对于命名空间，官方文档已经说得很详细[[查看](http://www.php.net/manual/zh/language.namespaces.rationale.php)]，我在这里做了一下实践和总结。

命名空间一个最明确的目的就是解决重名问题，PHP中不允许两个函数或者类出现相同的名字，否则会产生一个致命的错误。这种情况下只要避免命名重复就可以解决，最常见的一种做法是约定一个前缀。

例：项目中有两个模块：article和message board，它们各自有一个处理用户留言的类Comment。之后我可能想要增加对所有用户留言的一些信息统计功能，比如说我想得到所有留言的数量。这时 候调用它们Comment提供的方法是很好的做法，但是同时引入各自的Comment类显然是不行的，代码会出错，在另一个地方重写任何一个 Comment也会降低维护性。那这时只能重构类名，我约定了一个命名规则，在类名前面加上模块名，像这样：Article\_Comment、 MessageBoard\_Comment

可以看到，名字变得很长，那意味着以后使用Comment的时候会写上更多的代码（至少字符多了）。并且，以后如果要对各个模块增加更多的一些整合 功能，或者是互相调用，发生重名的时候就需要重构名字。当然在项目开始的时候就注意到这个问题，并规定命名规则就能很好的避免这个问题。另一个解决方法可 以考虑使用命名空间。

**注明：**

本文提到的常量：PHP5.3开始const关键字可以用在类的外部。const和define都是用来声明常量的（它们的区别不详述），但是在命名空间里，define的作用是全局的，而const则作用于当前空间。我在文中提到的常量是指使用const声明的常量。

**基础**命名空间将代码划分出不同的空间（区域），每个空间的常量、函数、类（为了偷懒，我下边都将它们称为元素）的名字互不影响， 这个有点类似我们常常提到的‘封装'的概念。

**创建一个命名空间需要使用namespace关键字，这样：**

复制代码代码如下:

<?php

//创建一个名为'Article'的命名空间  
namespace Article;

?>

要注意的是，当前脚本文件的第一个命名空间前面不能有任何代码，下面的写法都是错误的：

复制代码代码如下:

//例一  
//在脚本前面写了一些逻辑代码

<?php

$path = "/";

class Comment { }

namespace Article;

?>

//例二  
//在脚本前面输出了一些字符

<html></html>  
<?php

namespace Article;

?>

为什么要说第一个命名空间呢？因为同一脚本文件中可以创建多个命名空间。

下面我创建了两个命名空间，顺便为这两个空间各自添加了一个Comment类元素：

复制代码代码如下:

<?php

//创建一个名为'Article'的命名空间  
namespace Article;

//此Comment属于Article空间的元素  
class Comment { }

//创建一个名为'MessageBoard'的命名空间  
namespace MessageBoard;

//此Comment属于MessageBoard空间的元素  
class Comment { }  
?>

在不同空间之间不可以直接调用其它元素，需要使用命名空间的语法：

复制代码代码如下:

<?php

namespace Article;

class Comment { }

namespace MessageBoard;

class Comment { }

//调用当前空间（MessageBoard）的Comment类  
$comment = new Comment();

//调用Article空间的Comment类  
$article\_comment = new \Article\Comment();

?>

可以看到，在MessageBoard空间中调用article空间里的Comment类时，使用了一种像文件路径的语法： \空间名\元素名

除了类之外，对函数和常量的用法是一样的，下面我为两个空间创建了新的元素，并在MessageBoard空间中输出了它们的值。

复制代码代码如下:

<?php

namespace Article;

const PATH = '/article';

function getCommentTotal() {  
    return 100;  
}

class Comment { }

namespace MessageBoard;

const PATH = '/message\_board';

function getCommentTotal() {  
    return 300;  
}

class Comment { }

//调用当前空间的常量、函数和类  
echo PATH; ///message\_board  
echo getCommentTotal(); //300  
$comment = new Comment();

//调用Article空间的常量、函数和类  
echo \Article\PATH; ///article  
echo \Article\getCommentTotal(); //100  
$article\_comment = new \Article\Comment();

?>

然后我的确得到了Article空间的元素数据。

**子空间**命名空间的调用语法像文件路径一样是有道理的，它允许我们自定义子空间来描述各个空间之间的关系。

抱歉我忘了说，article和message board这两个模块其实都是处于同一个blog项目内。如果用命名空间来表达它们的关系，是这样：

复制代码代码如下:

<?php

//我用这样的命名空间表示处于blog下的article模块  
namespace Blog\Article;

class Comment { }

//我用这样的命名空间表示处于blog下的message board模块  
namespace Blog\MessageBoard;

class Comment { }

//调用当前空间的类  
$comment = new Comment();

//调用Blog\Article空间的类  
$article\_comment = new \Blog\Article\Comment();

?>

而且，子空间还可以定义很多层次，比如说 Blog\Article\Archives\Date

**公共空间**我有一个common\_inc.php脚本文件，里面有一些好用的函数和类：

复制代码代码如下:

<?php

function getIP() { }

class FilterXSS { }

?>

在一个命名空间里引入这个脚本，脚本里的元素不会归属到这个命名空间。如果这个脚本里没有定义其它命名空间，它的元素就始终处于公共空间中：

复制代码代码如下:

<?php

namespace Blog\Article;

//引入脚本文件  
include './common\_inc.php';

$filter\_XSS = new FilterXSS(); //出现致命错误：找不到Blog\Article\FilterXSS类

$filter\_XSS = new \FilterXSS(); //正确

?>

调用公共空间的方式是直接在元素名称前加 \ 就可以了，否则PHP解析器会认为我想调用当前空间下的元素。除了自定义的元素，还包括PHP自带的元素，都属于公共空间。

要提一下，其实公共空间的函数和常量不用加 \ 也可以正常调用（不明白PHP为什么要这样做），但是为了正确区分元素，还是建议调用函数的时候加上 \

**名称术语**在说别名和导入之前，需要知道关于空间三种名称的术语，以及PHP是怎样解析它们的。官方文档说得非常好，我就直接拿来套了。

1.非限定名称，或不包含前缀的类名称，例如 $comment = new Comment();。如果当前命名空间是Blog\Article，Comment将被解析为Blog\Article\Comment。如果使用 Comment的代码不包含在任何命名空间中的代码（全局空间中），则Comment会被解析为Comment。  
  
2.限定名称，或包含前 缀的名称，例如 $comment = new Article\Comment();。如果当前的命名空间是Blog，则Comment会被解析为Blog\Article\Comment。如果使用 Comment的代码不包含在任何命名空间中的代码（全局空间中），则Comment会被解析为Comment。  
  
3.完全限定名称，或包含了全局前缀操作符的名称，例如 $comment = new \Article\Comment();。在这种情况下，Comment总是被解析为代码中的文字名(literal name)Article\Comment。

其实可以把这三种名称类比为文件名（例如 comment.php）、相对路径名（例如 ./article/comment.php）、绝对路径名（例如 /blog/article/comment.php），这样可能会更容易理解。

**我用了几个示例来表示它们：**

复制代码代码如下:

<?php

//创建空间Blog  
namespace Blog;

class Comment { }

//非限定名称，表示当前Blog空间  
//这个调用将被解析成 Blog\Comment();  
$blog\_comment = new Comment();

//限定名称，表示相对于Blog空间  
//这个调用将被解析成 Blog\Article\Comment();  
$article\_comment = new Article\Comment(); //类前面没有反斜杆\

//完全限定名称，表示绝对于Blog空间  
//这个调用将被解析成 Blog\Comment();  
$article\_comment = new \Blog\Comment(); //类前面有反斜杆\

//完全限定名称，表示绝对于Blog空间  
//这个调用将被解析成 Blog\Article\Comment();  
$article\_comment = new \Blog\Article\Comment(); //类前面有反斜杆\

//创建Blog的子空间Article  
namespace Blog\Article;

class Comment { }

?>

其实之前我就一直在使用非限定名称和完全限定名称，现在它们终于可以叫出它们的名称了。

**别名和导入**别名和导入可以看作是调用命名空间元素的一种快捷方式。PHP并不支持导入函数或常量。

**它们都是通过使用use操作符来实现：**

复制代码代码如下:

<?php

namespace Blog\Article;

class Comment { }

//创建一个BBS空间（我有打算开个论坛）  
namespace BBS;

//导入一个命名空间  
use Blog\Article;  
//导入命名空间后可使用限定名称调用元素  
$article\_comment = new Article\Comment();

//为命名空间使用别名  
use Blog\Article as Arte;  
//使用别名代替空间名  
$article\_comment = new Arte\Comment();

//导入一个类  
use Blog\Article\Comment;  
//导入类后可使用非限定名称调用元素  
$article\_comment = new Comment();

//为类使用别名  
use Blog\Article\Comment as Comt;  
//使用别名代替空间名  
$article\_comment = new Comt();

?>

我注意到，如果导入元素的时候，当前空间有相同的名字元素将会怎样？显然结果会发生致命错误。

**例：**

复制代码代码如下:

<?php

namespace Blog\Article;

class Comment { }

namespace BBS;

class Comment { }

Class Comt { }

//导入一个类  
use Blog\Article\Comment;  
$article\_comment = new Comment(); //与当前空间的Comment发生冲突，程序产生致命错误

//为类使用别名  
use Blog\Article\Comment as Comt;  
$article\_comment = new Comt(); //与当前空间的Comt发生冲突，程序产生致命错误

?>

动态调用  
PHP提供了namespace关键字和\_\_NAMESPACE\_\_魔法常量动态的访问元素，\_\_NAMESPACE\_\_可以通过组合字符串的形式来动态访问：

复制代码代码如下:

<?php

namespace Blog\Article;

const PATH = '/Blog/article';

class Comment { }

//namespace关键字表示当前空间  
echo namespace\PATH; ///Blog/article  
$comment = new namespace\Comment();

//魔法常量\_\_NAMESPACE\_\_的值是当前空间名称  
echo \_\_NAMESPACE\_\_; //Blog\Article  
//可以组合成字符串并调用  
$comment\_class\_name = \_\_NAMESPACE\_\_ . '\Comment';  
$comment = new $comment\_class\_name();

?>

**字符串形式调用问题**  
上面的动态调用的例子中，我们看到了字符串形式的动态调用方式，如果要使用这种方式要注意两个问题。

**1. 使用双引号的时候特殊字符可能被转义**

复制代码代码如下:

<?php

namespace Blog\Article;

class name { }

//我是想调用Blog\Article\name  
$class\_name = \_\_NAMESPACE\_\_ . "\name"; //但是\n将被转义为换行符

$name = new $class\_name(); //发生致命错误

?>

**2. 不会认为是限定名称**

PHP在编译脚本的时候就确定了元素所在的空间，以及导入的情况。而在解析脚本时字符串形式调用只能认为是非限定名称和完全限定名称，而永远不可能是限定名称。

复制代码代码如下:

<?php

namespace Blog;

//导入Common类  
use Blog\Article\Common;  
//我想使用非限定名称调用Blog\Article\Common  
$common\_class\_name = 'Common';  
//实际会被当作非限定名称，也就表示当前空间的Common类，但我当前类没有创建Common类  
$common = new $common\_class\_name(); //发生致命错误：Common类不存在

//我想使用限定名称调用Blog\Article\Common  
$common\_class\_name = 'Article\Common';  
//实际会被当作完全限定名称，也就表示Article空间下的Common类，但我下面只定义了Blog\Article空间而不是Article空间  
$common = new $common\_class\_name(); //发生致命错误：Article\Common类不存在

namespace Blog\Article;

class Common { }

?>

**总结**我对PHP的命名空间刚刚接触，也不能随便给一些没有实践的建议。我个人认为命名空 间的作用和功能都很强大，如果要写插件或者通用库的时候再也不用担心重名问题。不过如果项目进行到一定程度，要通过增加命名空间去解决重名问题，我觉得工 作量不会比重构名字少。也不得不承认它的语法会对项目增加一定的复杂度，因此从项目一开始的时候就应该很好的规划它，并制定一个命名规范。