# 编码规范

## 1 目的

目的是让软件项目遵守公共一致的标准，形成良好一致的编程风格，使参与者可以很快的适应环境，减少编码出错的机会，在团队协作开发和项目后期维护中有更高的效率，以达到事半功倍的效果，使项目长远健康的发展。防止部分参与者出于节省时间的需要，自创一套风格并养成终生的习惯，导致其它人在阅读时浪费过多的时间和精力。

## 1.2 目标

能够遵守公共一致的编码标准对任何开发项目都很重要，特别是在多人的开发团队中， 编码标准能帮助确保代码的质量、减少 Bug 和容易维护。

* 适应环境，方便的融入到项目团队中
* 在一致的环境下，团队协作中有更高的效率，团队的成员可以减少犯错的机会
* 程序员可以方便的了解其他人的代码，弄清程序的状况，就和看自已的代码一样
* 防止自创一套风格并养成终生的习惯，一次次的犯同样的错误

## 1.3 开发工具

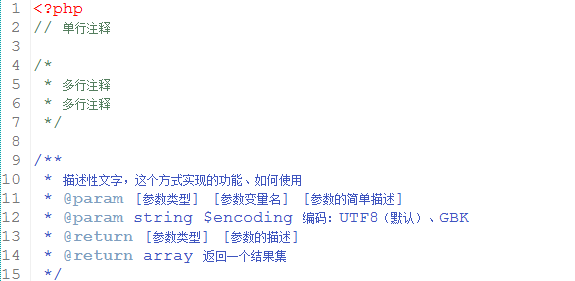
Zend Studio + Dreamweaver

## 1.4 PHP标记

* 统一使用完整的、标准的定界标记（<?php ?>）
* 对于只有包含PHP代码的文件，不使用结束标记（?>）。防止被意外注入，导致header()、setcookie()、session\_start()等设置头信息的函数发生失败

## 1.5 注释规范

主要使用的注释风格



* 文件头部注释

文件头部增加注释有利于团队之间功能对接，主要信息可以有：版权声明（@Copyright）、作者（@author）、功能简短说明、文件详细说明、版本信息（@version）、文件名、程序语言、编码、文件创建日期、修改日期信息

推荐格式：



* 类文件注释

类的简介、类的详细描述、类名（**@package**）、作者（**@author**）、版本信息（**@version**）、修改记录、所有非私有函数列表（**@param**）

* 方法或函数注释

函数描述、所有参数（**@param**）、所有可能的返回值（**@return**）

如果应用全局变量必须使用global标记（**@global**）

调用了其他函数或类的地方使用link或其他标记连接到应用部分（**@link**）

* 临时性代码注释

在一些调试代码的后面按照该方法添加注释

d

## 1.6 单双引号的使用

* 因为单引号包裹的字符串不会被PHP解析，所以除了复杂的字符串之外要求全部使用单引号，示例：

c

## 1.7 命名原则

所有命名必须具有描述性，杜绝使用拼音，尽量不要使用数字

* 文件命名原则

所有文件名必须小写

普通文件：名+.php

类文件 ：名+.class.php

函数库 ：名+.func.php

配置文件：名+.inc.php

* 类命名原则

首字母大写，如果是多个单词组合，则所有单词首字母都大写

* 函数名或方法名

函数或方法名中必须要有描述性，且尽量带有动词描述（get、set等)

* 常量名

常量名必须使用大写，如果是多个单词使用下划线分开

* 全局变量

以 **$global\_** 开头

## 2.0 类中成员属性和变量的声明

* 类中成员属性的声明必须放到类的顶部，也就是在方法上面声明，而且需要使用合 适的权限限制外部访问级别；
* 所有变量都必须初始化

$number = 0; //数值型初始化

$string = ''; //字符串初始化

$array = array(); //数组初始化

* 判断一个无法确定（不知道是否已被赋值）的变量时，可用 empty()或 isset()，而不要直接使用 if($switch)的形式，除非你确切的知道此变量一定已经被初始化并赋 值。
* 判断一个变量是否为数组，请使用 is\_array()，这种判断尤其适用于对数组进行遍 历的操作，例如 foreach()，因为如果不事先判断，foreach()会对非数组类型的变量 报错；
* 判断一个数组元素是否存在，可使用 isset($array['key'])，也可使用 empty();

## 2.1 函数的定义与使用

* 声明函数时参数的名字和变量的命名规范一致；
* 函数定义中的左小括号，与函数名紧挨，中间无需空格；
* 开始的左大括号与函数定义为同一行，中间加一个空格，不要另起一行；
* 如果使用具有默认值的参数，应该位于参数列表的后面；
* 函数不管在调用还是在声明的时候，参数与参数之间都要加入一个空格；
* 必须仔细检查并切实杜绝函数起始缩进位置与结束缩进位置不同的现象。
* 建议不要使用全局函数；

a