Пространства имен

Обзор пространств имен

В любой операционной системе директории служат для группировки связанных файлов и выступают в качестве пространства имен для находящихся в них файлов. В качестве конкретного примера файл foo.txt может находиться сразу в обоих директориях: /home/greg и /home/other, но две копии foo.txt не могут существовать в одной директории. Кроме того, для доступа к foo.txt извне директории /home/greg, мы должны добавить имя директории перед именем файла используя разделитель, чтобы получить /home/greg/foo.txt. Этот же принцип распространяется и на пространства имен в программировании.

В PHP пространства имен используются для решения двух проблем, с которыми сталкиваются авторы библиотек и приложений при создании повторно используемых элементов кода, таких как классы и функции:
1. Конфликт имен между вашим кодом и внутренними классами/функциями/константами PHP или сторонними.
2. Возможность создавать псевдонимы (или сокращения) для Ну_Очень_Длинных_Имен, чтобы облегчить первую проблему и улучшить читаемость исходного кода.

Пространства имен PHP предоставляют возможность группировать логически связанные классы, интерфейсы, функции и константы.

Названия пространств имен PHP и php, и составные названия, начинающиеся с этих (такие как PHP\Classes), являются зарезервированными для нужд языка и их не следует использовать в пользовательском коде.

Определение пространств имен

Пространства имен объявляются с помощью зарезервированного слова namespace. Файл, содержащий пространство имен, должен содержать его объявление в начале перед любым другим кодом, кроме зарезервированного слова declare.

```php
<?php
namespace MyProject;

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

?>
```

Определение подпространств имен

Так же как файлы и каталоги, пространства имен PHP позволяют создавать иерархию имен. Таким образом, имя пространства может быть определено с подуровнями.

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

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

?>
```

Вышеприведенный пример создает константу MyProject\Sub\Level\CONNECT_OK, класс MyProject\Sub\Level\Connection и функцию MyProject\Sub\Level\connect.

Описание нескольких пространств имен в одном файле

Несколько пространств имен также можно описать в одном файле с помощью двух допустимых синтаксических конструкций.

```php
<?php
namespace MyProject;

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

namespace AnotherProject;

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

Данный синтаксис не рекомендуется для комбинирования пространств имен в одном файле. Вместо этого рекомендуется использовать альтернативный синтаксис со скобками.

```php
<?php
namespace MyProject {

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

namespace AnotherProject {

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

Настоятельно не рекомендуется при программировании комбинировать несколько пространств имен в один файл. Основным применением этому может быть объединение нескольких PHP файлов в один файл.

Ключевое слово namespace и константа __NAMESPACE__

PHP поддерживает два способа к абстрактно доступным элементам в текущем пространстве имен таким, как магическая константа __NAMESPACE__ и ключевое слово namespace.

Значение константы __NAMESPACE__ - это строка, которая содержит имя текущего пространства имен. В глобальном пространстве, вне пространства имен, она содержит пустую строку.

```php
<?php
namespace MyProject;

echo '"', __NAMESPACE__, '"'; // выводит "MyProject"
?>
```

Пример использование константы __NAMESPACE__ в глобальном пространстве

```php
<?php

echo '"', __NAMESPACE__, '"'; // выводит ""
?>
```

Использование пространств имен: импорт/создание псевдонима имени

Возможность ссылаться на внешнее абсолютное имя по псевдониму или импортирование - это важная особенность пространств имен. Это похоже на возможность файловых систем unix создавать символические ссылки на файл или директорию.

В PHP создание псевдонима имени выполняется с помощью оператора use.

Глобальное пространство

Без определения пространства имен, определения всех классов и функций находятся в глобальном пространстве - также как это было в PHP до введения пространств имен. Добавление префикса \ к именам означает, что это имя должно находиться в глобальном пространстве, даже если вы находитесь в контексте определенного пространства имен.

```php
<?php
namespace A\B\C;

/* Эта функция является A\B\C\fopen */
function fopen() {
     /* ... */
     $f = \fopen(...); // вызов глобальной функции fopen
     return $f;
}
?>
```