后盾网 人人做后盾

www.houdunwang.com

namespace

命名空间

后盾网 2011-2016 v5.0

namespace概述

- 什么是命名空间？从广义上来说，命名空间是一种封装事物的方法。在很多地方都可以见到这种抽象概念。例如，在操作系统中目录用来将相关文件分组，对于目录中的文件来说，它就扮演了命名空间的角色。具体举个例子，文件 foo.txt 可以同时在目录/home/greg 和 /home/other 中存在，但在同一个目录中不能存在两个 foo.txt 文件。另外，在目录 /home/greg 外访问 foo.txt 文件时，我们必须将目录名以及目录分隔符放在文件名之前得到 /home/greg/foo.txt。这个原理应用到程序设计领域就是命名空间的概念。

解决的问题

- 用户编写的代码与PHP内部的类/函数/常量或第三方类/函数/常量之间的名字冲突。
- 为很长的标识符名称(通常是为了缓解第一类问题而定义的)创建一个别名（或简短）的名称，提高源代码的可读性。

作用类型

- 类，函数和常量

什么是namespace

声明注意

- 所有非 PHP 代码包括空白符都不能出现在命名空间的声明之前

示例

- <?php
- namespace MyProject;
- const CONNECT_OK = 1;
- class Connection { /* ... */ }
- function connect() { /* ... */ }
- ?>

基本使用

声明注意

- 与目录和文件的关系很象，PHP 命名空间也允许指定层次化的命名空间的名称。因此，命名空间的名字可以使用分层

示例

- <?php
- namespace MyProject\Sub\Level;
- const CONNECT_OK = 1;
- class Connection { /* ... */ }
- function connect() { /* ... */ }
- ?>

子命名空间

声明注意

- 可以在同一个文件中定义多个命名空间,但不建议使用这种方式

示例

- <?php
- namespace MyProject;
- const CONNECT_OK = 1;
- class Connection { /* ... */ }
- function connect() { /* ... */ }
- namespace AnotherProject;
- const CONNECT_OK = 1;
- class Connection { /* ... */ }
- function connect() { /* ... */ }
- ?>

多命名空间-简写

示例

```
<?php
namespace MyProject {
    const CONNECT_OK = 1;
    class Connection { /* ... */ }
    function connect() { /* ... */  }
}
namespace AnotherProject {
    const CONNECT_OK = 1;
    class Connection { /* ... */ }
    function connect() { /* ... */  }
}
?>
```

注意

- 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。

多命名空间-大括号写法

解释

可以将 PHP 命名空间与文件系统作一个简单的类比,即文件系统中的三种方式:

- 相对文件名形式如foo.txt
- 相对路径名形式如subdirectory/foo.txt
- 绝对路径名形式如/main/foo.txt

非限定名称(不包含前缀的类名称)

- 例如 $a=new foo(); 或 foo::staticmethod();。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。 警告：如果命名空间中的函数或常量未定义，则该非限定的函数名称或常量名称会被解析为全局函数名称或常量名称。

使用命名空间

限定名称(包含前缀的类名称)

- 例如 $a = new subnamespace\foo(); 或 subnamespace\foo::staticmethod();。如果当前的命名空间是 currentnamespace，则 foo 会被解析为 currentnamespace\subnamespace\foo。如果使用 foo 的代码是全局的，不包含在任何命名空间中的代码，foo 会被解析为subnamespace\foo。

完全限定名称(包含了全局前缀操作符的名称)

- 例如， $a = new \currentnamespace\foo(); 或 \currentnamespace\foo::staticmethod();。在这种情况下，foo 总是被解析为代码中的文字名(literal name)currentnamespace\foo。

注意

- 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()

使用命名空间

__NAMESPACE__

- 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。

示例

```
<?php
namespace MyProject;
function get($classname)
{
    $a = __NAMESPACE__ . '\\' . $classname;
    return new $a;
}
?>
```

__NAMESPACE__常量

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。它等价于类中的 self 操作符。

示例

```
<?php
namespace MyProject;
class test{
        static function run(){
                echo 'hdphp';
                }
        };
namespace\test::run();
?>
```

namespace关键字

别名导入

- 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。这有点类似于在类 unix 文件系统中可以创建对其它的文件或目录的符号连接。

别名/导入

全局空间

- 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。在名称前加上前缀 \ 表示该名称是全局空间中的名称，即使该名称位于其它的命名空间中时也是如此。

全局空间

说明

- 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。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。

示例

- 前提全局空间有一个p函数
- <?php
- namespace A\B\C;
- function p(){
- echo ‘ABC里里面的p’
- }
- p(); //优先调用当前空间的p
- \p();才能调用全局空间
- ?>

解析优先级