

Раздел 1:

Знакомство с ООП



План лайва

- В чем задача лайва
- Анатомия класса
- Конструктор и методы класса
- Пространства имен
- Автозагрузка
- Обзор домашнего задания



Задача лайва

На лайвах мы не рассматриваем теорию.

Лайвы нужны для демонстрации практической работы,
а также объяснения тонких моментов, которым не нашлось
места в учебнике.



Из чего состоит класс

Класс — это набор из свойств и методов, которые затем будут доступны в созданном объекте

Исключение составляют статические свойства/методы: они доступны напрямую из класса

```
class FormValidator
{
    public $formData = [];
    public $requiredFields = [];

    private $errors = [];

    public function getErrors()
    {
        return $this->errors;
    }
}
```



Свойства и модификаторы доступа

Инкапсуляция в ООП — это сокрытие внутренней реализации класса

Для реализации инкапсуляции в PHP есть поддержка модификаторов доступа:

- публичные
- приватные
- защищенные

.....

```
public function validate()  
{  
    $fields = array_merge($this->requiredFields, array_keys($this->formData));  
    $errors = [];
```

...

```
    return empty($this->errors);  
}
```

```
private function validateFilled($name) {  
    if (empty($this->formData[$name])) {  
        return "Это поле должно быть заполнено";  
    }
```

```
    return null;  
}
```

.....



Конструктор класса

Конструктор класса – это метод с именем `__construct`.

Конструктор автоматически вызывается при создании объекта.

Конструктор нужен для подготовки объекта к работе:
например, там можно размещать код для инициализации свойств

```
class FormValidator
{
    const METHOD = 'post';

    public $formData = [];
    public $requiredFields = [];

    private $errors = [];

    public function __construct($formData = [], $requiredFields = [])
    {
        $this->formData = $formData;
        $this->requiredFields = $requiredFields;
    }
}
```



Методы и this

Публичный интерфейс класса — это набор публичных методов, которые доступны вызывающему коду

Внутренняя реализация класса скрыта внутри приватных методов

Псевдопеременная `$this` используется для обращения к **текущему** объекту



Статические методы/свойства класса

Класс может иметь специальные свойства/методы, к которым можно обращаться без создания его экземпляра (объекта)

Такие свойства/методы называются **статическими**

Пример

```
class UserHelper {
```

```
    public static function isLoggedIn() {  
        return isset($_SESSION['user']);  
    }
```

Объявление статического метода

```
}
```

```
UserHelper::isLoggedIn();
```

Метод вызывается через ::
напрямую из класса



Пространство имен

- Пространство имён (неймспейсы) – это способ делить наборы классов по "пакетам"
- Неймспейсы нужны для семантики
- Неймспейсы для работы автозагрузки на основе соглашений

namespace **ha\validators**;


Корневое пространство имен

Вложенное пространство имен
– путь к папке с классом




Автозагрузка

Автозагрузка — это автоматическое подключение файлов классов, если соблюдается соглашение по именованию

 PHP-FIG

HomeBlogPSR sPersonnelBylawsFAQsGet Involved

PSR-4: Autoloader



The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#).

1. Overview

This PSR describes a specification for **autoloading** classes from file paths. It is fully interoperable, and can be used in addition to any other autoloading specification, including [PSR-0](#). This PSR also describes where to place files that will be autoloaded according to the specification.

2. Specification

1. The term "class" refers to classes, interfaces, traits, and other similar structures.
2. A fully qualified class name has the following form:

```
\<NamespaceName>(\<SubNamespaceNames>)*\<ClassName>
```

Additional Info:

- PSR-4: Autoloader
- PSR-4 Meta Document
- PSR-4 Example Implementations



Автозагрузка и Composer

1. Поместить классы в нужные папки и неймспейсы
2. Добавить конфиг composer.json
3. Выполнить `composer dump-autoload`

```
{
    "name": "htmlacademy/source",
    "autoload": {
        "psr-4": {
            "ha\\": "classes/"
        }
    },
    "require": {}
}
```



Обзор домашнего задания

