

# 本科毕业设计（论文）

外文参考文献译文及原文

学 院 计算机学院

专 业 计算机科学与技术

年级班别 2012级（1）班

学 号 3112005767

学生姓名 陈锐

指导教师 黄益民

2016 年 5 月

[本科毕业设计（论文） 1](#_Toc9512)

[1 简介 3](#_Toc5146)

[1.1 PHP 是什么？ 3](#_Toc4756)

[1.2 PHP 能做什么？ 4](#_Toc29992)

[2 简明教程 6](#_Toc13205)

[2.1 需要些什么？ 6](#_Toc1345)

[2.2 第一个 PHP 页面 7](#_Toc27407)

[2.3 实用的脚本 9](#_Toc25633)

[2.4 处理表单 12](#_Toc2221)

[2.5 在新版本的 PHP 中使用旧的 PHP 代码 13](#_Toc32307)

[2.6 下一步做什么？ 14](#_Toc31953)

[原文（PHP Manual） 14](#_Toc2389)

[1 Introduction 14](#_Toc6368)

[1.1 What is PHP? 14](#_Toc8298)

[1.2 What can PHP do? 15](#_Toc9976)

[2 A simple tutorial 18](#_Toc15549)

[2.1 What do I need? 18](#_Toc15445)

[2.2 Your first PHP-enabled page 18](#_Toc838)

[2.3 Something Useful 21](#_Toc1014)

[2.4 Dealing with Forms 24](#_Toc19790)

[2.5 Using old code with new versions of PHP 25](#_Toc27450)

[2.6 What's next? 26](#_Toc20236)

# **1 简介**

## **1.1 PHP 是什么？**

PHP（“PHP: Hypertext Preprocessor”，超文本预处理器的字母缩写）是一种被广泛应用的开放源代码的多用途脚本语言，它可嵌入到 HTML中，尤其适合 web 开发。

以上是一个简单的回答，不过这是什么意思呢？请看如下例子：

Example #1 一个介绍性的范例

<html>  
    <head>  
        <title>Example</title>  
    </head>  
    <body>  
  
        <?php  
        echo "Hi, I'm a PHP script!";  
        ?>  
  
    </body>  
</html>

请注意这个范例和其它用 C 或 Perl 语言写的脚本之间的区别——与用大量的命令来编写程序以输出 HTML 不同的是，PHP 页面就是 HTML，只不过在其中嵌入了一些代码来做一些事情（在本例中输出了 "Hi, I'm a PHP script!"）。PHP 代码被包含在特殊的[起始符和结束符 <?php 和 ?>](http://php.net/manual/zh/language.basic-syntax.phpmode.php) 中，使得可以进出“PHP 模式”。

和客户端的 JavaScript 不同的是，PHP 代码是运行在服务端的。如果在服务器上建立了如上例类似的代码，则在运行该脚本后，客户端就能接收到其结果，但他们无法得知其背后的代码是如何运作的。甚至可以将 web 服务器设置成让 PHP 来处理所有的 HTML 文件，这么一来，用户就无法得知服务端到底做了什么。

使用 PHP 的一大好处是它对于初学者来说极其简单，同时也给专业的程序员提供了各种高级的特性。当看到 PHP 长长的特性列表时，请不要害怕。可以很快的入门，只需几个小时就可以自己写一些简单的脚本。

尽管 PHP 的开发是以服务端脚本为目的，但事实上其功能远不局限与此。请继续读后面的章节，在“[PHP 能做什么](http://php.net/manual/zh/intro-whatcando.php)”一节中将获得更多的信息。如果对 web 编程感兴趣，也可以阅读[简明教程](http://php.net/manual/zh/tutorial.php)。

1.2 PHP 能做什么？

PHP 能做任何事。PHP 主要是用于服务端的脚本程序，因此可以用 PHP 来完成任何其它的 CGI 程序能够完成的工作，例如收集表单数据，生成动态网页，或者发送／接收 Cookies。但 PHP 的功能远不局限于此。

PHP 脚本主要用于以下三个领域：

* 服务端脚本。这是 PHP 最传统，也是最主要的目标领域。开展这项工作需要具备以下三点：PHP 解析器（CGI 或者服务器模块）、web 服务器和 web 浏览器。需要在运行 web 服务器时，安装并配置 PHP，然后，可以用 web 浏览器来访问 PHP 程序的输出，即浏览服务端的 PHP 页面。如果只是实验 PHP 编程，所有的这些都可以运行在自己家里的电脑中。请查阅[安装](http://php.net/manual/zh/install.php)一章以获取更多信息。
* 命令行脚本。可以编写一段 PHP 脚本，并且不需要任何服务器或者浏览器来运行它。通过这种方式，仅仅只需要 PHP 解析器来执行。这种用法对于依赖 cron（Unix 或者 Linux 环境）或者 Task Scheduler（Windows 环境）的日常运行的脚本来说是理想的选择。这些脚本也可以用来处理简单的文本。请参阅 [PHP 的命令行模式](http://php.net/manual/zh/features.commandline.php)以获取更多信息。
* 编写桌面应用程序。对于有着图形界面的桌面应用程序来说，PHP 或许不是一种最好的语言，但是如果用户非常精通 PHP，并且希望在客户端应用程序中使用 PHP 的一些高级特性，可以利用 PHP-GTK 来编写这些程序。用这种方法，还可以编写跨平台的应用程序。PHP-GTK 是 PHP 的一个扩展，在通常发布的 PHP 包中并不包含它。如果对 PHP-GTK 感兴趣，请访问其[» 网站](http://gtk.php.net/)以获取更多信息。

PHP 能够在所有的主流操作系统上[使用](http://php.net/manual/zh/install.php)，包括 Linux、Unix 的各种变种（包括 HP-UX、Solaris 和 OpenBSD）、Microsoft Windows、Mac OS X、RISC OS 等。今天，PHP已经支持了大多数的 web 服务器，包括 Apache、Microsoft Internet Information Server（IIS）、Personal Web Server（PWS）、Netscape 以及 iPlant server、Oreilly Website Pro Server、Caudium、Xitami、OmniHTTPd 等。对于大多数的服务器，PHP 提供了一个模块；还有一些 PHP 支持 CGI 标准，使得 PHP 能够作为 CGI 处理器来工作。

综上所述，使用 PHP，可以自由地选择操作系统和 web 服务器。同时，还可以在开发时选择使用面对过程和面对对象，或者两者混和的方式来开发。尽管 PHP 4 不支持 OOP 所有的标准，但很多代码仓库和大型的应用程序（包括 PEAR 库）仅使用 OOP 代码来开发。PHP 5 弥补了 PHP 4 的这一弱点，引入了完全的对象模型。

使用 PHP，并不局限于输出 HTML。PHP 还能被用来动态输出图像、PDF 文件甚至 Flash 动画（使用 libswf 和 Ming）。还能够非常简便的输出文本，例如 XHTML 以及任何其它形式的 XML 文件。PHP 能够自动生成这些文件，在服务端开辟出一块动态内容的缓存，可以直接把它们打印出来，或者将它们存储到文件系统中。

PHP 最强大最显著的特性之一，是它支持[很大范围的数据库](http://php.net/manual/zh/refs.database.php)。使用任何针对某数据库的扩展（例如 [mysql](http://php.net/manual/zh/book.mysqli.php)）编写数据库支持的网页非常简单，或者使用抽象层如 [PDO](http://php.net/manual/zh/book.pdo.php)，或者通过 [ODBC](http://php.net/manual/zh/book.uodbc.php) 扩展连接到任何支持 ODBC 标准的数据库。其它一些数据库也可能会用 [cURL](http://php.net/manual/zh/book.curl.php) 或者 [sockets](http://php.net/manual/zh/book.sockets.php)，例如 CouchDB。

PHP 还支持利用诸如 LDAP、IMAP、SNMP、NNTP、POP3、HTTP、COM（Windows 环境）等不计其数的协议的服务。还可以开放原始网络端口，使得任何其它的协议能够协同工作。PHP 支持和所有 web 开发语言之间的 WDDX 复杂数据交换。关于相互连接，PHP 已经支持了对 Java 对象的即时连接，并且可以透明地将其用作 PHP 对象。

PHP 具有极其有效的[文本处理](http://php.net/manual/zh/refs.basic.text.php)特性，包括 Perl 兼容正则表达式（[PCRE](http://php.net/manual/zh/book.pcre.php)）以及许多扩展和工具可用于[解析和访问 XML 文档](http://php.net/manual/zh/refs.xml.php)。PHP 将所有的 XML 功能标准化于坚实的 [libxml2](http://php.net/manual/zh/book.libxml.php) 扩展，并且还增加了 [SimpleXML](http://php.net/manual/zh/book.simplexml.php)，[XMLReader](http://php.net/manual/zh/book.xmlreader.php) 以及[XMLWriter](http://php.net/manual/zh/book.xmlwriter.php) 支持以扩充其功能。

另外，还有很多其它有趣的扩展库，在此根据[字母](http://php.net/manual/zh/extensions.php)和[分类](http://php.net/manual/zh/funcref.php)归类列出。还有一些附加的 PECL 扩展可能有也可能没有在 PHP 手册中列出，例如 [» XDebug](http://xdebug.org/)。

由于在这里无法列出 PHP 所有的特性和可提供的便利，请参阅[安装](http://php.net/manual/zh/install.php)以及[函数参考](http://php.net/manual/zh/funcref.php)有关章节以获取关于这里提到的扩展库更多的信息。

# **2 简明教程**

## **2.1 需要些什么？**

在本教程中，假设用户的服务器已经安装并运行了 PHP，所有以 .php 结尾的文件都将由 PHP 来处理。在大部分的服务器上， 这是 PHP 的默认扩展名，不过，也请询问服务器管理员以确认。如果服务器支持 PHP ，则不需要做任何事情。只用建立 .php 文件，并把它们放置 到 web 目录中，服务器将神奇地自动解析这些文件。不用编译任何东西，也不用安装 任何其它的工具，仅仅只需把这些使用了 PHP 的文件想象成简单的 HTML 文件，其中 只不过多了一种新的标识符，在这里可以做各种各样的事情。

假设用户希望在本地机器开发以节约宝贵的带宽。在这种情况下，需要安装一个诸如 [» Apache](http://httpd.apache.org/) 的 web 服务器，当然还有 [» PHP](http://www.php.net/downloads.php)。可能还希望安装一个数据库，例如 [» MySQL](http://dev.mysql.com/doc/)。

可以一个个的安装它们或选择一个更简单的方法。可以参考本手册中 [PHP 安装说明](http://php.net/manual/zh/install.php)的相关章节（假设已经配置好了某个 web 服务器）。若在自己安装 PHP 时出现了问题，建议在[» 安装邮件列表](http://www.php.net/mailing-lists.php)中询问。如果想使用更简便的方法安装 PHP，那么可以考虑[» 获取一个预先配置的安装包](http://wikipedia.org/wiki/List_of_AMP_packages)，用这个安装包，只用点击几下鼠标，就可以自动地安装所有这些系统。在任何操作系统下建立有 PHP 支持的 web 服务器都十分简单，包括 MacOSX、Linux 和 Windows。在 Linux 下，会发现 [» rpmfind](http://www.rpmfind.net/) 和 [» PBone](http://rpm.pbone.net/) 能够在获取 RPM 时提供帮助。也可以使用 [» apt-get](https://packages.debian.org/index) 搜索 Debian 的相关软件包。

2.2 第一个 PHP 页面

在 web 服务器根目录（DOCUMENT\_ROOT）下建立一个文件名为 hello.php，然后完成如下内容：

Example #1 第一个 PHP 脚本：hello.php

<html>  
 <head>  
  <title>PHP 测试</title>  
 </head>  
 <body>  
 <?php echo '<p>Hello World</p>'; ?>  
 </body>  
</html>

在浏览器的地址栏里输入 web 服务器的 URL 访问这个文件，在结尾加上“/hello.php”。如果本地开发，那么这个 URL 一般是 http://localhost/hello.php 或者 http://127.0.0.1/hello.php，当然这取决于 web 服务器的设置。如果所有的设置都正确，那么这个文件将被 PHP 解析，浏览器中将会输出如下结果：

<html>

<head>

<title>PHP 测试</title>

</head>

<body>

<p>Hello World</p>

</body>

</html>

该程序非常的简单，它仅仅只是利用了 PHP 的 [echo](http://php.net/manual/zh/function.echo.php) 语句显示了 Hello World。用户一定不会满足与此。请注意该文件无需被执行或以任何方式指定。服务器会找到该文件并提供给 PHP 进行解释，因为使用了“.php”的扩展名，服务器已被配置成自动传递有着“.php”扩展名的文件给 PHP。一个普通的 HTML 文件，加上了几个特别的标签，就可以做很多非常有趣的事情！

如果试过了这个例子，但是没有得到任何输出，或者浏览器弹出了下载框，或者浏览器以文本方式显示了源文件，可能的原因是服务器还没有支持 PHP，或者没有正确配置。需要请服务器的管理员根据本手册“[安装](http://php.net/manual/zh/install.php)”一章的内容使得服务器支持 PHP。如果本地开发，请阅读手册有关安装的章节以确保所有的设置都正确。还要确认通过浏览器访问的 URL 确实指向了服务器上的这个文件。如果只是从本地文件系统调用这个文件，它不会被 PHP 解析。如果问题仍然存在，请通过 [» PHP 在线支持](http://www.php.net/support.php)中的各种方式获取帮助。

以上例子的目的是为了显示 PHP 特殊标识符的格式。在这个例子中，用 <?php 来表示 PHP 标识符的起始，然后放入 PHP 语句并通过加上一个终止标识符 ?> 来退出 PHP 模式。可以根据自己的需要在 HTML 文件中像这样开启或关闭 PHP 模式。请参阅手册中“[PHP 基本语法](http://php.net/manual/zh/language.basic-syntax.php)”以获取更多信息。

Note: 关于换行

尽管换行在 HTML 中的实际意义不是很大，但适当地使用换行可以使 HTML 代码易读且美观。PHP 会在输出时自动删除其结束符 ?> 后的一个换行。该功能主要是针对在一个页面中嵌入多段 PHP 代码或者包含了无实质性输出的 PHP 文件而设计，与此同时也造成了一些疑惑。如果需要在 PHP 结束符 ?> 之后输出换行的话，可以在其后加一个空格，或者在最后的一个 echo/print 语句中加入一个换行。

Note: 关于文本编辑器

有很多文本编辑器以及集成开发环境（IDE）可以被用来建立、编辑和管理 PHP 文件。这些工具中的一部分被列在[» PHP 编辑器列表](http://en.wikipedia.org/wiki/List_of_PHP_editors)中。如果希望推荐其它的编辑器，请访问以上页面，并要求该页面的维护者将你推荐的编辑器加入到该列表中。使用支持语法高亮功能的编辑器会给开发带来很多帮助。

Note: 关于文字处理器

诸如 StarOffice Writer、Microsoft Word 和 Abiword 的文字处理器不适合用来编辑 PHP 程序。如果希望用以上这些工具的某一种来处理脚本，必须保证将结果存成了纯文本格式，否则 PHP 将无法读取并运行这些脚本。

Note: 关于 Windows 记事本

如果使用 Windows 记事本来编写 PHP 脚本，需要注意在保存文件时，文件的后缀名应该为 .php（记事本将自动在文件名后面加上 .txt 后缀，除非采取以下措施之一来避免这种情况）。当保存文件时，系统会让你指定文件的文件名，这时请将文件名加上引号（例如 "hello.php"）。或者，也可以点击“另存为”对话框中的“保存类型”下拉菜单，并将设置改为“所有文件”。这样在输入文件名的时候就不用加引号了。

现在已经成功建立了一个简单的 PHP 脚本，那么再来建立一个最著名的 PHP 脚本！调用函数 [phpinfo()](http://php.net/manual/zh/function.phpinfo.php)，将会看到很多有关自己系统的有用信息，例如[预定义变量](http://php.net/manual/zh/language.variables.predefined.php)、已经加载的 PHP 模块和[配置](http://php.net/manual/zh/configuration.php)信息。请花一些时间来查看这些重要的信息。

Example #2 从 PHP 获取系统信息

<?php phpinfo(); ?>

2.3 实用的脚本

现在来编写一些更实用的脚本，比如检查浏览页面的访问者在用什么浏览器。要达到这个目的，需要检查用户的 agent 字符串，它是浏览器发送的 HTTP 请求的一部分。该信息被存储在一个[变量](http://php.net/manual/zh/language.variables.php)中。在 PHP 中，变量总是以一个美元符开头。我们现在感兴趣的变量是 [$\_SERVER['HTTP\_USER\_AGENT']](http://php.net/manual/zh/reserved.variables.server.php)。

Note:

[$\_SERVER](http://php.net/manual/zh/reserved.variables.server.php) 是一个特殊的 PHP 保留变量，它包含了 web 服务器提供的所有信息，被称为超全局变量。请查阅本手册“[超全局变量](http://php.net/manual/zh/language.variables.superglobals.php)”中的有关内容以获取更多信息。这些特殊的变量是在 PHP [» 4.1.0](http://www.php.net/releases/4_1_0.php) 版本引入的。在这之前使用$HTTP\_\*\_VARS 数组，如 $HTTP\_SERVER\_VARS。尽管现在已经不用了，但它们在新版本中仍然存在（参见“[旧代码](http://php.net/manual/zh/tutorial.oldcode.php)”一节中的注解）。

要显示该变量，只需简单地进行如下操作：

Example #1 打印一个变量（数组元素）

<?php   
echo $\_SERVER['HTTP\_USER\_AGENT'];   
?>

该脚本的输出可能是：

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

PHP 有很多种不同[类型](http://php.net/manual/zh/language.types.php)的变量。在以上例子中我们打印了一个[数组](http://php.net/manual/zh/language.types.array.php)的单元。数组是一类非常有用的变量。

[$\_SERVER](http://php.net/manual/zh/reserved.variables.server.php) 只是 PHP 自动全局化的变量之一。可以查阅“[预定义变量](http://php.net/manual/zh/reserved.variables.php)”一节来查看这些变量的列表，或者也可以通过上节例子中 [phpinfo()](http://php.net/manual/zh/function.phpinfo.php) 函数的输出来查看。

可以在一个 PHP 标识中加入多个 PHP 语句，也可以建立一个代码块来做比简单的 echo 更多的事情。例如，如果需要识别 Internet Explorer，可以进行如下操作：

Example #2 [流程控制](http://php.net/manual/zh/language.control-structures.php)与[函数](http://php.net/manual/zh/language.functions.php)的使用

<?php  
if (strpos($\_SERVER['HTTP\_USER\_AGENT'], 'MSIE') !== FALSE) {  
    echo '正在使用 Internet Explorer。<br />';  
}  
?>

该脚本的输出可能是：

正在使用 Internet Explorer。<br />

这里要介绍一些新的原理。上面用了一个 [if](http://php.net/manual/zh/control-structures.if.php) 语句。如果用户对 C 语言的基本语法比较熟悉，则应该对此很熟悉，否则，可能需要拿起任何一本 PHP 介绍性的书籍并阅读前面的两三个章节，或者也可以阅读本手册的“[语言参考](http://php.net/manual/zh/langref.php)”一章。

需要介绍的第二个原理，是对 [strpos()](http://php.net/manual/zh/function.strpos.php) 函数的调用。[strpos()](http://php.net/manual/zh/function.strpos.php) 是 PHP 的一个内置函数，其功能是在一个字符串中搜索另外一个字符串。例如我们现在需要在 [$\_SERVER['HTTP\_USER\_AGENT']](http://php.net/manual/zh/reserved.variables.server.php)（即所谓的 haystack）变量中寻找 'MSIE'。如果在这个 haystack 中该字符串（即所谓的 needle）被找到（“草里寻针”），则函数返回 needle 在 haystack 中相对于开头的位置；如果没有，则返回 FALSE。如果该函数没有返回 FALSE，则 [if](http://php.net/manual/zh/control-structures.if.php) 会将条件判断为 TRUE 并运行其花括号 {} 内的代码；否则，则不运行这些代码。可以自己尝试利用 [if](http://php.net/manual/zh/control-structures.if.php)，[else](http://php.net/manual/zh/control-structures.else.php) 以及其它的函数如 [strtoupper()](http://php.net/manual/zh/function.strtoupper.php) 和 [strlen()](http://php.net/manual/zh/function.strlen.php) 来建立类似的脚本。在本手册中相关的页面也包含有范例。如果对如何使用函数不是很确定，可以阅读手册中有关“[如何阅读函数的定义](http://php.net/manual/zh/about.prototypes.php)”和“[函数](http://php.net/manual/zh/language.functions.php)”的有关章节。

以下我们进一步显示如何进出 PHP 模式，甚至是在一个 PHP 代码块的中间：

Example #3 混和 HTML 和 PHP 模式

<?php  
if (strpos($\_SERVER['HTTP\_USER\_AGENT'], 'MSIE') !== FALSE) {  
?>  
<h3>strpos() 肯定没有返回假 (FALSE)</h3>  
<p>正在使用 Internet Explorer</p>  
<?php  
} else {  
?>  
<h3>strpos() 肯定返回假 (FALSE)</h3>  
<center><b>没有使用 Internet Explorer</b></center>  
<?php  
}  
?>

该脚本的输出可能是：

<h3>strpos() 肯定没有返回假 (FALSE)</h3>

<p>正在使用 Internet Explorer</p>

和以上我们用一个 PHP 的 echo 语句来输出不同的是，我们跳出了 PHP 模式来直接写 HTML 代码。这里很值得注意的一点是，对于这两种情况而言，脚本的逻辑效率是相同的。在判断了 [strpos()](http://php.net/manual/zh/function.strpos.php) 函数的返回值是 TRUE 或是 FALSE，也就是判断了字符串 'MSIE' 是否被找到之后，最终只有一个 HTML 块被发送给浏览者。

2.4 处理表单

PHP 一个很有用的特点体现在它处理 PHP 表单的方式。需要理解的非常重要的原理，是表单的任何元素都在 PHP 脚本中自动生效。请参阅本手册中“[PHP 的外部变量](http://php.net/manual/zh/language.variables.external.php)”以获取关于在 PHP 中使用表单的详细信息及范例。以下是 HTML 表单的范例：

Example #1 一个简单的 HTML 表单

<form action="action.php" method="post">

<p>姓名: <input type="text" name="name" /></p>

<p>年龄: <input type="text" name="age" /></p>

<p><input type="submit" /></p>

</form>

该表单中并没有什么特殊的地方，其中没有使用任何特殊的标识符。当用户填写了该表单并点击了提交按钮，页面action.php 将被调用。在该文件中，可以加入如下内容：

Example #2 打印来自表单的数据

你好，<?php echo htmlspecialchars($\_POST['name']); ?>。  
你 <?php echo (int)$\_POST['age']; ?> 岁了。

该脚本的输出可能是：

你好，Joe。你 22 岁了。

除了[htmlspecialchars()](http://php.net/manual/zh/function.htmlspecialchars.php) 和 (int) 部分，这段程序做什么用显而易见。[htmlspecialchars()](http://php.net/manual/zh/function.htmlspecialchars.php) 使得 HTML 之中的特殊字符被正确的编码，从而不会被使用者在页面注入 HTML 标签或者 Javascript 代码。例如 age 字段，我们明确知道他是一个数值，因此我们将它[转换](http://php.net/manual/zh/language.types.type-juggling.php" \l "language.types.typecasting)为一个整形值(integer)来自动的消除任何不必要的字符。也可以使用 PHP 的 [filter](http://php.net/manual/zh/ref.filter.php) 扩展来自动完成该工作。PHP 将自动设置 [$\_POST['name']](http://php.net/manual/zh/reserved.variables.post.php) 和 [$\_POST['age']](http://php.net/manual/zh/reserved.variables.post.php) 变量。在这之前我们使用了超全局变量[$\_SERVER](http://php.net/manual/zh/reserved.variables.server.php)，现在我们引入了超全局变量 [$\_POST](http://php.net/manual/zh/reserved.variables.post.php)，它包含了所有的 POST 数据。请注意我们的表单提交数据的方法（method）。如果使用了 GET 方法，那么表单中的信息将被储存到超全局变量 [$\_GET](http://php.net/manual/zh/reserved.variables.get.php) 中。如果并不关心请求数据的来源，也可以用超全局变量 [$\_REQUEST](http://php.net/manual/zh/reserved.variables.request.php)，它包含了所有 GET、POST、COOKIE 和 FILE 的数据。

也可以在 PHP 中处理 XForms 的输入，尽管用户可能更喜欢使用长久以来支持良好的 HTML 表单。XForms 目前还不适合初学者使用，但是用户可能对它感兴趣。手册中在“特点”一章有一节对如何[处理从 XForum 接收到的数据](http://php.net/manual/zh/features.xforms.php)进行了简短的介绍。

2.5 在新版本的 PHP 中使用旧的 PHP 代码

现在，PHP 已经发展成为一种流行的脚本语言，可以在很多公共的资源里找到可以在自己的脚本中重新利用的代码。PHP 语言的开发者为向下兼容性下了很多功夫，因此在新版本的 PHP 下，老版本的代码应该可以在不作任何改动的情况下（理想地）运行。不过实际上，还是必须对老的代码做一些改动。

有可能影响到老版本的代码的最重要的两点改动分别是：

* 旧的 $HTTP\_\*\_VARS 数组从 PHP 5.4.0 开始将不再有效。 PHP [» 4.1.0](http://www.php.net/releases/4_1_0.php) 版本引入了如下[超全局数组变量](http://php.net/manual/zh/language.variables.superglobals.php)：[$\_GET](http://php.net/manual/zh/reserved.variables.get.php)、[$\_POST](http://php.net/manual/zh/reserved.variables.post.php)、[$\_COOKIE](http://php.net/manual/zh/reserved.variables.cookies.php)、 [$\_SERVER](http://php.net/manual/zh/reserved.variables.server.php)、[$\_FILES](http://php.net/manual/zh/reserved.variables.files.php)、[$\_ENV](http://php.net/manual/zh/reserved.variables.environment.php)、 [$\_REQUEST](http://php.net/manual/zh/reserved.variables.request.php) 以及 [$\_SESSION](http://php.net/manual/zh/reserved.variables.session.php)。
* 外部变量不再被默认注册为全局变量。也就是说，从 PHP [» 4.2.0](http://www.php.net/releases/4_2_0.php) 版开始，php.ini 中的设置选项 [register\_globals](http://php.net/manual/zh/ini.core.php" \l "ini.register-globals)默认值变成了 off。建议用以上提到的超全局数组变量来访问这些值。但可能老的脚本、书籍以及教程都可能建立在该设置为 on 的基础上。如果该选项被设置为 on，则可以在 URL http://www.example.com/foo.php?id=42 中直接使用变量 $id。但不管被设置为 on 还是 off，[$\_GET['id']](http://php.net/manual/zh/reserved.variables.get.php) 一直有效。

如果希望了解关于这些改动的细节，请参阅“[预定义变量](http://php.net/manual/zh/language.variables.predefined.php)”一章以及其中的连接。

## **2.6 下一步做什么？**

用现在掌握的知识，应该能够理解本手册中的大部分内容以及其中各式各样的脚本范例。

请查阅 PHP Conference 资料网站 [» http://talks.php.net/](http://talks.php.net/) 以观看更多幻灯片，这些幻灯片展示了许多 PHP 其它的功能。

# 原文（PHP Manual）

**1 Introduction**

## **1.1 What is PHP?**

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

Nice, but what does that mean? An example:

Example #1 An introductory example

<!DOCTYPE HTML>  
<html>  
    <head>  
        <title>Example</title>  
    </head>  
    <body>  
  
        <?php  
            echo "Hi, I'm a PHP script!";  
        ?>  
  
    </body>  
</html>

Instead of lots of commands to output HTML (as seen in C or Perl), PHP pages contain HTML with embedded code that does "something" (in this case, output "Hi, I'm a PHP script!"). The PHP code is enclosed in special [start and end processing instructions <?php and ?>](http://php.net/manual/en/language.basic-syntax.phpmode.php) that allow you to jump into and out of "PHP mode."

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

The best things in using PHP are that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer. Don't be afraid reading the long list of PHP's features. You can jump in, in a short time, and start writing simple scripts in a few hours.

Although PHP's development is focused on server-side scripting, you can do much more with it. Read on, and see more in the [What can PHP do?](http://php.net/manual/en/intro-whatcando.php) section, or go right to the [introductory tutorial](http://php.net/manual/en/tutorial.php) if you are only interested in web programming.

## **1.2 What can PHP do?**

Anything. PHP is mainly focused on server-side scripting, so you can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies. But PHP can do much more.

There are three main areas where PHP scripts are used.

Server-side scripting. This is the most traditional and main target field for PHP. You need three things to make this work. The PHP parser (CGI or server module), a web server and a web browser. You need to run the web server, with a connected PHP installation. You can access the PHP program output with a web browser, viewing the PHP page through the server. All these can run on your home machine if you are just experimenting with PHP programming. See the [installation instructions](http://php.net/manual/en/install.php) section for more information.

Command line scripting. You can make a PHP script to run it without any server or browser. You only need the PHP parser to use it this way. This type of usage is ideal for scripts regularly executed using cron (on \*nix or Linux) or Task Scheduler (on Windows). These scripts can also be used for simple text processing tasks. See the section about [Command line usage of PHP](http://php.net/manual/en/features.commandline.php) for more information.

Writing desktop applications. PHP is probably not the very best language to create a desktop application with a graphical user interface, but if you know PHP very well, and would like to use some advanced PHP features in your client-side applications you can also use PHP-GTK to write such programs. You also have the ability to write cross-platform applications this way. PHP-GTK is an extension to PHP, not available in the main distribution. If you are interested in PHP-GTK, visit [» its own website](http://gtk.php.net/).

PHP can be [used](http://php.net/manual/en/install.php) on all major operating systems, including Linux, many Unix variants (including HP-UX, Solaris and OpenBSD), Microsoft Windows, Mac OS X, RISC OS, and probably others. PHP has also support for most of the web servers today. This includes Apache, IIS, and many others. And this includes any web server that can utilize the FastCGI PHP binary, like lighttpd and nginx. PHP works as either a module, or as a CGI processor.

So with PHP, you have the freedom of choosing an operating system and a web server. Furthermore, you also have the choice of using procedural programming or object oriented programming (OOP), or a mixture of them both.

With PHP you are not limited to output HTML. PHP's abilities includes outputting images, PDF files and even Flash movies (using libswf and Ming) generated on the fly. You can also output easily any text, such as XHTML and any other XML file. PHP can autogenerate these files, and save them in the file system, instead of printing it out, forming a server-side cache for your dynamic content.

One of the strongest and most significant features in PHP is its support for a [wide range of databases](http://php.net/manual/en/refs.database.php). Writing a database-enabled web page is incredibly simple using one of the database specific extensions (e.g., for [mysql](http://php.net/manual/en/book.mysqli.php)), or using an abstraction layer like [PDO](http://php.net/manual/en/book.pdo.php), or connect to any database supporting the Open Database Connection standard via the [ODBC](http://php.net/manual/en/book.uodbc.php) extension. Other databases may utilize [cURL](http://php.net/manual/en/book.curl.php) or [sockets](http://php.net/manual/en/book.sockets.php), like CouchDB.

PHP also has support for talking to other services using protocols such as LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (on Windows) and countless others. You can also open raw network sockets and interact using any other protocol. PHP has support for the WDDX complex data exchange between virtually all Web programming languages. Talking about interconnection, PHP has support for instantiation of Java objects and using them transparently as PHP objects.

PHP has useful [text processing](http://php.net/manual/en/refs.basic.text.php) features, which includes the Perl compatible regular expressions ([PCRE](http://php.net/manual/en/book.pcre.php)), and many extensions and tools to [parse and access XML documents](http://php.net/manual/en/refs.xml.php). PHP standardizes all of the XML extensions on the solid base of [libxml2](http://php.net/manual/en/book.libxml.php), and extends the feature set adding [SimpleXML](http://php.net/manual/en/book.simplexml.php), [XMLReader](http://php.net/manual/en/book.xmlreader.php) and [XMLWriter](http://php.net/manual/en/book.xmlwriter.php) support.

And many other interesting extensions exist, which are categorized both [alphabetically](http://php.net/manual/en/extensions.php) and by [category](http://php.net/manual/en/funcref.php). And there are additional PECL extensions that may or may not be documented within the PHP manual itself, like[» XDebug](http://xdebug.org/).

As you can see this page is not enough to list all the features and benefits PHP can offer. Read on in the sections about [installing PHP](http://php.net/manual/en/install.php), and see the [function reference](http://php.net/manual/en/funcref.php) part for explanation of the extensions mentioned here.

**2 A simple tutorial**

**2.1 What do I need?**

In this tutorial we assume that your server has activated support for PHP and that all files ending in .php are handled by PHP. On most servers, this is the default extension for PHP files, but ask your server administrator to be sure. If your server supports PHP, then you do not need to do anything. Just create your .php files, put them in your web directory and the server will automatically parse them for you. There is no need to compile anything nor do you need to install any extra tools. Think of these PHP-enabled files as simple HTML files with a whole new family of magical tags that let you do all sorts of things.

Let us say you want to save precious bandwidth and develop locally. In this case, you will want to install a web server, such as [» Apache](http://httpd.apache.org/), and of course [» PHP](http://www.php.net/downloads.php). You will most likely want to install a database as well, such as[» MySQL](http://dev.mysql.com/doc/).

You can either install these individually or choose a simpler way. Our manual has [installation instructions for PHP](http://php.net/manual/en/install.php) (assuming you already have some web server set up). If you have problems with installing PHP yourself, we would suggest you ask your questions on our [» installation mailing list](http://www.php.net/mailing-lists.php). If you choose to go on the simpler route, then [» locate a pre-configured package](http://wikipedia.org/wiki/List_of_AMP_packages) for your operating system, which automatically installs all of these with just a few mouse clicks. It is easy to setup a web server with PHP support on any operating system, including MacOSX, Linux and Windows. On Linux, you may find [» rpmfind](http://www.rpmfind.net/) and [» PBone](http://rpm.pbone.net/) helpful for locating RPMs. You may also want to visit [» apt-get](https://packages.debian.org/index) to find packages for Debian.

**2.2 Your first PHP-enabled page**

Create a file named hello.php and put it in your web server's root directory (DOCUMENT\_ROOT) with the following content:

Example #1 Our first PHP script: hello.php

<html>  
 <head>  
  <title>PHP Test</title>  
 </head>  
 <body>  
 <?php echo '<p>Hello World</p>'; ?>   
 </body>  
</html>

Use your browser to access the file with your web server's URL, ending with the /hello.php file reference. When developing locally this URL will be something like http://localhost/hello.php or http://127.0.0.1/hello.php but this depends on the web server's configuration. If everything is configured correctly, this file will be parsed by PHP and the following output will be sent to your browser:

<html>

<head>

<title>PHP Test</title>

</head>

<body>

<p>Hello World</p>

</body>

</html>

This program is extremely simple and you really did not need to use PHP to create a page like this. All it does is display: Hello World using the PHP [echo](http://php.net/manual/en/function.echo.php) statement. Note that the file does not need to be executable or special in any way. The server finds out that this file needs to be interpreted by PHP because you used the ".php" extension, which the server is configured to pass on to PHP. Think of this as a normal HTML file which happens to have a set of special tags available to you that do a lot of interesting things.

If you tried this example and it did not output anything, it prompted for download, or you see the whole file as text, chances are that the server you are on does not have PHP enabled, or is not configured properly. Ask your administrator to enable it for you using the [Installation](http://php.net/manual/en/install.php) chapter of the manual. If you are developing locally, also read the installation chapter to make sure everything is configured properly. Make sure that you access the file via http with the server providing you the output. If you just call up the file from your file system, then it will not be parsed by PHP. If the problems persist anyway, do not hesitate to use one of the many [» PHP support](http://www.php.net/support.php) options.

The point of the example is to show the special PHP tag format. In this example we used <?php to indicate the start of a PHP tag. Then we put the PHP statement and left PHP mode by adding the closing tag, ?>. You may jump in and out of PHP mode in an HTML file like this anywhere you want. For more details, read the manual section on the [basic PHP syntax](http://php.net/manual/en/language.basic-syntax.php).

Note: A Note on Line Feeds

Line feeds have little meaning in HTML, however it is still a good idea to make your HTML look nice and clean by putting line feeds in. A linefeed that follows immediately after a closing ?> will be removed by PHP. This can be extremely useful when you are putting in many blocks of PHP or include files containing PHP that aren't supposed to output anything. At the same time it can be a bit confusing. You can put a space after the closing ?> to force a space and a line feed to be output, or you can put an explicit line feed in the last echo/print from within your PHP block.

Note: A Note on Text Editors

There are many text editors and Integrated Development Environments (IDEs) that you can use to create, edit and manage PHP files. A partial list of these tools is maintained at [» PHP Editors List](http://en.wikipedia.org/wiki/List_of_PHP_editors). If you wish to recommend an editor, please visit the above page and ask the page maintainer to add the editor to the list. Having an editor with syntax highlighting can be helpful.

Note: A Note on Word Processors

Word processors such as StarOffice Writer, Microsoft Word and Abiword are not optimal for editing PHP files. If you wish to use one for this test script, you must ensure that you save the file as plain text or PHP will not be able to read and execute the script.

Note: A Note on Windows Notepad

If you are writing your PHP scripts using Windows Notepad, you will need to ensure that your files are saved with the .php extension. (Notepad adds a .txt extension to files automatically unless you take one of the following steps to prevent it.) When you save the file and are prompted to provide a name for the file, place the filename in quotes (i.e. "hello.php"). Alternatively, you can click on the 'Text Documents' drop-down menu in the 'Save' dialog box and change the setting to "All Files". You can then enter your filename without quotes.

Now that you have successfully created a working PHP script, it is time to create the most famous PHP script! Make a call to the [phpinfo()](http://php.net/manual/en/function.phpinfo.php) function and you will see a lot of useful information about your system and setup such as available [predefined variables](http://php.net/manual/en/language.variables.predefined.php), loaded PHP modules, and [configuration](http://php.net/manual/en/configuration.php) settings. Take some time and review this important information.

Example #2 Get system information from PHP

<?php phpinfo(); ?>

**2.3 Something Useful**

Let us do something more useful now. We are going to check what sort of browser the visitor is using. For that, we check the user agent string the browser sends as part of the HTTP request. This information is stored in a[variable](http://php.net/manual/en/language.variables.php). Variables always start with a dollar-sign in PHP. The variable we are interested in right now is[$\_SERVER['HTTP\_USER\_AGENT']](http://php.net/manual/en/reserved.variables.server.php).

Note:

[$\_SERVER](http://php.net/manual/en/reserved.variables.server.php) is a special reserved PHP variable that contains all web server information. It is known as a superglobal. See the related manual page on [superglobals](http://php.net/manual/en/language.variables.superglobals.php) for more information. These special variables were introduced in PHP [» 4.1.0](http://www.php.net/releases/4_1_0.php). Before this time, we used the older $HTTP\_\*\_VARS arrays instead, such as$HTTP\_SERVER\_VARS. As of PHP 5.4.0 these older variables have been removed. (See also the note on [old code](http://php.net/manual/en/tutorial.oldcode.php).)

To display this variable, you can simply do:

Example #1 Printing a variable (Array element)

<?php  
echo $\_SERVER['HTTP\_USER\_AGENT'];  
?>

A sample output of this script may be:

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

There are many [types](http://php.net/manual/en/language.types.php) of variables available in PHP. In the above example we printed an [Array](http://php.net/manual/en/language.types.array.php) element. Arrays can be very useful.

[$\_SERVER](http://php.net/manual/en/reserved.variables.server.php) is just one variable that PHP automatically makes available to you. A list can be seen in the [Reserved Variables](http://php.net/manual/en/reserved.variables.php) section of the manual or you can get a complete list of them by looking at the output of the [phpinfo()](http://php.net/manual/en/function.phpinfo.php)function used in the example in the previous section.

You can put multiple PHP statements inside a PHP tag and create little blocks of code that do more than just a single echo. For example, if you want to check for Internet Explorer you can do this:

Example #2 Example using [control structures](http://php.net/manual/en/language.control-structures.php) and [functions](http://php.net/manual/en/language.functions.php)

<?php  
if (strpos($\_SERVER['HTTP\_USER\_AGENT'], 'MSIE') !== FALSE) {  
    echo 'You are using Internet Explorer.<br />';  
}  
?>

A sample output of this script may be:

You are using Internet Explorer.<br />

Here we introduce a couple of new concepts. We have an [if](http://php.net/manual/en/control-structures.if.php) statement. If you are familiar with the basic syntax used by the C language, this should look logical to you. Otherwise, you should probably pick up an introductory PHP book and read the first couple of chapters, or read the [Language Reference](http://php.net/manual/en/langref.php) part of the manual.

The second concept we introduced was the [strpos()](http://php.net/manual/en/function.strpos.php) function call. [strpos()](http://php.net/manual/en/function.strpos.php) is a function built into PHP which searches a string for another string. In this case we are looking for 'MSIE' (so-called needle) inside[$\_SERVER['HTTP\_USER\_AGENT']](http://php.net/manual/en/reserved.variables.server.php) (so-called haystack). If the needle is found inside the haystack, the function returns the position of the needle relative to the start of the haystack. Otherwise, it returns FALSE. If it does not returnFALSE, the [if](http://php.net/manual/en/control-structures.if.php) expression evaluates to TRUE and the code within its {braces} is executed. Otherwise, the code is not run. Feel free to create similar examples, with [if](http://php.net/manual/en/control-structures.if.php), [else](http://php.net/manual/en/control-structures.else.php), and other functions such as [strtoupper()](http://php.net/manual/en/function.strtoupper.php) and [strlen()](http://php.net/manual/en/function.strlen.php). Each related manual page contains examples too. If you are unsure how to use functions, you will want to read both the manual page on [how to read a function definition](http://php.net/manual/en/about.prototypes.php) and the section about [PHP functions](http://php.net/manual/en/language.functions.php).

We can take this a step further and show how you can jump in and out of PHP mode even in the middle of a PHP block:

Example #3 Mixing both HTML and PHP modes

<?php  
if (strpos($\_SERVER['HTTP\_USER\_AGENT'], 'MSIE') !== FALSE) {  
?>  
<h3>strpos() must have returned non-false</h3>  
<p>You are using Internet Explorer</p>  
<?php  
} else {  
?>  
<h3>strpos() must have returned false</h3>  
<p>You are not using Internet Explorer</p>  
<?php  
}  
?>

A sample output of this script may be:

<h3>strpos() must have returned non-false</h3>

<p>You are using Internet Explorer</p>

Instead of using a PHP echo statement to output something, we jumped out of PHP mode and just sent straight HTML. The important and powerful point to note here is that the logical flow of the script remains intact. Only one of the HTML blocks will end up getting sent to the viewer depending on the result of [strpos()](http://php.net/manual/en/function.strpos.php). In other words, it depends on whether the string MSIE was found or not.

**2.4 Dealing with Forms**

One of the most powerful features of PHP is the way it handles HTML forms. The basic concept that is important to understand is that any form element will automatically be available to your PHP scripts. Please read the manual section on [Variables from external sources](http://php.net/manual/en/language.variables.external.php) for more information and examples on using forms with PHP. Here is an example HTML form:

Example #1 A simple HTML form

<form action="action.php" method="post">

<p>Your name: <input type="text" name="name" /></p>

<p>Your age: <input type="text" name="age" /></p>

<p><input type="submit" /></p>

</form>

There is nothing special about this form. It is a straight HTML form with no special tags of any kind. When the user fills in this form and hits the submit button, the action.php page is called. In this file you would write something like this:

Example #2 Printing data from our form

Hi <?php echo htmlspecialchars($\_POST['name']); ?>.  
You are <?php echo (int)$\_POST['age']; ?> years old.

A sample output of this script may be:

Hi Joe. You are 22 years old.

Apart from the [htmlspecialchars()](http://php.net/manual/en/function.htmlspecialchars.php) and (int) parts, it should be obvious what this does. [htmlspecialchars()](http://php.net/manual/en/function.htmlspecialchars.php) makes sure any characters that are special in html are properly encoded so people can't inject HTML tags or Javascript into your page. For the age field, since we know it is a number, we can just [convert](http://php.net/manual/en/language.types.type-juggling.php" \l "language.types.typecasting) it to an [integer](http://php.net/manual/en/language.types.integer.php) which will automatically get rid of any stray characters. You can also have PHP do this for you automatically by using the[filter](http://php.net/manual/en/ref.filter.php) extension. The [$\_POST['name']](http://php.net/manual/en/reserved.variables.post.php) and [$\_POST['age']](http://php.net/manual/en/reserved.variables.post.php) variables are automatically set for you by PHP. Earlier we used the [$\_SERVER](http://php.net/manual/en/reserved.variables.server.php) superglobal; above we just introduced the [$\_POST](http://php.net/manual/en/reserved.variables.post.php) superglobal which contains all POST data. Notice how the method of our form is POST. If we used the method GET then our form information would live in the [$\_GET](http://php.net/manual/en/reserved.variables.get.php) superglobal instead. You may also use the [$\_REQUEST](http://php.net/manual/en/reserved.variables.request.php) superglobal, if you do not care about the source of your request data. It contains the merged information of GET, POST and COOKIE data.

You can also deal with XForms input in PHP, although you will find yourself comfortable with the well supported HTML forms for quite some time. While working with XForms is not for beginners, you might be interested in them. We also have a [short introduction to handling data received from XForms](http://php.net/manual/en/features.xforms.php) in our features section.

**2.5 Using old code with new versions of PHP**

Now that PHP has grown to be a popular scripting language, there are a lot of public repositories and libraries containing code you can reuse. The PHP developers have largely tried to preserve backwards compatibility, so a script written for an older version will run (ideally) without changes in a newer version of PHP. In practice, some changes will usually be needed.

Two of the most important recent changes that affect old code are:

The old $HTTP\_\*\_VARS arrays are not available as of PHP 5.4.0. The following [superglobal arrays](http://php.net/manual/en/language.variables.superglobals.php) were introduced in PHP [» 4.1.0](http://www.php.net/releases/4_1_0.php). They are: [$\_GET](http://php.net/manual/en/reserved.variables.get.php), [$\_POST](http://php.net/manual/en/reserved.variables.post.php), [$\_COOKIE](http://php.net/manual/en/reserved.variables.cookies.php), [$\_SERVER](http://php.net/manual/en/reserved.variables.server.php), [$\_FILES](http://php.net/manual/en/reserved.variables.files.php), [$\_ENV](http://php.net/manual/en/reserved.variables.environment.php), [$\_REQUEST](http://php.net/manual/en/reserved.variables.request.php), and[$\_SESSION](http://php.net/manual/en/reserved.variables.session.php).

External variables are no longer registered in the global scope by default. In other words, as of PHP [» 4.2.0](http://www.php.net/releases/4_2_0.php) the PHP directive [register\_globals](http://php.net/manual/en/ini.core.php" \l "ini.register-globals) is off by default in php.ini. The preferred method of accessing these values is via the superglobal arrays mentioned above. Older scripts, books, and tutorials may rely on this directive being on. If it were on, for example, one could use $id from the URL http://www.example.com/foo.php?id=42. Whether on or off, [$\_GET['id']](http://php.net/manual/en/reserved.variables.get.php) is available.

For more details on these changes, see the section on [predefined variables](http://php.net/manual/en/language.variables.predefined.php) and links therein.

**2.6 What's next?**

With your new knowledge you should be able to understand most of the manual and also the various example scripts available in the example archives.

To view various slide presentations that show more of what PHP can do, see the PHP Conference Material Site:[» http://talks.php.net/](http://talks.php.net/)