第五篇　创建实用的PHP和MySQL项目

第25章　在大型项目中使用PHP和MySQL

在本书的前几篇中，我们已经讨论了PHP和MySQL各个不同的组件及其应用。尽管我们试图列举出一些有趣且互相联系的例子，但实际上它们非常简单，只包含一两个脚本，每个脚本最多只有100行左右代码。

当创建真正的Web应用程序时，编写代码就不会这么简单了。几年前，“交互式”网站还处于表单邮件的阶段。如今，Web站点已经变成了Web应用程序——也就是说，符合一定规范的软件已经整个分布在Web网站。这种改变清楚地表明了网站规模的改变。网站已经从几行脚本代码发展到如今的成千上万行代码。如此规模的项目需要像一般软件开发那样进行规划和管理。

在我们开始介绍本书这部分将要涉及的项目之前，先浏览一下管理大型Web项目需要使用到的一些技术。这可以说是一门新兴艺术，要正确掌握它非常困难。这一点在软件市场中不难发现。

在本章中，我们将主要介绍以下内容：

- 在Web开发中应用软件工程
- 规划和运行Web应用程序项目
- 重用代码
- 编写可维护代码
- 实现版本控制
- 选择一个开发环境
- 项目的文档化
- 建立原型
- 分离逻辑、内容和外观：PHP、HTML和CSS
- 优化代码

25.1　在Web开发中应用软件工程

我们都知道，软件工程是软件开发过程中系统化的、可以量化的开发方法。也就是说，它是工程原则在软件开发中的应用。

很明显，软件工程在当今Web项目中的应用是很少的。这主要有两个原因：

首先，网站开发的管理经常与书面报告的管理方法相同。它是文档结构、图形设计和产品化的运用。它是一个面向文档的模式。这对中小型静态网站来说是很适宜的，也很有效。

但是随着网站动态内容的增加，网站的功能由原来简单地提供文档发展到提供服务，这样的模式就不再适应了。很多人根本就没有想到将软件工程实践应用到Web 项目中。

其次，软件项目之所以未能在Web项目中应用，还在于Web应用程序的开发与一般的应用程序的开发存在多方面的区别。对网站的设计，我们没有什么规划时间，但却面临很大压力，要求马上就要建好。对软件项目则要求有计划、有秩序地开发，并且会在规划上花费不少时间。因此，对网站设计来说，通常的感觉是没有时间规划。

当Web项目设计失败的时候，它出现的问题和任何软件设计失败时出现的问题是一样的：错误百出的应用程序、超过时间期限以及源代码晦涩难懂。

然而，关键是寻找适用于这种Web应用程序开发新规则的软件工程部分，而忽略非适用部分。

25.2　规划和运行Web应用程序项目

Web项目没有最好的设计方法或项目周期，但是，我们却必须为自己的项目仔细考虑一些问题。在这里，我们将给出这些问题，在后续内容中将详细讨论它们。这里列出的问题是按照一定顺序的，但是这个顺序如果不适合具体项目，可以不按照这个顺序。重要的是注意这些问题，选择适合具体项目的技术。

- 在项目开始之前，必须明确要创建什么，创建的最终目标是什么。要考虑谁将使用这个Web应用程序；也就是说，谁是目标用户。从技术上说，许多Web项目都是很完美的，但是最终失败了，因为没有人考察是否有人对他们的应用程序感兴趣。
- 要尝试将应用程序分成几个部分。应用程序有哪些部分或处理步骤？这些部分是如何工作的？它们之间能够相互协调吗？草拟其工作场景或举出一些例子，对于回答和解决这些问题是有效的。
- 列出各个部分之后，看看哪些部分已经存在了。如果一个以前完成的模块已经具有了该部分的功能，我们需要考虑是否可以直接使用它。不要忘了在公司内外查找现存的代码。特别是在源代码开放的社区，许多现存的代码组件是可以免费使用的。决定哪些代码必须得从头编写，并且大略估计一下工作量。
- 对工作进程问题作出决定。Web项目中这些问题太容易忽略。这里所说的工作进程问题，是指代码编写标准、目录结构、版本控制管理、开发环境、文档化级别和标准，以及对小组成员的任务分配。
- 基于所有已经获得的信息构建一个原型，展示给用户。并反复修改和展示。
- 请记住，对于所有这些，将应用程序的内容和逻辑分开是非常重要的。在后续的内容中，我们将详细解释这些观点。
- 对系统进行必要的优化。
- 像其他软件开发项目一样，一边开发，一边测试。

25.3　重用代码

通常，编程人员容易犯的一个错误是重写已经存在的代码。当知道需要什么应用程序组件或函数之后，应该在开发之前看看哪些组件和函数是可从别的地方获取的。

作为编程语言，PHP的一个优点就是它有大量内置的函数库。我们应该经常查看是否已经存在一个能够完成所需功能的函数。通常，找到所需的函数并不是很困难的。查找函数的一个好方法就是浏览按照函数组分类的手册。

有时候，编程人员重写函数是因为他们并没有查看手册，寻找是否有现存的函数已经提供了他们需要的功能。应该将手册页面作为书签保存在浏览器中，但是请注意，在线手册更新频繁。带注释的手册是一种非常好的资源，它包含了评注、建议和其他用户编写的示例代码。通常，这些资源还包含了问题报告和问题的解决方法。

如下站点提供了PHP手册的英文版本：http: //www.php.net/manual/an/。

一些有着不同语言背景的编程人员可能会编写一些封装函数，并且使用这些封装函数完全改变PHP原函数的名称以适应他们熟悉的语言。这种做法有时候也称为“*语法糖果*”。它是一个糟糕的主意，这将使得别人很难读懂这些代码，从而带来代码维护问题。如果正在学习一门新的语言，需要学会怎样正确使用它。另外，增加一层函数调用也会使代码执行速度下降。考虑到这些因素，这种做法应该尽量避免。

如果发现所需的功能在PHP主函数库里没有提供，那么有两个选择。如果需要的功能非常简单，可以选择自己编写该函数或对象。然而，如果需要创建一个相当复杂的功能——例如购物车、Web邮件系统或Web论坛——我们会发现这些东西别人已经做好了。在开放源代码社区工作的一个动力就是像这些应用程序组件的代码经常是免费的。如果发现某个组件与需要构建的相似甚至相同，那么就可将这些源代码作为基础，在此基础上修改或创建自己的组件。

如果已经开发完毕自己的函数或组件，应该认真考虑将这些函数或组件公布到PHP社区。只有大家都遵循这种原则，才能使PHP开发者社区继续是一个有帮助的、充满活力的、知识丰富的群体。

25.4　编写可维护代码

通常，代码的可维护性问题在Web应用中很容易被忽略，尤其是当网站创建得非常匆忙的时候。有时候，开始编写代码并快速完成它看起来比最初进行规划更重要。但是，在开始工作以前投入一点时间可以节省接下来的许多时间，当在编写的过程中重复编写应用程序的时候，就会发现这一点。

25.4.1　编码标准

大多数IT组织都制定了编码标准——选择文件名、变量名以形成内部编码准则、注释代码的准则、代码缩进的准则等。

因为文档模式经常应用于网站开发，因此在这里，代码标准经常会被忽略。对于独立编码或者在一个较小的团队里编码的人来说，容易低估代码标准化的重要性。当小组和项目不断发展的时候，若还不实行代码标准化的话，不但自己忙得手忙脚乱，而且许多编程人员也会对现存的代码摸不着头脑。

1. 定义命名惯例

定义一个命名规范的目的是为了：

■ 使代码易读。如果在定义变量和函数时使用的名称是非常直观的，就可以像阅读英语句子一样阅读代码，至少可以猜出它们的意思。

■ 使标识符易记。如果标识符格式统一，就更容易记住使用了什么变量，调用了什么函数。

变量名称应该能够描述其所包含的数据。如果使用变量表示一个人的姓氏，就可以给它取名$surname。也可以在变量名称的长度和可读性之间找一个平衡点。例如，将表示名字的变量取名为$n可以使其类型简单，但是这样的变量不具有良好的可读性。而如果取名为$surname_of_the_current_user，虽然可以提供详细的信息，但是对数据类型来说太长了（容易导致输入错误），并且不具有很大的价值。

还需要决定在什么时候使用大写。正如前面所提到的，变量名称在PHP中是区分大小写的。需要决定变量名称是否全部为小写、全部为大写或者两者混合，例如，将单词的第一个字母大写。我们倾向于全部使用小写，因为它们最容易记忆。

将常量和变量通过大小写区分是一个好主意——通常的模式是，变量都用小写表示（例如，$result），常量都用大写表示（例如，PI）。

一些编程人员喜欢将两个变量使用同一个名字，但是使用不同的大小写来区分，例如，$name和$Name，显而易见，这是非常糟糕的想法。

此外，最好还要避免一些可笑的大小写模式，例如，$WaReZ。没有人会记得它代表什么。

还要考虑多单词变量名称的命名模式。例如，我们已经见过如下所示的模式：

```
$username
$user_name
$UserName
```

以上都是我们曾经见到过的命名模式。选择哪一种规范都是没有问题的，但关键是要一直保持使用这种规范。另外，对于多单词变量名，一个名字应最多包含2～3个单词。

函数名称的命名规范需要考虑的因素与变量名称命名规范有许多相似之处，当然也有其自身的特点。通常，函数名是面向动词的。PHP内置的函数，例如，addslashes()或mysqli_connect()清楚地描述了它们的功能或它们要传递的参数。这大大提高了代码的可读性。请注意，这两个函数虽然都是多单词函数名，但是使用了不同的命名模式。从这点来说，PHP的命名是不一致的。原因之一可能就是因为参与的开发人员太多。但是最主要的原因是许多函数名称是直接从不同的语言和API拿过来的，而未经过改动。

此外，还要记住，在PHP中函数名称是不区分大小写的。我们应该一直坚持一种特定的格式，以避免造成混乱。

我们可能希望在许多PHP模块中使用模块命名模式，也就是说，在函数名前都加上模块名称作为前缀。例如，所有的MySQL函数都以mysqli_开头。所有的IMAP函数都以imap_开头。如果代码里有一个购物车模块，则可在该模块中的函数名前加上cart_。

然而，请注意，PHP5提供了过程接口和面向对象接口，在这两种接口中，函数名称是不同的。通常，过程式接口使用下划线（my_function()），而面向对象接口使用一种名为studlyCaps的模式（myFunction()）。

最后，编写代码的时候使用哪种命名惯例或标准对程序本身并没有多大的影响。只要遵循

和使用一致的方针。

2. 对代码进行注释

所有的程序都应该有一定程度的注释语句。那么使用何种级别的注释是适当的呢？通常，应该考虑为如下每个项目添加注释：

- **文件，无论是完整的脚本还是包含文件**——每个文件应该有一个注释，说明该文件的名称、功能、作者和更新日期。
- **函数**——函数的注释应指明函数的功能，输入参数和返回值。
- **类**——注释应描述类的用途。类方法应该具有与函数描述相同的注释。
- **脚本或函数里面的一大段代码**——在脚本之前添加一段伪码样式的注释然后再编写代码是很好的。因此，初始化脚本可能是这样：

```
<?
// validate input data
// send to database
// report results
?>
```

这种注释模式是非常方便的，因为在函数调用或其他代码编写之前，代码的注释已经做好了。

- **复杂的代码**——当整天围绕某件事，或者花费了很长时间来解决这个问题时，应该编写一段注释解释那样做的原因。这样，当下次看到这段代码时，就不会皱着眉头想，“究竟我为什么会那样做呢？”

另一个通常要遵循的原则是一边写代码一边做注释。有人希望在完成项目之后回头注释自己的代码。我敢保证，除非他没有惩罚性的开发时间表并且自律力比我们强，否则这样的事情不会发生。

3. 代码缩进

就像在所有的编程语言中一样，我们应该清楚一致地缩进代码。就像个人简历或商业信件的布局一样。缩进使代码更易读，更快地让人理解。

通常，任何属于某个控制结构的程序块都应该缩进于周围的代码。缩进的深度要明显（也就是说，大于一个空格）但不能太大。通常，要避免使用tabs键。虽然容易敲入，但是在许多人的屏幕上它占了太多的空间。我们在所有的项目中都使用了2～3个空格的缩进。

大括号的布局也是一个问题。最常用的两种布局模式如下所示：

模式1：

```
if (condition) {
  // do something
}
```

模式2：

```
if (condition)
{
  // do something else
}
```

使用哪一种模式由自己决定。但是一旦选择一个，就要在整个项目中保持一致，以免造成混淆。

25.4.2　分解代码

单段庞大的代码是非常可怕的。有人会编写这样一个庞大的脚本，它将所有的操作在一段主代码中完成。最好将它分解成多个函数或类，并将相关的函数和类保存到包含文件中。例如，可以将所有与数据库相关的函数放到一个名为dbfunctions.php的文件中。

将代码分解成清晰的代码块，原因如下所示：

- 它使源代码易读、易懂。
- 它可以提高源代码的重用性，将代码冗余减少到最小。例如，对前述的dbfunctions.php文件，可以在需要连接数据库的每个脚本中重用它。如果需要改变一种方式，只要在一个地方改变它即可。
- 方便团队工作。如果代码分成多个部分，就可将这些部分的编程责任分配给不同的团队成员。这就意味着，可以避免一个编程人员在等待另一个编程人员完成大块脚本GiantScript.php这样的情形，前者可以继续干他自己的工作。

在一个项目开始的时候，应该花些时间考虑如何将一个项目分成有计划的部分。这需要将功能区域分成多个部分。但也不要一味地陷进去，因为项目开始后功能区域的划分可能会变化。同时，还要决定哪些部分需要首先完成，哪些组件要基于另外的部分，还有开发所有组件的时间规划。

即使团队所有的工作人员分别编写各部分代码，通常将每个组件的主要责任分配给特定的人员是一个好办法。最终，如果某人的任务出了问题，该责任就可以由责任人承担。当然，还需要有人承担项目的管理工作——也就是说，他将确保各个部分工作正常并且能与其他部分配合工作。通常，这个人还要管理源代码的版本控制——在本章的后续内容中，我们将详细介绍版本控制。此人可以是项目经理，或者可以为他分配独立的责任。

25.4.3　使用标准的目录结构

在开始一个项目的时候，必须考虑组件结构如何反映到网站目录结构中。就像用一个庞大的脚本实现所有的功能是十分糟糕的一样，用一个大目录包含所有的东西也是非常糟糕的做法。决定如何按照组件、逻辑、内容和源代码库将目录分成多个部分。对目录结构进行文档化处理，并确认开发本项目的每一位工作人员都有一个副本，以确保他们可以从中查找需要的东西。

25.4.4　文档化和共享内部函数

在开发函数库的时候，应该允许开发队伍的每一位工作人员都可使用这些函数。通常，队伍中的每一个人都编写他们自己的一套数据库、日期和调试函数。事实上，这是一种时间浪费。应该让别人可以获取我们的函数和类。

请记住，即使代码存在于可以访问的区域或目录中，如果不告诉他们，他们也不会知道。

开发一个能够文档化内部函数库的系统，并使开发队伍中每一个编程人员可以获取它。

25.5 实现版本控制

版本控制是一门“艺术”，它适用于软件开发中的并发变更管理。通常，版本控制系统用作中央信息库或存档，它为访问和共享别人代码（或文档）提供一个可以控制的接口。

设想这样的情形，我们尝试去改进一些代码，但是却不幸将它弄坏，而且无论怎样也恢复不到原来的样子。或者，我们或客户认为网站早期的版本更好。或者，因为法律的原因必须回到早期版本。

设想另一种情形，开发队伍的两个成员希望对同一个文件进行修改。它们可能同时打开并编辑该文件，覆盖对方的修改。它们可能都工作于当地副本上，但是以不同的方式修改。如果可能发生这样的事情，必须让一位编程人员等待另一位编程人员完成对该文件的修改。

使用版本控制系统，可以解决所有这些问题。该系统可以记录信息库里每一个文件的修改情况，这样，不仅可以看到对它现在的描述，也可以清楚地了解到过去任何时候它的样子。版本控制系统的这个特性可将弄乱的代码恢复到已知的可工作的版本。也可以对一系列的特定文件设置标签，将其作为要发布的版本，这就意味着可以在这些代码的基础上继续开发，也可以随时获得已发表版本的副本。

版本控制还有助于解决多个编程人员合作编写程序的问题。每个编程人员可以在信息库里获取代码的一个副本（称为checking it out），并且当他们完成修改后，将这些修改提交到信息库（称checked in或committed）。版本控制系统也因此能够跟踪谁对某系统做了什么修改。

通常，这些系统还有管理并发更新的优点。并发更新是指两个编程人员可能在同一时刻修改同一个文件。例如，假设John和Mary都获取了他们所做项目的最近发布的一个副本。John完成了对某一文件的修改并将其提交到信息库。Mary也修改了那个文件，也试图提交。如果他们修改的不是文件的同一部分，版本控制系统就会将两个版本的文件合并。如果他们的修改互相冲突，系统将通知Mary并显示两种不同的版本。她就可以调整自己的代码以避免冲突。

大多数UNIX开发人员和开放源代码开发人员使用的版本控制系统是CVS。CVS是并发版本系统的缩写。CVS也是开放源代码。它与每个版本的UNIX捆绑在一起，也可以将其用于运行DOS或Windows的PC机和Mac机。它支持客户端-服务器模式，因此如果CVS服务器在网上是可见的，也可通过互联网进行连接，并从任何机器提交或获取代码。它用于PHP、Apache和Mozilla以及其他项目的开发，部分是因为这个原因。

从CVS的主页，可以为系统下载合适的CVS：http:// ximbiot.com/cvs/wiki/。

尽管基本CVS系统是一个命令行工具，然而不同的附加软件给它提供了美观的前台，包括基于Java的前台和Windows前台。这些也可以从CVS主页访问到。

Bitkeeper是一个非常有竞争力的版本控制产品，它为几个高性能的开放源代码项目所使用，其中包括MySQL和Linux内核。它对开放源代码项目来说是免费的：http://www.bitkeeper.com/。

当然，也可以选择商业化的CVS软件。其中之一就是运行在大多数常见平台上的perforce，它支持PHP。请注意，虽然它是商业软件，但是可以在开放源代码项目主页中找到其免费的许

可：http://www.perforce.com/。

25.6　选择一个开发环境

前面关于版本控制的讨论就会带来更多关于开发环境选择的话题。我们真正需要的是一个文本编辑器和一个用来测试的浏览器，但是大多数编程人员还是非常喜欢集成开发环境（IDE），通常他们在IDE下的开发效率更高。

目前，有许多适用于专门的PHP IDE的免费项目，包括KPHPDevelop，该开发环境适用于Linux平台的KDE桌面环境，可以从如下站点获得它：http://kphpdev.sourceforge.net/。

然而，目前最好的PHP IDE都是商业性的。例如，zend.com公司的Zend Studio，activestate.com公司的Komodo，以及nusphere.com公司的PHPEd，都提供了功能全面的集成开发环境。所有这些产品都提供了可供下载免费试用的版本，但要长期使用，必须付费。Komodo提供了一个非常便宜的非商业许可。

25.7　项目的文档化

我们可以为正在开发的项目制作许多种文档，这些文档包括但是不局限于下面这些：

- 设计文档
- 技术文档/开发指南
- 数据词典（包含类文档）
- 用户指南（尽管大多数Web应用程序是自我解释的）

在这里，我们并不是要讲解如何写技术文档，而是建议自动化这些过程，从而减轻开发工作的负担。

在一些语言中，有一些自动生成这些文档的方法——特别是技术文档和数据词典。例如，javadoc可以生成HTML格式的树型结构文件目录，该树型结构就包含了Java程序中类成员的原型及其描述。

对于PHP，也有一些可以提供该功能的工具。其中包括如下这些：

- phpdoc，可以从如下站点获取：http://www.phpdoc.de/。

它是PEAR用来对代码执行文档化处理的系统。请注意，术语*phpDoc*用于描述几个这种类型的项目，这就是其中之一。

- PHPDocumentor，可以从如下站点获取：http://phpdocu.sourceforge.net。

PHPDocumentor可以提供非常类似于javadoc的输出，而且稳定性非常好。看上去，PHPDocumentor的开发团队比我们这里介绍的其他两个开发团队更主动。

- phpautodoc，可以从如下站点获取：http://sourceforge.net/projects/phpautodoc/。

phpautodoc的输出结果也非常类似于javadoc的输出结果。

查找更多的这种类型（和常用的PHP组件）应用软件的一个好地方是SourceForge：http://sourceforge.net。

SourceForge主要用于UNIX/Linux社区，但是这里也有许多运行于其他平台的项目。

25.8　建立原型

通常，*建立原型*是一个适用于开发Web应用程序的开发周期。一个原型是对于获得用户需求很有意义的工具。通常，原型是应用程序的简化的一部分工作版本，可以用来与客户进行讨论，并作为最终系统的基础。对原型的多次反复讨论将最终产生应用程序。这种方法的好处就是它让我们更紧密地同客户或者终端用户一起工作，从而产生一个他们喜欢并且有主人翁感的系统。

为了能够较快地将一个原型“凑到一起”，需要一些特别的技术和工具。这是一个基于组件的方法发挥很好作用的地方。如果可以访问一系列已经存在的组件，这些组件在内部和公共地方都可以访问，我们将能够更快地实现它。另一个有用的快速开发原型的工具是模板。我们将在下一节详细介绍它。

使用建立原型的方法有两个主要问题。我们必须意识到这些问题到底是什么，从而避免它们，使这种方法发挥最大潜力。

第一个问题是编程人员通常会发现他们会因为某种原因很难丢弃他们所编写的代码。原型的编写通常会很快。可是过后，我们会发现自己没有以理想的或者近似理想的方法建立原型。代码的部分可以修改，但是如果整体结构错误，那就麻烦了。问题是Web应用程序经常要在巨大的时间压力下创建，因而没有时间来修改它。这样我们就被一个极难维护的糟糕设计缠住了。

可以像本章前面讨论的那样做一些计划来避免这个问题。但是也要记住，有时候，从头开始再次写新代码比修改原代码还容易。尽管这可能看起来是没有时间来完成的，但是在以后它可能会减少许多麻烦。

建立原型的第二个问题是一个系统可能会以一个无休止的原型结束。每次认为完成了，雇员又将提出更好的改进意见或者附加功能或网站更新。这种发展的蔓延可能让我们没完没了地构建一个项目。

要避免这些问题，需要草拟一份设计计划，规定项目的反复次数，设定某个日期之后如果不重新计划、预算和规划就不能添加新功能。

25.9　将逻辑和内容分离

我们可能对使用HTML描述一个网络文档结构，和用级联样式表单（*CSS*）描述其外观比较熟悉。这种将外观从内容分离出来的想法可以扩展到脚本编码中去。一般来说，如果将逻辑从内容中分离、内容从外观分离，那么网站将更易长期使用和维护。这就归结为PHP和HTML的分离。

对于只有行数不多的代码或脚本的简单项目，使用分离法是得不偿失的。但是当项目扩大的时候，就必须找到将逻辑和内容分离的方法。如果不这样分离，代码将越来越难维护。如果我们决定将某一新的设计应用于网站，要在原来的代码中嵌入许多代码，那么做这样的修改简直就是一场噩梦。

要分离逻辑和内容，有3种基本的方法，如下所示：

- 用包含文件保存不同部分的内容。这种方法虽然过分单纯，但是如果网站主要是静态的，

它很有效。在第5章“代码重用与函数编写”的TLA咨询公司例子中，我们已经解释了这种方法。

- 用一个函数或者带有一组成员函数的类API将动态内容插入到静态网页模板中。在第6章“面向对象的PHP”中，我们已经介绍了这种方法。
- 使用模板系统。模板系统可以用来解析静态模板，并通过正则表达式动态数据代替占位符标记。这样做的主要好处是，如果别人设计模板，例如图形设形师，他就根本不需要懂PHP代码。应该使提供的模板只需做最小的修改。

许多模板系统是可以从网站或其他地方获取的。最常用的可能是Smarty，可以从如下站点获得：http://smarty.php.net/。

25.10 优化代码

如果没有Web编程经验，代码的优化看起来真的很重要。当使用PHP的时候，用户等待Web应用程序的大部分时间都是来自连接和下载的次数。而优化代码对这些时间可能没有什么影响。

25.10.1 使用简单优化

可以进行一些简单的优化，使得数据库连接和下载时间有所改善。这里介绍的大多数优化都与使用PHP代码来集成数据库（如MySQL）的应用程序有关。这些优化包括：

- 减少数据库连接。通常，连接数据库是所有脚本中最慢的部分。可以通过使用持久稳固的连接来解决这个问题。
- 加速数据库查询。减少所做的查询数量，并且确保这些查询得到了优化。对一个复杂查询（因此比较慢）来说，通常可以使用几种方法来实现。在数据库的命令行界面运行查询，试验使用不同的方法来加速查询。在MySQL中，可以使用`EXPLAIN`语句来查看查询可能在什么地方绕弯路（该语句的使用已经在第12章中介绍过）。通常的原则是尽可能减少连接，增加索引。
- 使PHP中生成的静态内容减少到最小。如果生成的每段HTML都来自`echo`或`print()`语句，就要花多得多的时间（这是前述的分离逻辑和内容的论点之一）。这也适用于动态生成的图像按钮——可以使用PHP来生成一次按钮之后就能在需要的时候重用它。如果每次页面载入的时候需要从函数或模板生成纯静态网页，可以考虑运行一次函数或使用一次模板，然后保存结果。
- 尽可能使用字符串函数代替正则表达式。前者速度更快。

25.10.2 使用Zend产品

Zend科技公司拥有在PHP 4版本及其以后版本中使用的PHP脚本引擎（开放源代码）。除了基本引擎外，还可以下载Zend优化程序。这是一个多策略的优化程序，它可以优化代码，提高脚本运行速度从40%～100%不等。需要PHP 4.0.2或更高版本来运行这个优化程序。虽然源代码不是开放的，但是可以从Zend的站点免费下载它：http://www.zend.com。

这个插件是通过优化由脚本运行时编译产生的代码来实现的。其他Zend产品包括Zend Studio、Zend Accelerator、Zend Encoder，都需要商业性的许可协议。

25.11　测试

审查和测试代码是软件工程中另一个基本要点，然而在Web开发设计中，它常常被忽略。通常，测试者会满足于运行系统，测试两三个例子，然后说，“哦，是的，它工作得很好。”这是经常犯的一个错误。在准备将项目作为产品发布之前，要确保它已经在各种情况下经过了广泛的测试和审查。

我们推荐使用两种方法来降低代码错误率（永远不可能清除所有错误，但肯定可以排除大部分错误或将错误减到最小）。

首先，采用代码审查。也就是让另一位编程人员或另一组编程人员查看代码，并提出改进意见。通常经过这样的分析，我们能够对下面的问题提出建议：

- 可能忽略的错误
- 原程序设计者没有想到的测试用例
- 代码优化
- 安全性的提高
- 可以使用已有组件改进某一段代码
- 附加功能

即使一个人独立开发，我们建议也要找一个“代码查错者”（“查错者”和开发者情况差不多，可互相审查代码）。

第二个建议是让最终用户作为Web应用程序的测试人员。Web应用程序和桌面应用程序的主要区别就是每个人都将使用Web应用程序。我们不能假定每个人都能熟练使用计算机，也不可能给他们提供厚厚的手册或快速参考卡片。必须让应用程序自动存档而且自己说明自己。此外，还得考虑用户希望如何使用应用程序。可用性是极为重要的。

对于有经验的编程人员或网上冲浪者来说，很难理解没有经验的用户所遇到的问题。解决这个问题的方法之一就是请代表典型用户的人做测试员。

过去用作测试的一种方法是只基于beta版发布Web应用程序。当认为Web应用程序的主要错误已经消除时，可以将应用程序公布于小部分测试用户，并为网站提供小流量通信。可以许诺为前100位反馈网站使用意见的用户提供免费服务。我可以确信他们会带来一些开发人员原先并未想到的数据和用法及其组合。如果是在为某客户公司开发Web站点，通常，可以通过让公司的员工使用站点而提供许多没有经验的用户。（这也有内在的好处，它可以提高员工在网站中的主人翁感。）

25.12　进一步学习

这个领域有太多的内容。基本上，我们只是主要讨论了软件工程的科学，而关于软件工程，有许多图书已经介绍了。

解释了文档式网站和应用程序式网站这两种对立观点的著作是由Thomas A. Powell编写的

《Web站点工程学：不仅仅是设计Web页面》（*Web Site Engineering: Beyond Web Page Design*）。任何自己喜欢的软件工程图书也都可留下来作为后备图书。

要了解版本控制的信息，可以访问CVS网站：http://ximbiot.com/cvs/wiki/。

关于版本控制的书并不多（但是，它却非常重要，这让人惊讶），但是可以参考由Karl Franz Fogel编著的《使用CVS进行开放源代码开发》（*Open Source Development with CVS*）或由Gregor N. Purdy编著的《CVS袖珍参考》（*CVS Pocket Reference*）。

如果要查找PHP组件、IDE或文档系统，可以试试SourceForge网站：http://sourceforge.net。

在本章中，我们介绍的许多内容都在Zend网站中的文章中讨论到了。可以访问Zend网站，查找与本章内容相关的信息。也可以考虑从该站点下载优化器：http://www.zend.com。

如果对本章内容感兴趣，还可了解极限编程（Extreme Programming），它是针对软件需求频繁改变的领域（例如，Web开发领域）而研究的软件开发方法。极限编程的网站：http://www.extremeprogramming.org。

25.13　下一章

在第26章“调试”中，我们将讨论不同类型的程序错误、PHP错误信息和查找错误的技术。