдитесь по всем классам в этой папке, используя итераторы
4. На каждом шаге выводите краткую информацию по каждому классу (DocComment, название класса, кол-во методов и их список с краткими характеристиками)
5. Сохраните файл **info.php**

Упражнение 2. Проверка работы

1. Запустите в браузере **labs/patterns/info.php**
2. Убедитесь, что информация о классах выводится корректно
3. Исправьте существующие ошибки (если они появились)

## Лабораторная работа 4.3 (для домашней самостоятельной работы)

Построение формы с использованием интерфейса Reflection API и аннотаций

### Упражнение 1. Знакомство с аннотациями

1. Во фреймворке Zend Framework 2 есть компонент Zend\Form\Annotation, прочитайте информацию о нем <https://framework.zend.com/manual/2.4/en/modules/zend.form.quick-start.html#using-annotations>
2. Компонент зависит от библиотеки Doctrine <http://docs.doctrine-project.org/projects/doctrine-common/en/latest/reference/annotations.html#introduction>

### Упражнение 2. Использование аннотаций при создании форм

1. Создайте класс **User**:

```
use Zend\Form\Annotation;

/**
 * @Annotation\Name("user")
 * @Annotation\Hydrator("Zend\Stdlib\Hydrator\ObjectProperty")
 */
class User
{
 /**
 * @Annotation\Exclude()
 */
 public $id;

 /**
 * @Annotation\Filter({"name":"StringTrim"})
 * @Annotation\Validator({"name":"StringLength", "options":{"min":1,
"max":25}})
 * @Annotation\Validator({"name":"Regex", "options":{"pattern":"/^[a-
zA-Z][a-zA-Z0-9_-]{0,24}$/"})
 * @Annotation\Attributes({"type":"text"})
 * @Annotation\Options({"label":"Username:"})
 */
 public $username;

 /**
 * @Annotation\Type("Zend\Form\Element\Email")
 * @Annotation\Options({"label":"Your email address:"})
 */
 public $email;
}
```

2. Используя **Reflection API** или установленный компонент аннотаций создайте форму на основе класса **User**.

### Упражнение 3. Использование Reflection API для работы с модульной системой

1. Реализуйте вариант модульной системы с использованием Reflection API, который описан в книге Мэтта Зандстры.