Введение в пространства имен в ООП на PHP

Если при запуске PHP скрипта будут два класса с одинаковыми именами, то они вступят в конфликт, что приведет к фатальной ошибке. Это на самом деле не очень удобно, так как постоянно приходится следить за уникальностью имен.

Для примера рассмотрим следующую ситуацию: у вас есть сайт, на котором есть пользователи и админ. При этом в папке *users* хранятся классы для юзеров, а в папке *admin* - классы для админа.

Пусть и для юзеров, и для админа нужен некий класс **Page**, отвечающий за какие-то страницы сайта. При этом для юзеров будет свой класс, а для админа - свой. В таком случае нас и поджидает конфликт имен.

Самый простой способ решения этого конфликта - дать отличающиеся имена классам, например, **UsersPage** и **AdminPage**. Этот путь, однако, постепенно ведет к появлению очень длинных имен классов.

В PHP существует и другой путь решения проблемы - пространства имен. Суть в следующем: каждый класс может относится к какому-то пространству имен и при этом уникальность имен классов должна соблюдаться только внутри этого пространства.

То есть, для решения нашей проблемы мы можем сделать следующее: отнести один класс Page к какому-нибудь пространству имен, например, **Users**, а второй класс Page отнести к другому пространству имен, например, **Admin**.

Синтаксис пространств имен

Чтобы задать классу пространство имен, нужно первой строчкой файла, в котом хранится этот класс написать команду **namespace**, а после нее через пробел - название этого пространства.

Если класс относится к какому-нибудь пространству имен, то для создания объекта класса нужно будет указать не только имя класса, но и его пространство имен, разделенные обратным слешем. Давайте посмотрим на примере.

Пусть у нас есть класс Page, не относящийся ни к какому пространству имен. Тогда объект этого класса мы создадим следующим образом:

```php
<?php
  $page = new Page;
?>
```

Пусть теперь этот класс принадлежит пространству имен Admin. В этом случае объект этого класса мы будем создавать уже вот таким образом:

```php
<?php
  $page = new \Admin\Page;
?>
```

Посмотрим на примере

Давайте разнесем классы для юзеров и классы для админа по разным пространствам имен, чтобы избежать описанных выше конфликтов классов.

Для класса Page из файла */admin/page.php* укажем пространство имен Admin:

```php
<?php
  namespace Admin;

  class Page
  {

  }
?>
```

А для класса Page из файла файл */users/page.php* укажем пространство имен Users:

```php
<?php
  namespace Users;

  class Page
  {

  }
?>
```

Давайте теперь в файле */index.php* создадим объект одного и второго класса Page:

```php
<?php
  require_once '/admin/page.php';
  require_once '/users/page.php';

  $adminPage = new \Admin\Page;
  $usersPage = new \Users\Page;
?>
```

Задача 86.1

Пусть у вас есть папка core и папка project. В каждой из папок есть свой класс Controller. Сделайте так, чтобы эти классы принадлежали разным пространствам имен. В файле index.php создайте объекты одного и второго классов.

Подпространства имен

Пусть теперь у нас есть более сложная ситуация: для админа нужно сделать два класса Page - один с данными страницы, а второй - с представлением этих данных. Пусть первый класс Page находится в файле */admin/data/page.php*, а второй - в файле */admin/view/page.php*.

Выше мы уже решили, что все классы из папки admin будут относится к пространству имен Admin. Однако, теперь в этом самом пространстве у нас конфликт двух классов. Для решения проблемы можно сделать дополнительные подпространства имен. Например, можно сделать пространство имен Admin, а в нем подпространства Data и View. В таком случае имена этих подпространств просто записываются через обратный слеш - как при задании пространства имен, так и при создании объекта класса.

Здесь следует уточнить, что уровень вложенности подпространств не ограничен (можно создавать под под пространства в подпространствах и так далее).

Итак, давайте доделаем наш описанный выше пример.

Для класса Page из файла */admin/data/page.php* укажем пространство имен Admin\Data:

```php
<?php
  namespace Admin\Data;

  class Page
  {

  }
?>
```

Для класса Page из файла */admin/view/page.php* укажем пространство имен Admin\Data:

```php
<?php
  namespace Admin\View;

  class Page
  {

  }
?>
```

Создадим объекты наших классов в файле */index.php*:

```php
<?php
  require_once '/admin/data/page.php';
  require_once '/admin/view/page.php';

  $adminDataPage = new \Admin\Data\Page;
  $adminViewPage = new \Admin\View\Page;
?>
```

Задача 86.2

Пусть у вас есть папка *modules/cart*. Сделайте так, чтобы все классы из этой папки относились к пространству имен Modules\Cart.

Задача 86.3

Пусть у вас есть папка *modules/shop/cart/*. Сделайте так, чтобы все классы из этой папки относились к пространству имен Modules\Shop\Cart.

Некоторые замечания

В примерах выше имена пространств имен совпадают с именами папок, в которых хранятся файлы. Делать так - хорошая практика, но обязательным это не является.

Задача 86.4

Пусть у вас есть папка *modules*, а в ней файл marketCart.php и файл shopCart.php. Пусть в обоих файлах размещается класс Cart. Сделайте так, чтобы класс из первого файла принадлежал пространству имен Market\Cart, а из второго файла - пространству Shop\Cart.