缩进

缩进使用4个空格，而不是 tab。如果你使用 Emacs 编辑 PEAR 代码，你应该设置 indent-tabs-mode 为 nil。下面是一个 mode hook 的示例，用于设置 Emacs 符合缩进标准（你必须确保在编辑 PHP 文件时，这些设置发生作用）：

  (defun php-mode-hook ()  
  (setq tab-width 4  
        c-basic-offset 4  
        c-hanging-comment-ender-p nil  
        indent-tabs-mode  
        (not  
         (and (string-match "/\\(PEAR\\|pear\\)/" (buffer-file-name))  
              (string-match "\.php$" (buffer-file-name))))))

这里是同等效果的 vim 规则：

  set expandtab  
  set shiftwidth=4  
  set softtabstop=4  
  set tabstop=4

控制结构

控制结构包含 if、for、while、switch 等。这里有一个 if 语句的示例和一个 switch 语句的示例：

if 语句的示例：

<?php  
if ((condition1) || (condition2)) {  
    action1;  
} elseif ((condition3) && (condition4)) {  
    action2;  
} else {  
    defaultaction;  
}  
?>

switch 语句的示例：

<?php  
switch (condition) {  
case 1:  
    action1;  
    break;  
case 2:  
    action2;  
    break;  
default:  
    defaultaction;  
    break;  
}  
?>

控制语句应该在控制关键词和开始的圆括号之间应该有一个空格，以此和函数调用进行区别。

强烈建议你总是使用花括号将控制结构各部分标识出来。即使是在技术上可以不使用花括号的地方。这可以增加代码的可读性，同时避免在结构部分增加新行后引入逻辑上的错误。

原始代码：

if (condition)  
    return true;  
else  
    return false;

修改后的代码：

if (condition)  
    do something; // 出现逻辑错误  
    return true;  
else  
    return false;

正确的做法：

if (condition) {  
    do something;  
    return true;  
} else {  
    return false;  
}

函数调用

调用函数时，函数名和开始的括号之间不应该有空白字符。参数和开始及结束的括号之间不应有空格。而除第一个参数外，其他参数都应该用一个空格分隔。这里有一个示例：

<?php  
$var = foo($bar, $baz, $quux);  
?>

像上面的示例代码，赋值运算等号两边都应该使用一个空格。如果是相关的赋值运算，应该采用下面的形式以提供更好的可读性：

<?php  
$short = foo($bar);  
$long\_variable = foo($baz);  
?>

函数定义

按照“one true brace”约定声明函数：

<?php  
function fooFunction($arg1, $arg2 = '')  
{  
    if (condition) {  
        statement;  
    }  
    return $val;  
}  
?>

“one true brace”约定就是开始的花括号单独占一行，而不是跟在其他语句后面。

具有默认值的参数应该位于参数列表的后面（事实上 PHP 语言定义也要求如此）。如果适合，函数应该总是返回一个有意义的值。这里有一个稍微长一点的示例：

<?php  
function connect(&$dsn, $persistent = false)  
{  
    if (is\_array($dsn)) {  
        $dsninfo = &$dsn;  
    } else {  
        $dsninfo = DB::parseDSN($dsn);  
    }  
  
    if (!$dsninfo || !$dsninfo['phptype']) {  
        return $this->raiseError();  
    }  
    return true;  
}  
?>

注释

类型（class）的联机文档应该符合 PHPDoc（类似于 JavaDoc）的约定。更多关于 PHPDoc 的信息可以访问 <http://www.phpdoc.de/> 获得。

此外，强烈鼓励使用非文档注释。一般性规则是对于那些容易忘记作用的代码添加简短的介绍性注释。

推荐使用 C 样式的注释（/\* \*/）和标准 C++ 注释（//)，而不应该使用 Perl/shell 样式的注释（#）。

包含代码

无论在什么地方无条件包含一个类型文件，应该使用 require\_once()。如果有条件的包含一个类型文件（例如使用工厂方法），应该使用 include\_once()。使用两者中的任何一个都能够确保类型文件只包含一次。它们共享一个文件列表，因此你不需要担心混淆他们 —— 一个文件使用 require\_once() 包含后不会在 include\_once() 中再一次被包含。

备注：include\_once() 和 require\_once() 是一个声明，而不是函数。你不需要使用圆括号将文件名扩起来（不过使用括号也不会出现错误）。

PHP 代码标记

总是使用 <?php ?> 来界定 PHP 代码，而不要使用 <? ?> 速记方式。这是为了符合 PEAR 一致性所必须的，同时也是在不同操作系统和不同安装设置环境下移植 PHP 代码所要求的。

头注释块

PEAR 发布的所有源代码文件头部都应该包含下面的注释块：

<?php  
/\* vim: set expandtab tabstop=4 softtabstop=4 shiftwidth=4: \*/  
// +----------------------------------------------------------------------+  
// | PHP version 4                                                        |  
// +----------------------------------------------------------------------+  
// | Copyright (c) 1997-2002 The PHP Group                                |  
// +----------------------------------------------------------------------+  
// | This source file is subject to version 2.0 of the PHP license,       |  
// | that is bundled with this package in the file LICENSE, and is        |  
// | available at through the world-wide-web at                           |  
// | http://www.php.net/license/2\_02.txt.                                 |  
// | If you did not receive a copy of the PHP license and are unable to   |  
// | obtain it through the world-wide-web, please send a note to          |  
// | license@php.net so we can mail you a copy immediately.               |  
// +----------------------------------------------------------------------+  
// | Authors: Original Author <author@example.com>                        |  
// | Your Name <you@example.com>                                          |  
// +----------------------------------------------------------------------+  
//  
// $ Id $  
?>

这里没有硬性规定要将一个代码贡献者的名字添加到文件注释的作者列表中。一般情况下，他们的更改属于“substantial”目录（意味大约10%到20%的代码被改写）。有一个例外就是代码贡献者重写了函数或者贡献了新的程序逻辑。

简单的代码重组和 bug 修复不应该增加新作者，这是不恰当的。

不在核心 PEAR 仓库中的文件应该有一个类似的注释块来说明版权、许可协议和作者。所有文件应该包含一个模式行（modeline，用于 vim 和 emacs）以确保一致性。

使用 CVS

这个小节的内容仅仅适用于 cvs.php.net 上使用 CVS 的包。

在 每一个文件中包含 $Id $（两个 $ 符号之间的关键字不能够有空格，但由于此文档也是用 CVS 管理，因此只好加个空格，避免被 CVS 替换掉） CVS 关键字，以便查看文件当前状态和最后修改时间等信息。如果已经存在“Last Modified:”这样的信息，则替换为 $Id $ 标记。

这个小节的其他内容假定你有关于 CVS 标记和分支（branches）的基本知识。

CVS 标记用于标识你包中的文件释放之前所作的修订。下面的列表是必需使用和建议使用的 CVS 标记：

RELEASE\_n\_n

（必需的）用于标记一个释放版本。如果你没有使用这个标记，别人就没有办法在你发布一个释放版本时从 CVS 服务器获取你的包。

QA\_n\_n

（分支，可选）如果你觉得在发布释放版本之前要提供一个候选释放版本，那么一个好主意就是增加一个分支。这个分支可以让你将释放版本隔离出来，并在正式发布释放版本之前可以为这些分支单独应用更新。期间，正常的开发工作可以在主干上继续进行。

MAINT\_n\_n

（分支，可选）如果你需要创建一个较小修改的释放版本（例如在 1.2 之后发布 1.2.1）。那么你可以创建一个分支来达到目的。

仅有 RELEASE 标记是必须的，其他标记为了方便推荐你使用。

下面是一个如何为 "Money\_Fast" 包 1.2 释放版本增加标记的示例：

$ cd pear/Money\_Fast  
$ cvs tag RELEASE\_1\_2  
T Fast.php  
T README  
T package.xml

经过上面的操作，就可以在 PEAR Web 网站上获取你的一系列释放版本了。

这里是一个如何建立 QA 分支的示例：

$ cvs tag QA\_2\_0\_BP  
...  
$ cvs rtag -b -r QA\_2\_0\_BP QA\_2\_0  
$ cvs update -r QA\_2\_0  
$ cvs tag RELEASE\_2\_0RC1  
...and then the actual release, from the same branch:  
$ cvs tag RELEASE\_2\_0

"QA\_2\_0\_BP" 标记是一个 "branch point" 标记，用于标记分支的开始。总是用这样的标记来标明分支的开始是一个好主意。MAINT 分支可以使用 RELEASE 标记作为它的分支起点。

示例中的 URL

所有示例中用到的 URL 地址都应该是 "example.com"、"example.org"、"example.net"。

命名约定

一般而言，class、函数和变量的名字应该总是能够描述让代码阅读者能够容易的知道这些代码的作用。

Classes

class 应该具有一个描述性的名字。可能时应该避免使用缩写。class 名字应该总是用一个大写字母开始。从 class 名字中也能够反映出 PEAR class 的层次。层次中的每个级别都用下划线进行分隔。好的 class 名字示例如下：

Log  
Net\_Finger  
HTML\_Upload\_Error

函数和方法

函数和方法应该使用 "studly caps" 样式命名。函数应该将所在包的名字作为前缀，以避免与其他包的函数发生名字冲突。名字的受字母（前缀之后）应该是小写，每一个新单词应该以大写字母开头。下面是一些示例：

connect()  
getData()  
buildSomeWidget()  
XML\_RPC\_serializeData()

私有 class 成员和属性（意味着 class 成员和属性只应该由同一个 class 中声明的成员使用。不过 PHP 并不支持强制性的私有命名空间）应该用一个下划线开头。示例：

\_sort()  
\_initTree()  
$this->\_status

常数

常数应该总是全部使用大写字母命名，用下划线来分隔单词。常数名字的前缀应该使用大写的 class/包 名字。例如：DB:: 包使用的所有常数都已 DB\_ 开头。

全局变量

如果你的包需要定义全局变量，那么应该用下划线跟上包的名字和另一个下划线作为开头。例如，PEAR 包使用一个全局变量名为 $\_PEAR\_destructor\_object\_list。

预定义的值 true、false 和 null

PHP 的内建值 true、false 和 null 必须全部用小写字母书写。