What are namespaces in the PHP language?

Dear readers of our blog! In the last article we wrote about two methods of inserting a context advertisement using the PHP language. Today we will consider **PHP namespaces**, using the simple examples of programming. The good news – the Namespaces are easy. To preview we'll challenge ourselves to explain it quickly. Let's go.

Create class in PHP.

```
<?php
class Foo
{
  public function doAwesomeFooThings ()
  {
  // you should write your code here
  }
}
?>
```

Me Foo – he is a php 5.2 class, that does a lot of impotent things. Say Hi to the readers:

```
<?php
  class Foo
  {
  public function doAwesomeFooThings ()
    {
     echo "Hi, readers!";
  }
  }
  ?>
```

Ok. Using Foo is easy: simply new Foo().

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

To keep up with the times let’s put a new brand php 5.3 Namespace. A Namespace like a directory and by adding namespace, who now lives in Acme\Tools.

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

To use Foo we call him by its fancy new name. This is just the referring to a file with its absolute path.

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

And that’s really it. Adding a namespace to a class is like organizing files from one directory into a bunch of subdirectories. To refer to a class use it fully qualified name starting with a \.
Since running around but this giant name is a drug, let’s add a shortcut. The use statement let us call \Acme\Tools\Foo() class by a nickname.

```
<?php
```

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

We can call anything or just led it default to Foo.

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

Great. But what about old school none name space php classes. For that le’s pick on datetime – a handy class that’s called a php. And got some new bells and whistles in php 5.3. For ever and ever creating a new datetime object is the same – new datetime().

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

And differ in a normal file this still works. But in a namespace file php think you are talking about class in the Acme\Tools namespace.

```
<?php
namespace Acme\Tools;
class Foo
{
  public function doAwesomeFooThings ()
  {
     echo “Hi, readers!";
     // This resolves to \Acme\Tools
```

```
      $dt = new DateTime();
  }
}
?>
```

You can either refer to a class by its fully qualified name \Datetime.

```
<?php
namespace Acme\Tools;
class Foo
{
  public function doAwesomeFooThings ()
  {
     echo “Hi, readers!”;
     // This resolves to global scope
     $dt = new \DateTime();
  }
}
?>
```

Or add a use statement.

```
<?php
namespace Acme\Tools;
use \DateTime;
class Foo
{
  public function doAwesomeFooThings ()
    {
     echo “Hi, readers!”;
     // This resolves to global scope
     $dt = new DateTime();
  }
}
?>
```

The use statement looks silly. But it tells php that when you use datetime you mean a none name space class, state time. Or

give of the beginning \ with the use statement. Everything works completely the same with or without it.

```
<?php
namespace Acme\Tools;
use DateTime;
class Foo
{
  public function doAwesomeFooThings ()
  {
     echo “Hi, readers!”;
     // This resolves to global scope
     $dt = new DateTime();
  }
}
?>
```

And we typically don’t see it. OK. Bye!