**PHP设计模式：**

**命名空间(namespace)**

链接地址: <http://php.net/manual/zh/language.namespaces.php>

**定义：**

从广义上来说，命名空间是一种封装事物的方法。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。

一个namespace就相当于对应一个文件路径，查找这个类的时候，就会去对应的文件路径查找类定义文件了。

**定义命名空间：**

通过关键字**namespace**来声明

**namespace** MyProject;

**定义子命名空间（层次化的命名空间）：**

**namespace**MyProject\Sub\Level;

**定义多个命名空间，大括号语法：**

1. 在实际的编程实践中，不提倡在同一个文件中定义多个命名空间。
2. <?php  
   namespace MyProject {  
     
   const CONNECT\_OK = 1;  
   class Connection { /\* ... \*/ }  
   function connect() { /\* ... \*/  }  
   }  
     
   namespace AnotherProject {  
     
   const CONNECT\_OK = 1;  
   class Connection { /\* ... \*/ }  
   function connect() { /\* ... \*/  }  
   }  
   ?>

类名的引用方式（3种）：

命名空间内的函数使用：命名空间的名称+斜杠“\”+函数名称

1. 非限定名称，或不包含前缀的类名称：以当前命名空间+它， $a=new foo();
2. 限定名称,或包含前缀的名称： 以当前命名空间+它，$a = new subnamespace\foo();
3. 完全限定名称，或包含了全局前缀操作符的名称（以斜杠开头表示根）：$a = new \currentnamespace\foo();

**注意：**

1. 命名空间必须是程序脚本的第一条语句，如果一个文件中包含命名空间，它必须在其它所有代码之前声明命名空间，除了一个以外：**[declare](http://php.net/manual/zh/control-structures.declare.php)**关键字（用于定义源文件编码方式：declare(encoding='UTF-8');）。
2. 只有以下类型的代码受命名空间的影响，

它们是：类（包括抽象类和traits）、接口、函数和常量。

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

3.同一个命名空间可以定义在多个文件中，即允许将同一个命名空间的内容分割存放在不同的文件中。

4.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

**类自动载入特性：（php5.3之后）**

Sql\_autoload\_register( )函数，允许有多个autoload（ ）函数，并且不会冲突

PHP面向对象PSR-0规范：

1. 命名空间必须与绝对路径一致
2. 类名首字母必须大写
3. 除入口文件外，其他“.php”必须只有一个类