Что такое пространство имен – namespaces – в PHP?

Доброго времени суток, уважаемые читатели нашего блога! В прошлой статье мы начали тему, связанную с объектно-ориентированным программированием в PHP. Сегодня мы копнем поглубже и рассмотрим **пространство имен – namespaces** – в языке PHP, используя простые примеры программирования. И первая хорошая новость – Namespaces прост в изучении. Начнем!
Создадим класс в PHP

```
<?php
class Foo
{
  public function doAwesomeFooThings ()
  {
   // здесь должен быть ваш код
  }
}
?>
```

Мы заполняем PHP 5.2 класс, который делает много важных вещей. Например, скажем «Привет» читателям:

```
<?php
  class Foo
  {
  public function doAwesomeFooThings ()
    {
     echo “Привет, читатели!”;
  }
  }
  ?>
```

Как использовать класс *Foo*? Использование *Foo* простое, напишем:

```
<?php
```

```
  require ‘foo.php’;
  $foo = new Foo();
  ?>
```

Для того, чтобы идти в ногу со временем, давайте используем новое понятие, которое появилось в PHP 5.3 – **пространство имен Namespace**. Пространство имен (namespace) используется подобно вложенным папкам, добавим это пространство, расположенное в *Acme\Tools*.

```
<?php
// это файл foo.php
namespace Acme\Tools;
class Foo
{
  public function doAwesomeFooThings ()
  {
    echo “Привет, читатели”;
  }
}
?>
```

Для использования *Foo* нужно назвать его по-новому. Это будет ссылка на файл, представляющая собой полный путь.

```
<?php
require ‘foo.php’;
$foo = new \Acme\Tools\Foo();
?>
```

Вот и все. Добавляя пространство имен в класс – это словно организация файлов из одной директории в несколько поддиректорий. Для ссылки на класс используйте его имя, начинающееся со слеша «\». Большое имя воспринимается тяжело, поэтому давайте добавим короткое. Для этого дадим классу *\Acme\Tools\Foo()* псевдоним.

```
<?php
require ‘foo.php’;
use \Acme\Tools\Foo as SomeFooClass;
$foo = new SomeFooClass();
?>
```

Можно как-нибудь назвать класс, либо использовать имя по умолчанию *Foo*.

```
<?php
require ‘foo.php’;
use \Acme\Tools\Foo;
$foo = new Foo();
?>
```

Хорошо. Как насчет старых классов PHP без имени пространства? Для этого перейдем к подходящему для нашего случая классу *datetime*. И возьмем несколько новых фишек от PHP 5.3. Создание нового объекта даты *datetime* выглядит аналогично:

```
<?php
require ‘foo.php’;
use \Acme\Tools\Foo;
$foo = new Foo();
$dt = new DateTime();
?>
```

Для обычного файла это все еще работает. А вот в файле с объявлением пространства namespace PHP считает, что речь идет о файле в директории *Acme**Tools*:

```
<?php
namespace Acme\Tools;
class Foo
{
  public function doAwesomeFooThings ()
  {
     echo “Hi listeners”;
     // здесь поиск будет вестись в директории \Acme\Tools
     $dt = new DateTime();
  }
}
?>
```

Или вы можете сослаться на класс в глобальном пространстве, используя полное имя *Datetime*:

```
<?php
namespace Acme\Tools;
class Foo
{
  public function doAwesomeFooThings ()
  {
     echo “Hi listeners”;
     // здесь поиск будет вестись в глобальном пространстве
     $dt = new \DateTime();
  }
}
?>
```

Предыдущий фрагмент кода показывает, что мы обращаемся к глобальному классу *DateTime*

Чтобы обратиться глобальному классу *DateTime* можно это сделать с помощью *use*:

```
<?php
namespace Acme\Tools;
use \DateTime;
class Foo
{
  public function doAwesomeFooThings ()
    {
     echo “Hi listeners”;
     // здесь поиск будет вестись в глобальном пространстве
     $dt = new DateTime();
  }
}
?>
```

Использование *use* выглядит глупо, но оно говорит PHP, что когда вы вызываете DateTime, вы имеете в виду класс без имени, указывающий на время. Если убрать слэш перед DateTime, то все будет работать точно так же.

```
<?php
namespace Acme\Tools;
use DateTime;
```

```
class Foo
{
  public function doAwesomeFooThings ()
  {
     echo “Hi listeners”;
     // здесь поиск будет вестись в глобальном пространстве
     $dt = new DateTime();
  }
}
?>
```

И вы даже не заметите этого. OK. Пока!

P.S.