PHP 实践之路 - 面向对象篇课程

主讲：孙胜利

新浪微博：@私房库

微博主页：http://weibo.com/sifangku

第四章：命名空间

一、定义命名空间

二、使用命名空间

三、__NAMESPACE__常量和 namespace 关键字

四、别名或导入

导语：

1、如何才能在计算机里拥有同名的文件？

把他们放在不同的文件夹里

2、那么如何在 PHP 运行过程中能加载同名的类或者函数、常量等？

把他们放在不同的“命名空间”中

3、怎么造出“命名空间”呢？怎么把代码放进去呢？怎么使用放进去的类或者函数、常量呢?

这些就是我们本章学习的内容了

定义命名空间

1、命名空间通过关键字 namespace 来声明

namespace　命名空间的名称;

注：

1）命名空间的名称自行定义，只要符合 PHP 的命名规则即可，我们习惯用大驼峰形式的名称

2）必须在其它所有代码（除了 declare）之前声明命名空间

3）只有以下类型的代码受命名空间的影响：类（包括抽象类和 traits）、接口、函数和常量

4）命名空间中常量的定义有些注意点

5）命名空间和文件所在目录是没有任何关系的，但是我们极力命名空间应该和所在的目录人为的关联起来，否则时间长了维护起来会比较混乱。

知识补充：

如果没有定义任何命名空间,所有的类与函数的定义都是在**全局空间**,与 PHP 引入命名空间概念前一样。被 include、require 的文件里的代码 默认为全局命名空间。

2、定义层次化的命名空间

PHP 命名空间允许指定层次化的命名空间的名称，类似于多层目录!

```
namespace MyProject\Sub\Level;
```

注：

层次化的命名空间和文件所在目录是没有任何关系的，但是我们极力推荐层次化的命名空间应该和所在的目录人为的建立点关联，否则时间长了维护起来会比较混乱。

3、在同一个文件中定义多个命名空间

在实际的编程实践中，非常不提倡在同一个文件中定义多个命名空间。

方法一：

```php
<?php
namespace MyProject;

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

namespace AnotherProject;

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

方法二：

```php
<?php
namespace MyProject {

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

namespace AnotherProject {

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

注意,将全局的非命名空间中的代码与命名空间中的代码组合在一起，只能使用大括号形式的语法。全局代码必须用一个不带名称的 namespace 语句加上大括号括起来：

```php
<?php
namespace MyProject {

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

namespace { // global code
session_start();
$a = MyProject\connect();
echo MyProject\Connection::start();
}
?>
```

使用命名空间

1、访问 PHP 命名空间中的内容与在文件系统中访问一个文件是同样的原理。

1）相对文件名形式如 foo.txt。它会被解析为 当前目录/foo.txt

2）相对路径名形式如 subdirectory/foo.txt。它会被解析为 当前目录/subdirectory/foo.txt

3）绝对路径名形式如/main/foo.txt。它会被解析为/main/foo.txt

同理：

1)**非限定名称**,即不包含前缀的类名称,例如 $a=new foo(); 如果当前命名空间是 currentnamespace,

foo 将被解析为 currentnamespace\foo。

如果使用 foo 的代码是全局的，不包含在任何命名空间中的代码，则 foo 会被解析为 foo。

注：

1>在一个命名空间中，当 PHP 遇到一个非限定的类、函数或常量名称时，它使用不同的优先策略来解析该名称。

2>类名称总是解析到当前命名空间中的名称。因此在访问系统内部或不包含在命名空间中的类名称时，必须使用完全限定名称。

3>对于函数和常量来说，如果当前命名空间中不存在该函数或常量，PHP 会退而使用全局空间中的函数或常量。

2）**限定名称,**即包含前缀的名称，例如 $a = new subnamespace\foo();

若当前命名空间是 currentnamespace，则 foo 会被解析为 currentnamespace\subnamespace\foo

如果使用 foo 的代码是全局的，foo 会被解析为 subnamespace\foo

3）**完全限定名称**，即包含了全局前缀操作符的名称，例如， $a = new \currentnamespace\foo();

在这种情况下，foo 总是被解析为\currentnamespace\foo

注意：访问任意全局类、函数或常量，都可以使用完全限定名称，例如 \strlen() 或 \Exception 或 \INI_ALL。

总结：使用时首先需要看下当前使用的环境是在哪个命名空间下、然后再看下要使用的类（包括抽象类和 traits）、接口、函数和常量在哪个命名空间下。

__NAMESPACE__常量和 namespace 关键字

1、PHP 支持两种抽象的访问当前命名空间内部元素的方法，__NAMESPACE__ 魔术常量和 namespace 关键字。

2、常量__NAMESPACE__的值是包含当前命名空间名称的字符串。在全局的，不包括在任何命名空间中的代码，它包含一个空的字符串。

3、关键字 namespace 可用来显式访问当前命名空间或子命名空间中的元素。

它类似于类中的 self 操作符。

别名或导入

1、允许通过别名引用或导入外部的完全限定名称，是命名空间的一个重要特征。我们可以为类名称使用别名、为接口使用别名或为命名空间名称使用别名。PHP 5.6 开始允许导入函数或常量或者为它们设置别名。

2、如何操作？

举例：

```
use My\Full\NSname;

use My\Full\Classname as Another;

use Expection;// 导入一个全局类

use function My\Full\functionName;//PHP 5.6+

use function My\Full\functionName as func;//PHP 5.6+

use const My\Full\CONSTANT;//PHP 5.6+

use My\Full\Classname as Another, My\Full\NSname;//多条 use 语句简写
```

注：导入时前导的反斜杠是不必要的也不推荐的，因为导入的名称必须是完全限定的，不会根据当前的命名空间作相对解析。