



无兄弟，不编程！

Broadview®
www.broadview.com.cn



细说 PHP

第2版

LAMP兄弟连 组编 高洛峰 编著
PHP爱好者的营地，LAMP开发者的联盟



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

DVD光盘
150小时
视频教学

内 容 简 介

PHP 是开发 Web 应用系统最理想的工具,易于使用、功能强大、成本低廉、高安全性、开发速度快且执行灵活。全书以实用为目标设计,包含 PHP 开发最主流的各项技术,对每一个知识点都进行了深入详细的讲解,并附有大量的实例代码,图文并茂。系统地介绍了 PHP 的相关技术及其在实际 Web 开发中的应用。

全书共六个部分,分为 30 个章节,每一章都是 PHP 独立知识点的总结。内容涵盖了动态网站开发的前台技术(HTML+CSS)、PHP 编程语言的语法、PHP 的常用功能模块和实用技巧、MySQL 数据库的设计与应用、PHP 面向对象的程序设计思想、数据库抽象层 PDO、Smarty 模板技术、Web 开发的设计模式、自定义框架 BroPHP、Web 项目开发整个流程等目前 PHP 开发中最主流的技术。每一章中都有大量的实用示例,以及详尽的注释,加速读者的理解和学习,也为每章的技术点设置了大量的自测试题。最后以一个比较完整的、采用面向对象思想,以及通过 MVC 模式设计,并结合 Smarty 模板,基于 BroPHP 框架的 CMS 系统为案例,详细介绍了 Web 系统开发从设计到部署的各个细节,便于更好地进行开发实践。

对于 PHP 应用开发的新手而言,本书不失为一本好的入门教材,内容既实用又全面,所有实例都可以在开发中直接应用,并辅以大量的视频教程,使读者轻松掌握所学知识。另外,本书也适合有一定基础的网络开发人员和网络爱好者,以及大中专院校的师生阅读与参考。不仅可以作为 PHP 开发的学习用书,还可以作为从事 Web 开发的程序员的参考用书和必备手册。对于行家来说,本书也是一本难得的参考手册,读者必将从中获益。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

细说 PHP / 高洛峰编著. —2 版. —北京: 电子工业出版社, 2012.10
ISBN 978-7-121-18563-2

I. ①细… II. ①高… III. ①PHP 语言—程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字(2012)第 222286 号

策划编辑: 李 冰

责任编辑: 高洪霞

印 刷: 北京天宇星印刷厂

装 订: 三河市皇庄路通装订厂

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 860×1092 1/16 印张: 51.75 字数: 1458 千字

印 次: 2012 年 10 月第 1 次印刷

印 数: 3500 册 定价: 109.00 元(含 DVD 光盘 1 张)

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

LAMP兄弟连

无兄弟，不编程

《细说 PHP（第2版）》是在第1版热销的基础上，与兄弟连六年教学实践的完美结合。

六年时间，兄弟连由一个名不见经传的小公司，成长为国内 PHP 培训领域翘楚，累计培训 PHP 程序员四千余人、论坛会员突破十万、技术视频下载量逾百万、电子杂志《草根》日渐成熟、《细说 PHP（第1版）》图书销量更是名列 PHP 原创图书之首。《细说 PHP（第2版）》的出版也是对兄弟连多年教学积累的梳理与总结，书中穿插了更多的教学案例、课后习题，辅以原创视频，加入了兄弟连课程体系中特级课的部分知识点，逻辑性更强，知识点更加完善。

第2版历时一年多，作者高洛峰对本书倾注了大量的心血，除延续第1版深入浅出、循序渐进、剖析细致等特点外，结合自身多年的教学经验，力求做到更精炼、更实用、更全面。

写作本书的初衷不仅是开发一本培训教材，更主要的原因是希望推广 PHP 技术。兄弟连的学员越来越多，但更多人不一定有时间和精力参加面授学习，一本专业、全面的技术书籍无疑是他们最需要的。本书使读者对 PHP 技术有更深刻、更系统的了解，掌握 Web 系统开发的完整思路，通过本书内容的学习足以完成动态网站的建设和一些常用的 Web 系统软件开发工作。

兄弟连还将陆续推出 PHP、Linux、Java、Android、Python 等各方面的图书，期望可以对开源技术在国内的推广做出贡献，也希望广大读者朋友继续支持我们，支持兄弟连。我们也将一直秉承兄弟连的口号“无兄弟，不编程！”，让众多的技术爱好者、从业者、创业者团结起来，拓展人脉、相互学习、相互帮助，为未来的职业发展打下良好的基础。

今天的兄弟连，不仅仅是一所培训学校，更是技术人的筑梦工场，我们秉承兄弟连的核心价值：一是优秀的教学；二是严格的管理；三是职业素质课贯穿始终。教书育人，成就英才，一直是我们的目标，愿本书的出版，让更多朋友进入 PHP 的领域，成就自己的职业梦想！

关注兄弟连微博：<http://weibo.com/lampbrother>

兄弟连创始人 李超

前言

PREFACE

PHP 是一种开源免费的开发语言，具有程序开发速度快、运行快、技术本身学习快等快捷性的特点，无疑是当今 Web 开发中最佳的编程语言。与 JSP 和 ASP 相比，PHP 具有简易性、高安全性和执行灵活等优点，使用 PHP 开发的 Web 项目，在软件方面的投资成本较低、运行稳定。因此现在越来越多的供应商、用户和企业投资者日益认识到，使用 PHP 开发的各種商业应用和协作构建各种网络应用程序，变得更加具有竞争力，更加吸引客户。无论是从性能、质量还是价格上，PHP 都将成为企业和政府信息化所必须考虑的开发语言。

本书包括六大部分+附录，作为 PHP 学习的七个阶段，从了解 Web 开发构件开始到可以完成一个标准化高质量的项目为止。所有内容皆为当今 Web 项目开发必用的内容，涵盖了 PHP 的绝大多数知识点，对于某一方面的介绍再从多角度进行延伸。全部内容围绕 PHP 的面向对象思想设计编写，帮助读者深刻理解 PHP 开发技术，一步一步引导读者从 PHP 面向过程的开发模式进入到面向对象的开发时代。本书全部技术点以 PHP 5 最流行的版本为主，详细地介绍了 PHP 及与其相关的 Web 技术，可以帮助读者在较短的时间内熟悉并掌握比较实用的 PHP 技术。其中包括当前比较流行的 DIV+CSS 标准化网页布局、PHP 面向对象技术、数据库抽象层 PDO 和 Smarty3 模板引擎、学习型 PHP 框架 BroPHP 等主流技术，实用性非常强。本书所涉及的实例全部以特定的应用为基础，读者在学习和工作过程中，可以直接应用本书给出的一些独立模块和编程思想。

本书编写的宗旨是让读者能拥有一本 PHP 方面的学习和开发使用的最好书籍，章节虽然不是很多，但对所罗列出的每个知识点都进行了细化和延伸，并力求讲解到位，让读者可以轻松地读懂。所介绍的知识点是不需要借助其他任何书籍进行辅助和补充的。而且对于几乎每个知识点都有对应且详实的可运行的代码配套，对所有实例代码都附有详细注释、说明及运行效果图。在大部分章节的最后一节都结合一个实用的案例，把该章中涉及的零散知识点串在一起进行分析总结，步骤详细，可操作性强。另外在每个章节的最后还为读者安排了大量的和本章知识点配套的自测试题（附加在光盘中），能更好地帮助读者掌握理论知识点，提高实际编程能力，寓学于练。

本书的出版距离上一版发行整三年的时间，在第 1 版发行后的一年就开始筹划第 2 版。所有实例都经过了反复的测试，每一句话都进行了反复的推敲，在这两年时间里几乎占用了笔者的全部业余时间。为《细说 PHP》（第 2 版）筹划的几个重要事件如下：

1. 根据第 1 版读者的反馈，在第 2 版中对大部分内容进行重新整理和优化。

2. 用一年时间为本书重新录制了长达 150 小时的视频教程，全面覆盖了书中的每个知识点。
3. 专门为本书开发了一个学习型的 PHP 框架：BroPHP，这是目前国内唯一一个学习型的专用 PHP 框架。并用半年时间通过了几百个项目的测试和应用，已经非常成熟和稳定。
4. 书中的每个应用实例都做到了最优，直接可以应用在实际项目开发中。
5. 结合多家互联网公司的项目开发资料，总结出了一份标准的项目开发流程和几个重要的标准化项目开发文档。

超强资源配套学习，跟踪服务帮助读者提高

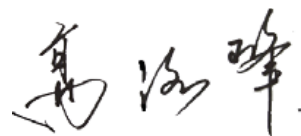
在本书所附的 DVD 光盘中，附书中所有开发实例的源代码，读者在开发中可以直接使用。本书部分章节及课后习题、附录由于书的容量限制，也附加到了光盘中。光盘中还赠送配套的教学视频，共计长达 150 个小时（由于光盘空间有限，部分视频需要读者到兄弟连论坛：bbs.lampbrother.net 下载）。通过参考本书再结合视频教学光盘，可以加快对知识点的掌握，加快学习进度。

为了帮助读者学习到更多的 PHP 技术，在兄弟连论坛还可以下载常用的技术手册、安装 LAMP 环境所需要的开源软件和本书每章后面为读者安排的大量自测题配套答案。笔者及“LAMP 兄弟连”的全体讲师和技术人员也会及时回答读者提问，与读者进行在线技术交流，并为读者提供各类技术文章，帮助读者提高开发水平，解决读者在开发中遇到的疑难问题。

本书适合读者

- 接受 PHP 培训的学员
- Web 开发爱好者
- 网站维护及管理人员
- 初级或专业的网站开发人员
- 大中专院校的教师及培训中心的讲师
- 进行毕业设计和对 PHP 感兴趣的学生
- 从事 ASP 或 JSP 而想转向 PHP 开发的程序员

本书由高洛峰主笔编著，参加编写及审校工作的人员有李明、李超、李文凯、赵桐正、丛浩、闫海静、张礼军、李捷、张涛，在此一并表示感谢。



2012 年 8 月

目 录

CONTENTS

第 1 部分 Web 开发入门篇

第 1 章 LAMP 网站构建 2

1.1	介绍网站给你认识	2
1.1.1	Web应用的优势	3
1.1.2	Web 2.0 时代的互联网	4
1.1.3	Web开发标准	5
1.1.4	认识脚本语言	6
1.1.5	了解HTTP协议	6
1.2	动态网站开发所需的Web构件	10
1.2.1	客户端浏览器	10
1.2.2	超文本标记语言HTML	12
1.2.3	层叠样式表CSS	13
1.2.4	客户端脚本编程语言JavaScript	13
1.2.5	Web服务器	14
1.2.6	服务器端编程语言	15
1.2.7	数据库管理系统	16
1.3	几种主流的Web应用程序平台	17
1.3.1	Web应用程序开发平台对比分析	17
1.3.2	动态网站开发平台技术比较	18
1.4	Web的工作原理	19
1.4.1	情景 1：服务器不带应用程序服务器和数据库	19
1.4.2	情景 2：带应用程序服务器的Web服务器	20
1.4.3	情景 3：浏览器访问服务器端的数据库	21

1.5	LAMP网站开发组合概述	22
1.5.1	Linux操作系统	22
1.5.2	Web服务器Apache	22
1.5.3	MySQL数据库管理系统	23
1.5.4	PHP后台脚本编程语言	23
1.5.5	LAMP发展趋势	25
1.6	学PHP需要学习什么内容	26
1.6.1	学PHP之前的准备	26
1.6.2	学PHP时需要了解或掌握的内容	27
1.6.3	优秀的Web程序员是怎样练成的	28
1.7	小结	29
	本章必须掌握的知识点	29
	本章需要了解的内容	29

第 2 章 HTML 的设计与应用 30

2.1	网页制作概述	30
2.1.1	HTML基础	30
2.1.2	简单HTML实例制作	31
2.2	HTML语言的语法	32
2.2.1	HTML标签和元素	32
2.2.2	HTML语法不区分字母大小写	32
2.2.3	HTML标签属性	33
2.2.4	HTML颜色值的设置	33
2.2.5	HTML文档注释	33
2.2.6	HTML代码格式	33
2.2.7	HTML字符实体	34

2.3	HTML文件的主体结构	34
2.4	HTML文档头部元素<head>	35
2.4.1	<title>元素	35
2.4.2	<base>元素	35
2.4.3	<link>元素	36
2.4.4	<meta>元素	36
2.5	HTML文档主体标记	37
2.6	文字版面的编辑	37
2.6.1	格式标签	38
2.6.2	文本标签	39
2.7	创建图像和链接	41
2.7.1	插入图片	41
2.7.2	建立锚点和超链接	42
2.8	使用HTML表格	42
2.9	HTML框架结构	45
2.10	HTML表单设计	47
2.11	小结	51
	本章必须掌握的知识点	51
	本章需要了解的内容	51
	本章需要拓展的内容	51
	本章的学习建议	52
第 3 章 层叠样式表 CSS		53
3.1	CSS简介	53
3.2	CSS规则的组成	54
3.2.1	CSS注释	55
3.2.2	长度单位	56
3.2.3	颜色单位和URL值	56
3.3	在HTML文档中放置CSS的几种方式	57
3.3.1	内联样式表	57
3.3.2	嵌入一个样式表	57
3.3.3	连接到一个外部的样式表	57
3.4	CSS选择器	58
3.4.1	HTML选择器	58
3.4.2	类选择器	58
3.4.3	ID选择器	59
3.4.4	关联选择器	59
3.4.5	组合选择器	59

3.4.6	伪元素选择器	60
3.5	CSS常见的样式属性和值	60
3.5.1	字体属性	60
3.5.2	颜色属性	61
3.5.3	背景属性	61
3.5.4	文本属性	62
3.5.5	边框属性	62
3.5.6	鼠标光标属性	64
3.5.7	列表属性	64
3.5.8	综合示例	65
3.6	小结	67
	本章必须掌握的知识点	67
	本章需要了解的内容	67
	本章需要拓展的内容	67
	本章的学习建议	68

第 4 章 DIV+CSS 网页标准化布局		69
4.1	DIV+CSS对页面布局的优势	69
4.2	“无意义”的HTML元素div和span	70
4.3	W3C盒子模型	70
4.4	和页面布局有关的CSS属性	72
4.5	盒子区块框的定位	74
4.5.1	相对定位	74
4.5.2	绝对定位	74
4.6	使用盒子模型的浮动布局	75
4.6.1	设置浮动	75
4.6.2	行框和清理	76
4.7	DIV+CSS的兼容性问题	78
4.7.1	不同浏览器解释盒子模型的差异	79
4.7.2	设置浏览器去遵循W3C标准	80
4.8	使用盒子模型设计页面布局	81
4.8.1	居中设计	81
4.8.2	设置两列浮动的布局	82
4.8.3	设置三列浮动的布局	83
4.8.4	设置多列浮动的布局	84
4.9	DIV+CSS网站首页页面布局实例	85
4.9.1	HTML文件的设计	85
4.9.2	CSS文件设计	87

4.10	小结	89
	本章必须掌握的知识点	89
	本章需要了解的内容	89
	本章需要拓展的内容	89
	本章的学习建议	89

第 2 部分 PHP 基础篇

第 5 章 从搭建你的 PHP 开发环境开始 92

5.1	几种常见的PHP环境安装方式	92
5.1.1	Linux系统下源代码包方式安装环境	92
5.1.2	在Windows系统上安装Web工作环境	93
5.1.3	搭建学习型的PHP工作环境	93
5.2	环境安装对操作系统的选择	93
5.2.1	选择网站运营的操作系统	93
5.2.2	选择网站开发的操作系统	94
5.3	安装集成PHP开发环境	94
5.3.1	安装前准备	94
5.3.2	安装步骤	95
5.3.3	环境测试	96
5.4	phpMyAdmin的配置与应用	98
5.4.1	HTTP身份验证模式	99
5.4.2	cookie身份验证模式	99
5.4.3	config身份验证模式	100
5.5	小结	101
	本章必须掌握的知识点	101
	本章需要了解的内容	101
	本章需要拓展的内容	101

第 6 章 PHP 的基本语法 102

6.1	PHP在Web开发中的应用	102
6.1.1	就从认识PHP开始吧	102
6.1.2	PHP都能做什么	103
6.2	第一个PHP脚本程序	105
6.3	PHP语言标记	108
6.3.1	将PHP代码嵌入HTML中的位置	109
6.3.2	解读开始和结束标记	109
6.4	指令分隔符“分号”	111

6.5	程序注释	111
6.6	在程序中使用空白的处理	113
6.7	变量	113
6.7.1	变量的声明	114
6.7.2	变量的命名	115
6.7.3	可变变量	116
6.7.4	变量的引用赋值	116
6.8	变量的类型	117
6.8.1	类型介绍	118
6.8.2	布尔型 (boolean)	118
6.8.3	整型 (integer)	119
6.8.4	浮点型 (float或double)	120
6.8.5	字符串 (String)	120
6.8.6	数组 (Array)	122
6.8.7	对象 (Object)	123
6.8.8	资源类型 (Resource)	123
6.8.9	NULL类型	124
6.8.10	伪类型介绍	124
6.9	数据类型之间相互转换	125
6.9.1	自动类型转换	125
6.9.2	强制类型转换	126
6.9.3	类型转换细节	126
6.9.4	变量类型的测试函数	127
6.10	常量	128
6.10.1	常量的定义和使用	128
6.10.2	常量和变量	129
6.10.3	系统中的预定义常量	129
6.10.4	PHP中的魔术常量	129
6.11	PHP中的运算符	130
6.11.1	算术运算符	131
6.11.2	字符串运算符	133
6.11.3	赋值运算符	133
6.11.4	比较运算符	134
6.11.5	逻辑运算符	135
6.11.6	位运算符	136
6.11.7	其他运算符	139
6.11.8	运算符的优先级	140
6.12	表达式	141

6.13	小结	142
	本章必须掌握的知识点	142
	本章需要了解的内容	142
	本章需要拓展的内容	142

第 7 章 PHP 的流程控制结构 143

7.1	分支结构	143
7.1.1	单一条件分支结构 (if)	144
7.1.2	双向条件分支结构 (else从句)	145
7.1.3	多向条件分支结构 (elseif子句)	145
7.1.4	多向条件分支结构 (switch语句)	147
7.1.5	巢状条件分支结构	148
7.1.6	条件分支结构实例应用 (简单计算器) ..	149
7.2	循环结构	151
7.2.1	while语句	152
7.2.2	do...while循环	154
7.2.3	for语句	155
7.3	特殊的流程控制语句	157
7.3.1	break语句	157
7.3.2	continue语句	158
7.3.3	exit语句	159
7.4	小结	160
	本章必须掌握的知识点	160
	本章需要了解的内容	160

第 8 章 PHP 的函数应用 161

8.1	函数的定义	161
8.2	自定义函数	162
8.2.1	函数的声明	162
8.2.2	函数的调用	164
8.2.3	函数的参数	165
8.2.4	函数的返回值	166
8.3	函数的工作原理和结构化编程	168
8.4	PHP变量的范围	168
8.4.1	局部变量	169
8.4.2	全局变量	170
8.4.3	静态变量	171
8.5	声明及应用各种形式的PHP函数	171

8.5.1	常规参数的函数	173
8.5.2	伪类型参数的函数	173
8.5.3	引用参数的函数	173
8.5.4	默认参数的函数	175
8.5.5	可变个数参数的函数	176
8.5.6	回调函数	177

8.6	递归函数	181
8.7	使用自定义函数库	182
8.8	小结	183
	本章必须掌握的知识点	183
	本章需要了解的内容	183
	本章需要拓展的内容	183

第 9 章 PHP 中的数组与数据结构 184

9.1	数组的分类	184
9.2	数组的定义	186
9.2.1	直接赋值的方式声明数组	186
9.2.2	使用array()语言结构新建数组	188
9.2.3	多维数组的声明	188
9.3	数组的遍历	190
9.3.1	使用for语句循环遍历数组	191
9.3.2	使用foreach语句遍历数组	193
9.3.3	联合使用list()、each()和while循环 遍历数组	195
9.3.4	使用数组的内部指针控制函数遍历数组 ..	198
9.4	预定义数组	199
9.4.1	服务器变量：\$_SERVER	200
9.4.2	环境变量：\$_ENV	200
9.4.3	URL GET变量：\$_GET	200
9.4.4	HTTP POST变量：\$_POST	201
9.4.5	request变量：\$_REQUEST	202
9.4.6	HTTP文件上传变量：\$_FILES	202
9.4.7	HTTP Cookies：\$_COOKIE	202
9.4.8	Session变量：\$_SESSION	203
9.4.9	Global变量：\$GLOBALS	203
9.5	数组的相关处理函数	203
9.5.1	数组的键/值操作函数	203
9.5.2	统计数组元素的个数和唯一性	206

9.5.3	使用回调函数处理数组的函数	208
9.5.4	数组的排序函数	211
9.5.5	拆分、合并、分解和接合数组	215
9.5.6	数组与数据结构	218
9.5.7	其他有用的数组处理函数	220
9.6	操作PHP数组需要注意的一些细节	221
9.6.1	数组运算符	221
9.6.2	删除数组中的元素操作	222
9.6.3	关于数组下标的注意事项	222
9.7	小结	223
	本章必须掌握的知识点	223
	本章需要了解的内容	223
	本章需要拓展的内容	223

第 10 章 PHP 面向对象的程序设计 224

10.1	面向对象的介绍	224
10.1.1	类和对象之间的关系	225
10.1.2	面向对象的程序设计	225
10.2	如何抽象一个类	226
10.2.1	类的声明	226
10.2.2	成员属性	227
10.2.3	成员方法	228
10.3	通过类实例化对象	229
10.3.1	实例化对象	229
10.3.2	对象类型在内存中的分配	230
10.3.3	对象中成员的访问	232
10.3.4	特殊的对象引用“\$this”	234
10.3.5	构造方法与析构方法	235
10.4	封装性	238
10.4.1	设置私有成员	239
10.4.2	私有成员的访问	240
10.4.3	__set()、__get()、__isset()和__unset()四个方法	242
10.5	继承性	247
10.5.1	类继承的应用	247
10.5.2	访问类型控制	249
10.5.3	子类中重载父类的方法	251
10.6	常见的关键字和魔术方法	253

10.6.1	final关键字的应用	253
10.6.2	static关键字的使用	254
10.6.3	单态设计模式	255
10.6.4	const关键字	257
10.6.5	instanceof关键字	257
10.6.6	克隆对象	257
10.6.7	类中通用的方法__toString()	259
10.6.8	__call()方法的应用	259
10.6.9	自动加载类	261
10.6.10	对象串行化	262

10.7 抽象类与接口 265

10.7.1	抽象类	265
10.7.2	接口技术	266

10.8 多态性的应用 268

10.9 面向对象版图形计算器 270

10.9.1	需求分析	270
10.9.2	功能设计及实现	271
10.9.3	类的组织架构	276

10.10 小结 277

	本章必须掌握的知识点	277
	本章需要了解的内容	278
	本章需要拓展的内容	278

第 11 章 字符串处理 279

11.1 字符串的处理介绍 279

11.1.1	字符串的处理方式	279
11.1.2	字符串类型的特点	280
11.1.3	双引号中变量解析总结	280

11.2 常用的字符串输出函数 281

11.3 常用的字符串格式化函数 284

11.3.1	去除空格和字符串填补函数	284
11.3.2	字符串大小写的转换	285
11.3.3	和HTML标签相关的字符串格式化	286
11.3.4	其他字符串格式化函数	290

11.4 字符串比较函数 291

11.4.1	按字节顺序进行字符串比较	291
11.4.2	按自然排序进行字符串比较	292

11.5 小结 293

本章必须掌握的知识点	293
本章需要拓展的内容	293
第 12 章 正则表达式	294
12.1 正则表达式简介	294
12.1.1 选择PHP正则表达式的处理函数库	294
12.2 正则表达式的语法规则	295
12.2.1 定界符	296
12.2.2 原子	296
12.2.3 元字符	298
12.2.4 模式修正符	301
12.3 与Perl兼容的正则表达式函数	302
12.3.1 字符串的匹配与查找	302
12.3.2 字符串的替换	305
12.3.3 字符串的分割和连接	310
12.4 文章发布操作示例	312
12.5 小结	317
本章必须掌握的知识点	317
本章需要了解的内容	317
本章需要扩展的内容	317

第 3 部分 PHP 常用功能模块篇

第 13 章 PHP 的错误和异常处理	320
13.1 错误处理	320
13.1.1 错误报告级别	321
13.1.2 调整错误报告级别	321
13.1.3 使用trigger_error()函数来替代die()	323
13.1.4 自定义错误处理	323
13.1.5 写错误日志	325
13.2 异常处理	327
13.2.1 异常处理实现	328
13.2.2 扩展PHP内置的异常处理类	328
13.2.3 捕获多个异常	330
13.3 小结	332
本章必须掌握的知识点	332
本章需要了解的内容	332

第 14 章 PHP 的日期和时间	333
14.1 UNIX时间戳	333
14.1.1 将日期和时间转变成UNIX时间戳	333
14.1.2 日期的计算	335
14.2 在PHP中获取日期和时间	335
14.2.1 调用getdate()函数取得日期/时间信息	335
14.2.2 日期和时间格式化输出	336
14.3 修改PHP的默认时区	337
14.4 使用微秒计算PHP脚本执行时间	338
14.5 日历类	339
14.6 小结	343
本章必须掌握的知识点	343
本章需要了解的内容	343
本章需要拓展的内容	343
本章的学习建议	343

第 15 章 文件系统处理	344
15.1 文件系统概述	344
15.1.1 文件类型	344
15.1.2 文件的属性	345
15.2 目录的基本操作	348
15.2.1 解析目录路径	348
15.2.2 遍历目录	349
15.2.3 统计目录大小	351
15.2.4 建立和删除目录	352
15.2.5 复制目录	352
15.3 文件的基本操作	353
15.3.1 文件的打开与关闭	353
15.3.2 写入文件	355
15.3.3 读取文件内容	356
15.3.4 访问远程文件	358
15.3.5 移动文件指针	359
15.3.6 文件的锁定机制	360
15.3.7 文件的一些基本操作函数	363
15.4 文件的上传与下载	364
15.4.1 文件上传	364
15.4.2 处理多个文件上传	367
15.4.3 文件下载	368

第4部分 数据库开发篇

15.5	设计经典的文件上传类	369
15.5.1	需求分析	369
15.5.2	程序设计	370
15.5.3	文件上传类代码实现	370
15.5.4	文件上传类的应用过程	375
15.6	小结	376
	本章必须掌握的知识点	376
	本章需要了解的内容	377
	本章需要拓展的内容	377
	本章的学习建议	377

第16章 PHP 动态图像处理 378

16.1	PHP中GD库的使用	378
16.1.1	画布管理	380
16.1.2	设置颜色	380
16.1.3	生成图像	381
16.1.4	绘制图像	381
16.1.5	在图像中绘制文字	383
16.2	设计经典验证码类	386
16.2.1	设计验证码类	386
16.2.2	应用验证码类的实例对象	389
16.2.3	表单中应用验证码	389
16.2.4	实例演示	390
16.3	PHP图片处理	390
16.3.1	图片背景管理	390
16.3.2	图片缩放	392
16.3.3	图片裁剪	393
16.3.4	添加图片水印	395
16.3.5	图片旋转和翻转	396
16.4	设计经典的图像处理类	398
16.4.1	需求分析	398
16.4.2	程序设计	399
16.4.3	图像处理类代码实现	399
16.4.4	图像处理类的应用过程	404
16.5	小结	406
	本章必须掌握的知识点	406
	本章需要了解的内容	406
	本章需要拓展的内容	406

第17章 MySQL 数据库概述 408

17.1	数据库的应用	408
17.1.1	数据库在Web开发中的重要地位	409
17.1.2	为什么PHP会选择MySQL作为自己 的黄金搭档	409
17.1.3	PHP和MySQL的合作方式	409
17.1.4	结构化查询语言SQL	410
17.2	MySQL数据库的常见操作	411
17.2.1	MySQL数据库的连接与关闭	411
17.2.2	创建新用户并授权	412
17.2.3	创建数据库	412
17.2.4	创建数据表	413
17.2.5	数据表内容的简单管理	414
17.3	小结	416
	本章必须掌握的知识点	416

第18章 MySQL 数据表的设计 417

18.1	数据表 (Table)	417
18.2	数据值和列类型	418
18.2.1	数值类的数据列类型	418
18.2.2	字符串类数据列类型	419
18.2.3	日期和时间型数据列类型	420
18.2.4	NULL值	421
18.2.5	类型转换	421
18.3	数据字段属性	421
18.4	数据表对象管理	422
18.4.1	创建表 (CREATE TABLE)	422
18.4.2	修改表 (ALTER TABLE)	423
18.4.3	删除表 (DROP TABLE)	424
18.5	数据表的类型及存储位置	425
18.5.1	MyISAM数据表	425
18.5.2	InnoDB数据表	425
18.5.3	如何选择InnoDB还是MyISAM表 类型	425
18.5.4	数据表的储存位置	426
18.6	数据表的默认字符集	426

18.6.1	字符集	427
18.6.2	字符集支持原理	427
18.6.3	创建数据对象时修改字符集	428
18.7	创建索引	428
18.7.1	主键索引 (PRIMARY KEY)	428
18.7.2	唯一索引 (UNIQUE)	429
18.7.3	常规索引 (INDEX)	430
18.7.4	全文索引 (FULLTEXT)	430
18.8	规范化	431
18.8.1	起点	431
18.8.2	第一范式	432
18.8.3	第二范式	432
18.8.4	第三范式	434
18.8.5	规范化理论	435
18.9	数据库的设计技巧	436
18.9.1	数据库设计要求	436
18.9.2	起名字的技巧	436
18.9.3	数据库具体设计工作中的技巧	436
18.10	小结	437
	本章必须掌握的知识点	437
	本章需要了解的内容	437
	本章需要拓展的内容	437

第 19 章 SQL 语句设计 438

19.1	操作数据表中的数据记录 (DML)	438
19.1.1	使用INSERT语句向数据表中添加数据	438
19.1.2	使用UPDATE语句更新数据表中已存在的数据	439
19.1.3	使用DELETE语句删除数据表中不需要的数据记录	440
19.2	通过DQL命令查询数据表中的数据	441
19.2.1	选择特定的字段	441
19.2.2	使用AS子句为字段取别名	442
19.2.3	DISTINCT关键字的使用	442
19.2.4	在SELECT语句中使用表达式的列	443
19.2.5	使用WHERE子句按条件检索	444
19.2.6	根据空值 (NULL) 确定检索条件	445

19.2.7	使用BETWEEN AND进行范围比较查询	445
19.2.8	使用IN进行范围比对查询	445
19.2.9	使用LIKE进行模糊查询	446
19.2.10	多表查询 (连接查询)	446
19.2.11	嵌套查询 (子查询)	449
19.2.12	使用ORDER BY对查询结果排序	449
19.2.13	使用LIMIT限定结果行数	450
19.2.14	使用统计函数	450
19.2.15	使用GROUP BY对查询结果分组	451
19.3	查询优化	452
19.4	小结	454
	本章必须掌握的知识点	454
	本章需要拓展的内容	454
	本章的学习建议	454

第 20 章 PHP 访问 MySQL 的扩展函数 455

20.1	PHP访问MySQL数据库服务器的流程	455
20.2	在PHP脚本中连接MySQL服务器	457
20.2.1	在PHP程序中选择已创建的数据库	458
20.2.2	执行SQL命令	458
20.2.3	在PHP脚本中处理SELECT查询结果集	460
20.3	设计完美分页类	462
20.3.1	需求分析	462
20.3.2	程序设计	462
20.3.3	完美分页类的代码实现	463
20.3.4	分页类的应用过程	468
20.4	管理books表实例	470
20.4.1	需求分析	470
20.4.2	程序设计	471
20.5	PHP的mysqli扩展介绍	477
20.5.1	启用mysqli扩展模块	478
20.5.2	mysqli扩展接口的应用概述	479
20.6	小结	480
	本章必须掌握的知识点	480

本章需要了解的内容	480
本章需要拓展的内容	480
本章的学习建议	480

第 21 章 数据库抽象层 PDO 481

21.1 PDO所支持的数据库	481
21.2 PDO的安装	482
21.3 创建PDO对象	483
21.3.1 以多种方式调用构造方法	484
21.3.2 PDO对象中的成员方法	486
21.4 使用PDO对象	487
21.4.1 调整PDO的行为属性	487
21.4.2 PDO处理PHP程序和数据库之间的数据类型转换	487
21.4.3 PDO的错误处理模式	488
21.4.4 使用PDO执行SQL语句	489
21.5 PDO对预处理语句的支持	491
21.5.1 了解PDOStatement对象	491
21.5.2 准备语句	492
21.5.3 绑定参数	493
21.5.4 执行准备好的查询	494
21.5.5 获取数据	495
21.5.6 大数据对象的存取	499
21.6 PDO的事务处理	499
21.6.1 MySQL的事务处理	500
21.6.2 构建事务处理的应用程序	500
21.7 小结	502
本章必须掌握的知识点	502
本章需要了解的内容	502
本章需要拓展的内容	502

第 5 部分 PHP 开发高级篇

第 22 章 MemCache 管理与应用 504

22.1 MemCache概述	504
22.1.1 初识MemCache	504

22.1.2 MemCache在Web中的应用	505
-------------------------	-----

22.2 memcached的安装及管理 507

22.2.1 Linux下安装MemCache软件	507
22.2.2 Windows下安装memcached软件	507
22.2.3 memcached服务器的管理	508

22.3 使用Telnet作为memcached的客户端管理 509

22.3.1 连接memcached服务器	509
22.3.2 基本的memcached客户端命令	509
22.3.3 查看当前memcached服务器的运行状态信息	510
22.3.4 数据管理指令	510

22.4 PHP的memcached管理接口 511

22.4.1 安装PHP中的MemCache应用程序扩展接口	512
22.4.2 MemCache应用程序扩展接口	513
22.4.3 MemCache的实例应用	518

22.5 memcached服务器的安全防护 519

22.6 小结	519
本章必须掌握的知识点	519
本章需要了解的内容	520
本章需要拓展的内容	520

第 23 章 会话控制 521

23.1 为什么要使用会话控制 521

23.2 会话跟踪的方式 522

23.3 Cookie的应用 523

23.3.1 Cookie概述	523
23.3.2 向客户端计算机中设置Cookie	524
23.3.3 在PHP脚本中读取Cookie的资料内容	525
23.3.4 数组形态的Cookie应用	525
23.3.5 删除Cookie	526
23.3.6 基于Cookie的用户登录模块	526

23.4 Session的应用 528

23.4.1 Session概述	528
23.4.2 配置Session	529
23.4.3 Session的声明与使用	530

23.4.4	注册一个会话变量和读取Session	531
23.4.5	注销变量与销毁Session	531
23.4.6	Session的自动回收机制	533
23.4.7	传递Session ID	533
23.5	一个简单的邮件系统实例	536
23.5.1	为邮件系统准备数据	536
23.5.2	编码实现邮件系统	537
23.5.3	邮件系统执行说明	539
23.6	自定义Session处理方式	540
23.6.1	自定义Session的存储机制	540
23.6.2	使用数据库处理Session信息	543
23.6.3	使用memcached处理Session信息	546
23.7	小结	549
	本章必须掌握的知识点	549
	本章需要了解的内容	549
	本章需要拓展的内容	549

第 24 章 PHP 的模板引擎 Smarty 550

24.1	什么是模板引擎	550
24.2	自定义模板引擎	552
24.2.1	自定义模板引擎类	552
24.2.2	使用自己的模板引擎	554
24.2.3	应用自定义模板引擎的示例分析	556
24.3	选择Smarty模板引擎	559
24.4	安装Smarty及初始化配置	560
24.4.1	安装Smarty	561
24.4.2	初始化Smarty类库的默认设置	561
24.4.3	第一个Smarty的简单示例	564
24.5	Smarty的基本应用	566
24.5.1	PHP程序员常用和Smarty相关的操作	567
24.5.2	模板设计时美工的常用操作	568
24.6	Smarty模板设计的基本语法	569
24.6.1	模板中的注释	569
24.6.2	模板中的变量应用	569
24.6.3	模板中的函数应用	572
24.6.4	忽略Smarty解析	574
24.7	在Smarty模板中的变量应用	574

24.7.1	从配置文件中读取变量	575
24.7.2	在模板中使用保留变量	578
24.8	在Smarty模板中的变量调解器	580
24.8.1	变量调解器函数的使用方式	580
24.8.2	Smarty默认提供的变量调解器	581
24.8.3	自定义变量调解器插件	582
24.9	Smarty模板中自定义函数	585
24.9.1	为Smarty模板扩充函数插件	585
24.9.2	为Smarty模板扩充块函数插件	586
24.10	Smarty模板中的内置函数	587
24.10.1	变量声明	588
24.10.2	流程控制	589
24.10.3	声明和调用模板函数	592
24.10.4	数组遍历	593
24.10.5	Smarty提供的其他内置函数	598
24.11	Smarty的模板继承特性	599
24.11.1	使用{extends}函数实现模板继承	599
24.11.2	在子模板中覆盖父模板中的部分内容区域	600
24.11.3	合并子模板和父模板的{block}标签内容	601
24.12	Smarty的缓存控制	602
24.12.1	在Smarty中控制缓存	603
24.12.2	每个模板多个缓存	604
24.12.3	为缓存实例消除处理开销	605
24.12.4	清除缓存	605
24.12.5	关闭局部缓存	606
24.13	小结	606
	本章必须掌握的知识点	606
	本章需要了解的内容	606
	本章需要拓展的内容	607

第 25 章 MVC 模式与 PHP 框架 608

25.1	MVC模式在Web中的应用	608
25.1.1	MVC模式的工作原理	608
25.1.2	MVC模式的优缺点	609
25.2	PHP开发框架	610
25.2.1	什么是框架	611

25.2.2	为什么要用框架	611
25.2.3	框架和MVC设计模式的关系	612
25.2.4	比较流行的PHP框架	612
25.3	划分模块和操作	614
25.3.1	为项目划分模块	614
25.3.2	为模块设置操作	615
第 26 章 超轻量级 PHP 框架 BroPHP 616		
26.1	BroPHP框架概述	616
26.1.1	系统特点	616
26.1.2	环境要求	617
26.1.3	BroPHP框架源码的目录结构	617
26.2	单一入口	618
26.2.1	基于BroPHP框架的单一入口编写规则	618
26.3	部署项目应用目录	619
26.3.1	项目部署方式	620
26.3.2	URL访问	622
26.4	BroPHP框架的基本设置	623
26.4.1	默认开启	623
26.4.2	配置文件	623
26.4.3	内置函数	624
26.5	声明控制器 (Control)	625
26.5.1	控制器的声明 (模块)	625
26.5.2	操作的声明	626
26.5.3	页面跳转	627
26.5.4	重定向	628
26.6	设计视图 (View)	629
26.6.1	视图与控制器之间的交互	629
26.6.2	切换模板风格	630
26.6.3	模板文件的声明规则	630
26.6.4	display()用新用法	631
26.6.5	在模板中的几个常用变量应用	631
26.6.6	在PHP程序中定义资源位置	632
26.7	应用模型 (Model)	632
26.7.1	BroPHP数据库操作接口的特性	632
26.7.2	切换数据库驱动	633
26.7.3	声明和实例化Model	634

26.7.4	数据库的统一操作接口	637
26.8	自动验证	654
26.9	缓存设置	656
26.9.1	基于memcached缓存设置	656
26.9.2	基于Smarty的缓存机制	657
26.10	调试模式	658
26.11	内置扩展类库	659
26.11.1	分页类Page	659
26.11.2	验证码类Vcode	660
26.11.3	图像处理类Image	661
26.11.4	文件上传类FileUpload	662
26.12	自定义功能扩展	664
26.12.1	自定义扩展类库	664
26.12.2	自定义扩展函数库	664
26.13	小结	664
	本章必须掌握的知识点	664
	本章需要了解的内容	665

第 6 部分 项目开发篇

第 27 章 B/S 结构软件开发流程 668	
27.1	软件开发过程的划分
27.2	需求开发
27.2.1	需求分析流程
27.2.2	需求分析说明
27.2.3	输出
27.3	系统设计
27.3.1	系统设计流程
27.3.2	系统设计说明
27.4	编码测试
27.4.1	编码与测试流程
27.4.2	编码说明
27.4.3	结果测试说明
27.5	试运行
27.5.1	软件试运行流程
27.5.2	软件试运行说明
27.6	实施
27.6.1	软件实施流程

27.6.2	软件实施说明	680
27.7	验收	681
27.7.1	软件验收流程	681
27.7.2	软件验收说明	682
27.7.3	验收标准	683
27.8	服务与维护	683
27.8.1	责任人	683
27.8.2	收集信息	683
27.8.3	维护分析	684
27.8.4	软件维护	684
27.8.5	改进	684
27.8.6	输出	684
27.9	项目管理	684
27.9.1	软件项目的计划	685
27.9.2	软件项目的组织	687
27.9.3	项目小组组织形式	687
27.10	项目参考	688

第 28 章 需求分析说明书 689

28.1	文档介绍	689
28.1.1	编写说明	690
28.1.2	项目背景	690
28.1.3	读者对象	690
28.1.4	参考资料	690
28.1.5	术语与缩写解释	691
28.2	任务概述	691
28.2.1	产品的描述	691
28.2.2	系统目标	692
28.2.3	系统功能结构	692
28.2.4	系统流程图	692
28.3	业务描述	694
28.3.1	后台登录管理	694
28.3.2	后台操作界面管理	695
28.3.3	常规管理	698
28.3.4	公告管理	700
28.3.5	友情链接管理	702
28.3.6	相册管理	705
28.3.7	图片管理	708

28.3.8	栏目管理	710
28.3.9	文章管理	713
28.3.10	幻灯片管理	716
28.3.11	用户组管理	719
28.3.12	用户管理	721
28.3.13	前台首页管理	724
28.3.14	栏目列表管理	725
28.3.15	文章内容管理	727
28.3.16	文章搜索管理	728
28.3.17	登录注册管理	730
28.3.18	个人空间管理	731
28.3.19	消息管理	736
28.3.20	动态管理	739
28.4	系统运行环境	743
28.4.1	硬件环境	743
28.4.2	软件环境	743
28.5	需求设计评审	744

第 29 章 数据库设计说明书 745

29.1	引言	745
29.1.1	编写目的	746
29.1.2	背景	746
29.1.3	定义	746
29.1.4	参考资料	746
29.2	外部设计	746
29.2.1	标识符和状态	747
29.2.2	使用它的程序	747
29.2.3	约定	747
29.2.4	支持软件	747
29.3	结构设计	748
29.4	逻辑结构设计	754
29.4.1	ER图向关系模型的转化	754
29.4.2	确定关系模式	754
29.4.3	消除冗余	755
29.5	物理结构设计	755
29.5.1	设计数据表结构	755
29.5.2	创建数据表	760
29.5.3	数据表记录的输入	764

29.6 安全保密设计764

29.6.1 完整性764

29.6.2 数据库设计的其他问题765

第 30 章 程序设计说明书 766

30.1 引言766

30.1.1 编写目的766

30.1.2 背景767

30.1.3 定义767

30.1.4 使用技术767

30.1.5 参考资料767

30.2 系统的结构767

30.2.1 项目的目录结构768

30.2.2 模块结构768

30.2.3 程序结构769

30.3 用户管理模块设计说明774

30.3.1 功能774

30.3.2 流程逻辑774

30.3.3 接口775

30.3.4 存储分配775

30.3.5 注释设计775

30.3.6 限制条件775

30.3.7 测试计划776

30.3.8 尚未解决的问题776

30.3.9 获取添加用户的界面操作add()776

30.3.10 用户数据入库的操作insert()777

30.3.11 查询用户列表操作index()778

30.3.12 获取修改用户的界面操作mod()779

30.3.13 用户数据修改的操作update()780

30.3.14 删除用户操作del()781

附 录

附录 A 编码规范 784

A.1 绪论784

A.1.1 适用范围784

A.1.2 目标784

A.1.3 开发工具785

A.2 PHP的文件格式785

A.2.1 PHP开始和结束标记785

A.2.2 注释规范786

A.2.3 空行和空白786

A.2.4 字符串的使用787

A.2.5 命名原则788

A.2.6 语言结构791

A.2.7 其他规范细节793

A.3 MySQL设计规范794

A.3.1 数据表的设计794

A.3.2 索引设计原则795

A.3.3 SQL语句设计796

A.4 模板设计796

附录 B PHP 的安全和优化 798

B.1 网站安全Security798

B.1.1 安全配置PHP799

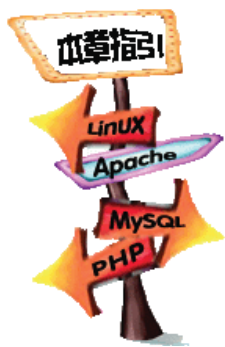
B.1.2 隐藏配置细节802

B.1.3	隐藏敏感数据.....	803
B.1.4	清理用户数据.....	804
B.1.5	数据加密.....	806
B.2	网站优化Optimize	807
B.2.1	PHP脚本级优化	807
B.2.2	使用代码优化工具.....	809
B.2.3	缓存加速.....	810
B.2.4	HTTP加速.....	810
B.2.5	启用GZIP内容压缩.....	810

附录 C、附录 D 见本书光盘

第16章

PHP 动态图像处理



PHP 不仅限于处理文本数据，还可以创建不同格式的动态图像，包括 GIF、PNG、JPG、WBMP 和 XPM 等。在 PHP 中，是通过使用 GD 扩展库实现对象图像的处理的，不仅可以创建新图像，而且可以处理已有的图像。更方便的是，PHP 不仅可以将动态处理后的图像以不同格式保存在服务器中，还可以直接将图像流输出到浏览器。例如验证码、股票走势图、电子相册等动态图像处理。

16.1

PHP 中 GD 库的使用

在 PHP 中，有一些简单的图像函数是可以直接使用的，但大多数要处理的图像，都需要在编译 PHP 时加上 GD 库。除了安装 GD 库之外，在 PHP 中还可能需其他的库，这可以根据需要支持哪些图像格式而定。GD 库可以在 <http://www.boutell.com/gd/> 免费下载，不同的 GD 版本支持的图像格式不完全一样，最新的 GD 库版本支持 GIF、JPEG、PNG、WBMP、XBM 等格式的图像文件，此外还支持一些如 FreeType、Type 1 等字体库。通过 GD 库中的函数可以完成各种点、线、几何图形、文本及颜色的操作和处理，也可以创建或读取多种格式的图像文件。

在 PHP 中，通过 GD 库处理图像的操作，都是先在内存中处理，操作完成以后再以文件流的方式，输出到浏览器或保存在服务器的磁盘中。创建一个图像应该完成如下所示的 4 个基本步骤。

(1) 创建画布：所有的绘图设计都需要在一个背景图片上完成，而画布实际上就是在内存中开辟的一块临时区域，用于存储图像的信息。以后的图像操作都将基于这个背景画布，该画布的管理就类似于我们在画画时使用的画布。

(2) 绘制图像：画布创建完成以后，就可以通过这个画布资源，使用各种画像函数设置图像的颜色、填充画布、画点、线段、各种几何图形，以及向图像中添加文本等。

(3) 输出图像：完成整个图像的绘制以后，需要将图像以某种格式保存到服务器指定的文件中，或将图像直接输出到浏览器上显示给用户。但在图像输出之前，一定要使用 `header()` 函数发送 Content-type



通知浏览器，这次发送的是图片不是文本。

(4) 释放资源：图像被输出以后，画布中的内容也不再有用。出于节约系统资源的考虑，需要及时清除画布占用的所有内存资源。

我们先来了解一个非常简单的创建图像脚本。在下面的脚本文件 image.php 中，按前面介绍的绘制图像的四个步骤，使用 GD 库动态输出一个扇形统计图。代码如下所示：

```
1 <?php
2 //创建画布，返回一个资源类型的变量$image.并在内存中开辟一块临时区域
3 $image = imagecreatetruecolor(100, 100); //创建画布的大小为100x100
4
5 //设置图像中所需的颜色，相当于在画画时准备的染料盒
6 $white = imagecolorallocate($image, 0xFF, 0xFF, 0xFF); //为图像分配颜色为白色
7 $gray = imagecolorallocate($image, 0xC0, 0xC0, 0xC0); //为图像分配颜色为灰色
8 $darkgray = imagecolorallocate($image, 0x90, 0x90, 0x90); //为图像分配颜色为暗灰色
9 $navy = imagecolorallocate($image, 0x00, 0x00, 0x80); //为图像分配颜色为深蓝色
10 $darknavy = imagecolorallocate($image, 0x00, 0x00, 0x50); //为图像分配颜色为暗深蓝色
11 $red = imagecolorallocate($image, 0xFF, 0x00, 0x00); //为图像分配颜色为红色
12 $darkred = imagecolorallocate($image, 0x90, 0x00, 0x00); //为图像分配颜色为暗红色
13
14 imagefill($image, 0, 0, $white); //为画布背景添充背景颜色
15 //动态制作3D 效果
16 for ($i = 60; $i > 50; $i--) { //循环10次画出立体效果
17     imagefilledarc($image, 50, $i, 100, 50, -160, 40, $darknavy, IMG_ARC_PIE);
18     imagefilledarc($image, 50, $i, 100, 50, 40, 75, $darkgray, IMG_ARC_PIE);
19     imagefilledarc($image, 50, $i, 100, 50, 75, 200, $darkred, IMG_ARC_PIE);
20 }
21
22 imagefilledarc($image, 50, 50, 100, 50, -160, 40, $navy, IMG_ARC_PIE); //画一椭圆弧且填充
23 imagefilledarc($image, 50, 50, 100, 50, 40, 75, $gray, IMG_ARC_PIE); //画一椭圆弧且填充
24 imagefilledarc($image, 50, 50, 100, 50, 75, 200, $red, IMG_ARC_PIE); //画一椭圆弧且填充
25
26 imagestring($image, 1, 15, 55, '34.7%', $white); //水平地画一行字符串
27 imagestring($image, 1, 45, 35, '55.5%', $white); //水平地画一行字符串
28
29 // 向浏览器中输出一个GIF格式的图片
30 header('Content-type: image/png'); //使用头函数告诉浏览器以图像方式处理以下输出
31 imagepng($image); //向浏览器输出
32 imagedestroy($image); //销毁图像释放资源
```

直接通过浏览器请求该脚本，或是将该脚本所在的 URL，赋给 HTML 中 IMG 标记的 src 属性，都可以获取动态输出的图像结果，如图 16-1 所示。

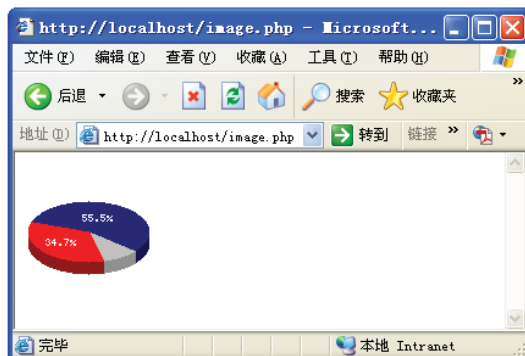


图 16-1 使用 PHP 的 GD 库动态绘制统计图

16.1.1 画布管理

使用 PHP 的 GD 库处理图像时，必须对画布进行管理。创建画布就是在内存中开辟一块存储区域，以后在 PHP 中对图像的所有操作都是基于这个图布处理的，图布就是一个图像资源。在 PHP 中，可以使用 `imagecreate()` 和 `imageCreateTrueColor()` 两个函数创建指定的画布。这两个函数的作用是一致的，都是建立一个指定大小的画布，它们的原型如下所示：

```
resource imagecreate ( int $x_size, int $y_size )           //新建一个基于调色板的图像
resource imagecreatetruecolor ( int $x_size, int $y_size ) //新建一个真彩色图像
```

虽然这两个函数都可以创建一个新的画布，但各自能够容纳颜色的总数是不同的。`imageCreate()` 函数可以创建一个基于普通调色板的图像，通常支持 256 色。而 `imageCreateTrueColor()` 函数可以创建一个真彩色图像，但该函数不能用于 GIF 文件格式。当画布创建后，返回一个图像标识符，代表了一幅宽度为 `$x_size` 和高度为 `$y_size` 的空白图像引用句柄。在后续的绘图过程中，都需要使用这个资源类型的句柄。例如，可以通过调用 `imagex()` 和 `imagey()` 两个函数获取图像的大小。代码如下所示：

```
1 <?php
2 $img = imagecreatetruecolor(300, 200);           //创建一个300*200的画布
3 echo imagesx($img);                             //输出画布宽度300
4 echo imagey($img);                             //输出画布高度200
```

另外，画布的引用句柄如果不再使用，一定要将这个资源销毁，释放内存与该图像的存储单元。画布的销毁过程非常简单，调用 `imagedestroy()` 函数就可以实现。其语法格式如下所示：

```
bool imagedestroy ( resource $image )           //销毁一图像
```

如果该方法调用成功，就会释放与参数 `$image` 关联的内存。其中参数 `$image` 是由图像创建函数返回的图像标识符。

16.1.2 设置颜色

在使用 PHP 动态输出美丽图像的同时，也离不开颜色的设置，就像画画时需要使用调色板一样。设置图像中的颜色，需要调用 `imageColorAllocate()` 函数完成。如果在图像中需要设置多种颜色，只要多次调用该函数即可。该函数的原型如下所示：

```
int imagecolorallocate ( resource $image, int $red, int $green, int $blue ) //为一幅图像分配颜色
```

该函数会返回一个标识符，代表了由给定的 RGB 成分组成的颜色。参数 `$red`、`$green` 和 `$blue` 分别是所需要的颜色的红、绿、蓝成分。这些参数是 0 到 255 的整数或者十六进制的 0x00 到 0xFF。第一个参数 `$image` 是画布图像的句柄，该函数必须调用 `$image` 所代表的图像中的颜色。但要注意，如果是使用 `imagecreate()` 函数建立的画布，则第一次对 `imagecolorallocate()` 函数的调用，会给基于调色板的图像填充背景色。该函数的使用代码如下所示：



```
1 <?php
2 $im = imagecreate(100, 100); //为设置颜色函数提供一个画布资源
3 //背景设为红色
4 $background = imagecolorallocate($im, 255, 0, 0); //第一次调用即为画布设置背景颜色
5 //设定一些颜色
6 $white = imagecolorallocate($im, 255, 255, 255); //返回由十进制整数设置为白色的标识符
7 $black = imagecolorallocate($im, 0, 0, 0); //返回由十进制整数设置为黑色的标识符
8 //十六进制方式
9 $white = imagecolorallocate($im, 0xFF, 0xFF, 0xFF); //返回由十六进制整数设置为白色的标识符
10 $black = imagecolorallocate($im, 0x00, 0x00, 0x00); //返回由十六进制整数设置为黑色的标识符
```

16.1.3 生成图像

使用 GD 库中提供的函数动态绘制完成图像以后，就需要输出到浏览器或者将图像保存起来。在 PHP 中，可以将动态绘制完成的画布，直接生成 GIF、JPEG、PNG 和 WBMP 四种图像格式。可以通过调用下面四个函数生成这些格式的图像：

bool imagegif (resource \$image [, string \$filename])	//以 GIF 格式将图像输出
bool imagejpeg(resource \$image [, string \$filename [, int \$quality]])	//以 JPEG 格式将图像输出
bool imagepng (resource \$image [, string \$filename])	//以 PNG 格式将图像输出
bool imagewbmp (resource \$image [, string \$filename [, int \$foreground]])	//以 WBMP 格式将图像输出

以上四个函数的使用类似，前两个参数的使用是相同的。第一个参数 \$image 为必选项，是前面介绍的图像引用句柄。如果不为这些函数提供其他参数，访问时则直接将原图像流输出，并在浏览器中显示动态输出的图像。但一定要在输出之前，使用 header() 函数发送标头信息，用来通知浏览器使用正确的 MIME 类型对接收的内容进行解析，让它知道我们发送的是图片而不是文本的 HTML。以下代码段通过自动检测 GD 库支持的图像类型，来写出移植性更好的 PHP 程序。如下所示：

```
1 <?php
2 if (function_exists("imagegif")) { //判断生成GIF格式图像的函数是否存在
3     header("Content-type: image/gif"); //发送标头信息设置MIME类型为image/gif
4     imagegif($im); //以GIF格式将图像输出到浏览器
5 } elseif (function_exists("imagejpeg")) { //判断生成JPEG格式图像的函数是否存在
6     header("Content-type: image/jpeg"); //发送标头设置MIME类型为image/jpeg
7     imagejpeg($im, "", 0.5); //以JPEG格式将图像输出到浏览器
8 } elseif (function_exists("imagepng")) { //判断生成PNG格式图像的函数是否存在
9     header("Content-type: image/png"); //发送标头设置MIME类型为image/png
10    imagepng($im); //以PNG格式将图像输出到浏览器
11 } elseif (function_exists("imagewbmp")) { //判断生成WBMP格式图像的函数是否存在
12     header("Content-type: image/vnd.wap.wbmp"); //设置MIME类型为image/vnd.wap.wbmp
13     imagewbmp($im); //以WBMP格式将图像输出到浏览器
14 } else { //如果没有可以使用的生成图像函数
15     die("在PHP服务器中，不支持图像"); //则PHP不支持图像操作，退出
16 }
```

如果希望将 PHP 动态绘制的图像保存在本地服务器上，则必须在第二个可选参数中指定一个文件名字符串。这样，不仅不会将图像直接输出到浏览器，也不需要使用 header() 函数发送标头信息。

如果使用 imageJPEG() 函数生成 JPEG 格式的图像，还可以通过第三个可选参数 \$quality 指定 JPEG 格式图像的品质，该参数可以提供的值是从 0（最差品质，但文件最小）到 100（最高品质，文件也最大）的整数，默认值为 75。也可以为函数 imageWBMP() 提供第三个可选参数 \$foreground，指定图像的前景颜色，默认颜色值为黑色。

16.1.4 绘制图像

在 PHP 中绘制图像的函数非常丰富，包括点、线、各种几何图形等可以想象出来的平面图形，都可以通过 PHP 中提供的各种画图函数完成。我们在这里只介绍一些常用的图像绘制，如果使用我们没有介绍过的函数，可以参考手册实现。另外，这些图形绘制函数都需要使用画布资源，并在画布中的位置通过坐标（原点是该画布左上角的起始位置，以像素为单位，沿着 X 轴正方向向右延伸，Y 轴正方向向下延伸）决定，而且还可以通过函数中的最后一个参数，设置每个图形的颜色。画布中的坐标系如图 16-2 所示。

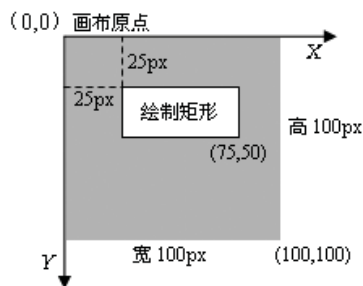


图 16-2 使用 PHP 绘制图像的坐标演示

1. 图形区域填充

通过 PHP 仅仅绘制出只有边线的几何图形是不够的，还可以使用对应的填充函数，完成图形区域的填充。除了每个图形都有对应的填充函数之外，还可以使用 `imageFill()` 函数实现区域填充。

该函数的语法格式如下：

```
bool imagefill ( resource $image, int $x, int $y, int $color ) //区域填充
```

该函数在参数 `$image` 代表的图像上，相对于图像左上角 (0,0) 坐标处，从坐标 (`$x`, `$y`) 处用参数 `$color` 指定的颜色执行区域填充。与坐标 (`$x`, `$y`) 点颜色相同且相邻的点都会被填充。例如在下面的示例中，将画布的背景设置为红色。代码如下所示：

```
1 <?php
2 $im = imagecreatetruecolor(100, 100); //创建100*100大小的画布
3 $red = imagecolorallocate($im, 255, 0, 0); //设置一个颜色变量为红色
4
5 imagefill($im, 0, 0, $red); //将背景设为红色
6
7 header('Content-type: image/png'); //通知浏览器这不是文本而是一个图片
8 imagepng($im); //生成PNG格式的图片输出给浏览器
9
10 imagedestroy($im); //销毁图像资源，释放画布占用的内存空间
```

2. 绘制点和线

画点和线是绘制图像中最基本的操作，如果灵活使用，可以通过它们绘制出千变万化的图像。在 PHP 中，使用 `imageSetPixel()` 函数在画布中绘制一个单一像素的点，并且可以设置点的颜色。其函数的原型如下所示：

```
bool imagesetpixel ( resource $image, int $x, int $y, int $color ) //画一个单一像素
```

该函数在第一个参数 `$image` 中提供的画布上，距离原点分别为 `$x` 和 `$y` 的坐标位置，绘制一个颜色为 `$color` 的一个像素点。理论上使用画点函数便可以画出所需要的所有图形，也可以使用其他的绘图函数。如果需要绘制一条线段，可以使用 `imageline()` 函数，其语法格式如下所示：

```
bool imageline ( resource $image, int $x1, int $y1, int $x2, int $y2, int $color ) //画一条线段
```

我们都知道两点确定一条线段，所以该函数使用 `$color` 颜色在图像 `$image` 中，从坐标 (`$x1`, `$y1`) 开始到 (`$x2`, `$y2`) 坐标结束画一条线段。



3. 绘制矩形

可以使用 `imageRectangle()` 函数绘制矩形，也可以通过 `imageFilledRectangle()` 函数绘制一个矩形并填充。这两个函数的语法格式如下：

```
bool imageRectangle ( resource $image, int $x1, int $y1, int $x2, int $y2, int $color )    //画一个矩形
bool imagefilledrectangle ( resource image, int $x1, int $y1, int $x2, int $y2, int $color )    //画一矩形并填充
```

这两个函数的行为类似，都是在 `$image` 图像中画一个矩形，只不过前者是使用 `$color` 参数指定矩形的边线颜色，而后者则是使用这个颜色填充矩形。相对于图像左上角的 `(0, 0)` 位置，矩形的左上角坐标为 `($x1, $y1)`，右下角坐标为 `($x2, $y2)`。

4. 绘制多边形

可以使用 `imagePolygon()` 函数绘制一个多边形，也可以通过 `imageFilledPolygon()` 函数绘制一个多边形并填充。这两个函数的语法格式如下：

```
bool imagepolygon (resource $image, array $points, int $num_points, int $color )    //画一个多边形
bool imagefilledpolygon (resource $image, Sarray $points, int $num_points, int $color)    //画一多边形并填充
```

这两个函数的行为类似，都是在 `$image` 图像中画一个多边形，只不过前者是使用 `$color` 参数指定多边形的边线颜色，而后者则是使用这个颜色填充多边形。第二个参数 `$points` 是一个 PHP 数组，包含了多边形的各个顶点坐标。即 `points[0]=x0`, `points[1]=y0`, `points[2]=x1`, `points[3]=y1`，依此类推。第三个参数 `$num_points` 是顶点的总数，必须大于 3。

5. 绘制椭圆

可以使用 `imageEllipse()` 函数绘制一个椭圆，也可以通过 `imageFilledEllipse()` 函数绘制一个椭圆并填充。这两个函数的语法格式如下：

```
bool imageellipse ( resource $image, int $cx, int $cy, int $w, int $h, int $color )    //画一个椭圆
bool imagefilledellipse ( resource $image, int $cx, int $cy, int $w, int $h, int $color )    //画一个椭圆填充
```

这两个函数的行为类似，都是在 `$image` 图像中画一个椭圆，只不过前者是使用 `$color` 参数指定椭圆的边线颜色，而后者则是使用它填充颜色。相对于画布左上角坐标 `(0, 0)`，以 `($cx, $cy)` 坐标为中心画一个椭圆，参数 `$w` 和 `$h` 分别指定了椭圆的宽和高。如果成功则返回 `TRUE`，失败则返回 `FALSE`。

6. 绘制弧线

前面介绍的 3D 扇形统计图示例，就是使用绘制填充圆弧的函数实现的。可以使用 `imageArc()` 函数绘制一条弧线，以及圆形和椭圆形。这个函数的语法格式如下：

```
bool imagearc ( resource $image, int $cx, int $cy, int $w, int $h, int $s, int $e, int $color )    //画椭圆弧
```

相对于画布左上角坐标 `(0, 0)`，该函数以 `($cx, $cy)` 坐标为中心，在 `$image` 所代表的图像中画一个椭圆弧。其中参数 `$w` 和 `$h` 分别指定了椭圆的宽度和高度，起始点和结束点以 `$s` 和 `$e` 参数以角度指定。`0°` 位于三点钟位置，以顺时针方向绘画。如果要绘制一个完整的圆形，首先要将参数 `$w` 和 `$h` 设置为相等的值，然后将起始角度 `$s` 指定为 0，结束角度 `$e` 指定为 360。如果需要绘制填充圆弧，可以查询 `imageFilledArc()` 函数使用。

16.1.5 在图像中绘制文字

在图像中显示的文字也需要按坐标位置画上去。在 PHP 中不仅支持比较多的字体库，而且提供了非常灵活的文字绘制方法。例如，在图像中绘制缩放、倾斜、旋转的文字等。可以使用 `imageString()`、`imageStringUP()` 或 `imageChar()` 等函数使用内置的字体文字绘制到图像中。这些函数的原型如下所示：

<code>bool imagestring (resource \$image, int \$font, int \$x, int \$y, string \$s, int \$color)</code>	//水平地画一行字符串
<code>bool imagestringup (resource \$image, int \$font, int \$x, int \$y, string \$s, int \$color)</code>	//垂直地画一行字符串
<code>bool imagechar (resource \$image, int \$font, int \$x, int \$y, char \$c, int \$color)</code>	//水平地画一个字符
<code>bool imagecharup (resource \$image, int \$font, int \$x, int \$y, char \$c, int \$color)</code>	//垂直地画一个字符

在上面列出来的四个函数中，前两个函数 `imageString()` 和 `imageStringUP()` 分别用来向图像中水平和垂直输出一行字符串，而后两个函数 `imageChar()` 和 `imageCharUP()` 分别用来向图像中水平和垂直输出一个字符。虽然这四个函数有所差异，但调用方式类似。它们都是在 `$image` 图像中绘制由第五个参数指定的字符串或字符，绘制的位置都是从坐标 `($x, $y)` 开始输出。如果是水平地画一行字符串则是从左向右输出，而垂直地画一行字符串则是从下而上输出。这些函数都可以通过最后一个参数 `$color` 给出文字的颜色。第二个参数 `$font` 则给出了文字字体标识符，其值为整数 1、2、3、4 或 5，则是使用内置的字体，数字越大则输出的文字尺寸就越大。下面是在一个图像中输出文字的示例：

```

1 <?php
2     $im = imagecreate(150, 150); //创建一个150*150的画布
3
4     $bg = imagecolorallocate($im, 255, 255, 255); //设置画布的背景为白色
5     $black = imagecolorallocate($im, 0, 0, 0); //设置一个颜色变量为黑色
6
7     $string = "LAMPBrother"; //在图像中输出的字符串
8
9     imageString($im, 3, 28, 70, $string, $black); //水平将字符串输到图像中
10    imageStringUp($im, 3, 59, 115, $string, $black); //垂直由下而上输到图像中
11    for($i=0,$j=strlen($string); $i<strlen($string); $i++,$j--){ //循环单个字符输到图像中
12        imageChar($im, 3, 10*($i+1), 10*($j+2), $string[$i], $black); //向下倾斜输出每个字符
13        imageCharUp($im, 3, 10*($i+1), 10*($j+2), $string[$i], $black); //向上倾斜输出每个字符
14    }
15
16    header('Content-type: image/png'); //设置输出的头部标识符
17    imagepng($im); //输出PNG格式的图片

```

直接请求该脚本在浏览器中显示的图像如图 16-3 所示。

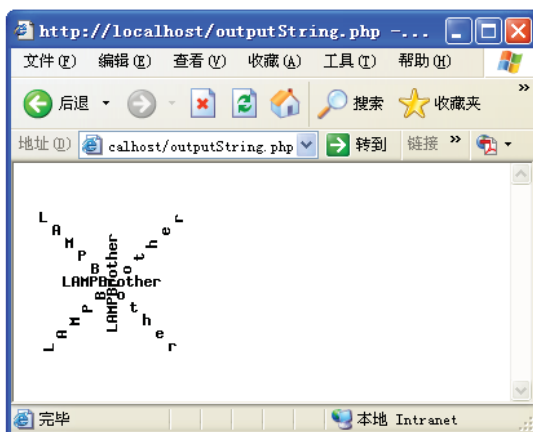


图 16-3 使用 PHP 的 GD 库绘制内置字体

除了通过上面介绍的四个函数输出内置的字体外，还可以使用 `imageTtfText()` 函数，输出一种可以缩放的与设备无关的 TrueType 字体。TrueType 是用数学函数描述字体轮廓外形，既可以用做打印字体，又可以用做屏幕显示，各种操作系统都可以兼容这种字体。由于它是由指令对字形进行描述，因此它与分辨率无关，输出时总是按照打印机的分辨率输出。无论放大或缩小，字符总是光滑的，不会有锯齿出现。例如在 Windows 系统中，字体库所在的文件夹 `C:\WINDOWS\Fonts` 下，对 TrueType 字体都有标注，如 `simsum.ttf` 为 TrueType 字体中的“宋体”。`imageTtfText()` 函数的原型如下所示：

```
array imageTtfText(resource $image, float $size, float $angle, int $x, int $y, int $color, string $fontfile, string $text)
```

该函数需要多个参数，其中参数 `$image` 需要提供一个图像资源。参数 `$size` 用来设置字体大小，根据 GD 库版本不同，应该以像素大小指定（GD1）或点大小（GD2）。参数 `$angle` 是角度制表示的角度， 0° 为从左向右读的文本，更高数值表示逆时针旋转。例如， 90° 表示从下向上读的文本。并由 `($x, $y)` 两个参数所表示的坐标，定义了第一个字符的基本点，大概是字符的左下角。而这和 `imageString()` 函数有所不同，其 `($x, $y)` 坐标定义了第一个字符的左上角。参数 `$color` 指定颜色索引。使用负的颜色索引值具有关闭防锯齿的效果。参数 `$fontfile` 是想要使用的 TrueType 字体的路径。根据 PHP 所使用的 GD 库的不同，当 `fontfile` 没有以 “/” 开头时则 “.ttf” 将被加到文件名之后，并且会在库定义字体路径中尝试搜索该文件名。最后一个参数 `$text` 指定需要输出的文本字符串，可以包含十进制数字化字符表示（形式为：Ŭ）来访问字体中超过位置 127 的字符。UTF-8 编码的字符串可以直接传递。如果字符串中使用的某个字符不被字体支持，一个空心矩形将替换该字符。

`imageTtfText()` 函数返回一个含有 8 个单元的数组，表示了文本外框的四个角，顺序为左下角，右下角，右上角，左上角。这些点是相对于文本的而和角度无关，因此“左上角”指的是以水平方向看文字时其左上角。我们通过在下列中的脚本，生成一个白色的 `400×30` 像素的 PNG 图像，其中有黑色（带灰色阴影）“宋体”字体写的“LAMP 兄弟连——无兄弟，不编程！”。代码如下所示：


```

1 <?php
2 $im = imagecreatetruecolor(400, 30); //创建400 300像素大小的画布
3
4 $white = imagecolorallocate($im, 255, 255, 255); //创建白色
5 $grey = imagecolorallocate($im, 128, 128, 128); //创建灰色
6 $black = imagecolorallocate($im, 0, 0, 0); //创建黑色
7
8 imagefilledrectangle($im, 0, 0, 399, 29, $white); //输出一个使用白色填充的矩形作为背景
9
10 //如果有中文输出, 需要将其转码, 转换为UTF-8的字符串才可以直接传递
11 $text = iconv("GB2312", "UTF-8", "LAMP兄弟连——无兄弟, 不编程!");
12 //指定字体, 将系统中与simsum.ttc对应的字体复制到当前目录下
13 $font = 'simsum.ttc';
14
15 imagettftext($im, 20, 0, 12, 21, $grey, $font, $text); //输出一个灰色的字符串作为阴影
16 imagettftext($im, 20, 0, 10, 20, $black, $font, $text); //在阴影之上输出一个黑色的字符串
17
18 header("Content-type: image/png"); //通知浏览器将输出格式为PNG的图像
19 imagepng($im); //向浏览器中输出PNG格式的图像
20
21 imagedestroy($im); //销毁资源, 释放内存占用的空间

```

直接请求该脚本在浏览器中显示的图像如图 16-4 所示。



图 16-4 使用 PHP 的 GD 库绘制与设备无关的 TrueType 字体

16.2 设计经典验证码类

验证码就是将一串随机产生的数字或符号, 动态生成一幅图片。再在图片中加上一些干扰像素, 只要用户可以通过肉眼识别其中的信息即可。并在表单提交时使用, 只有审核成功后才能使用某项功能。很多地方都需要使用验证码, 经常出现在用户注册、登录或者在网上发帖子时。因为你的 Web 站有时会碰到客户机恶意攻击, 其中一种很常见的攻击手段就是身份欺骗。它通过在客户端脚本写入一些代码, 然后利用其客户机在网站, 论坛反复登录。或者攻击者创建一个 HTML 窗体, 其窗体包含了你注册窗体或发帖窗体等相同的字段。然后利用“http-post”传输数据到服务器, 服务器就会执行相应的创建账户, 提交垃圾数据等操作。如果服务器本身不能有效验证并拒绝此非法操作, 它会很严重耗费其系统资源, 降低网站性能甚至使程序崩溃。验证码就是为了防止有人利用机器人自动批量注册、对特定的注册用户用特定程序暴力破解方式进行不断的登录、灌水等。因为验证码是一个混合了数字或符号的图片, 人眼看起来都费劲, 机器识别起来就更困难了。这样可以确保当前访问者是一个人而非机器。



16.2.1 设计验证码类

我们通过本章节中介绍的图像处理内容，设计一个验证码类 `Vcode`。将该类声明在文件 `vcode.class.php` 中，并通过面向对象的特性将一些实现的细节封装在该类中。只要在创建对象时，为构造方法提供三个参数，包括创建验证码图片的宽度、高度及验证码字母个数，就可以成功创建一个验证码类的对象。默认验证码的宽度为 80 个像素，高度为 20 个像素，由 4 个字母或数字组成。该类的声明代码如下所示：

```
1 <?php
2 /**
3     file: vcode.class.php
4     验证码类, 类名Vcode
5 */
6 class Vcode {
7     private $width;           // 验证码图片的宽度
8     private $height;          // 验证码图片的高度
9     private $codeNum;         // 验证码字符的个数
10    private $disturbColorNum;  // 干扰元素数量
11    private $checkCode;        // 验证码字符
12    private $image;            // 验证码资源
13
14    /**
15     * 构造方法用来实例化验证码对象，并为一些成员属性初使化
16     * @param int $width 设置验证码图片的宽度，默认宽度值为80像素
17     * @param int $height 设置验证码图片的高度，默认高度值为20像素
18     * @param int $codeNum 设置验证码中字母和数字的个数，默认个数为4个
19     */
20    function __construct($width=80, $height=20, $codeNum=4) {
21        $this->width = $width;
22        $this->height = $height;
23        $this->codeNum = $codeNum;
24        $number = floor($height*$width/15);
25        if($number > 240-$codeNum)
26            $this->disturbColorNum = 240-$codeNum;
27        else
28            $this->disturbColorNum = $number;
29        $this->checkCode = $this->createCheckCode();
30    }
```

```

31
32  /**
33  * 用于输出验证码图片，也向服务器的SESSION中保存了验证码，使用echo 输出对象即可
34  */
35  function __toString(){
36      /* 加到session中，存储下标为code */
37      $_SESSION["code"] = strtoupper($this->checkCode);
38      $this->outImg();
39      return '';
40  }
41
42  /* 内部使用的私有方法，用于输出图像 */
43  private function outImg(){
44      $this->getCreateImage();
45      $this->setDisturbColor();
46      $this->outputText();
47      $this->outputImage();
48  }
49
50  /* 内部使用的私有方法，用来创建图像资源，并初使化背景 */
51  private function getCreateImage(){
52      $this->image = imagecreatetruecolor($this->width,$this->height);
53
54      $backColor = imagecolorallocate($this->image, rand(225,255),rand(225,255),rand(225,255));
55
56      @imagefill($this->image, 0, 0, $backColor);
57
58      $border = imageColorAllocate($this->image, 0, 0, 0);
59      imageRectangle($this->image,0,0,$this->width-1,$this->height-1,$border);
60  }
61
62  /* 内部使用的私有方法，随机生成用户指定个数的字符串,去掉了容易混淆的字符oOLlZ和数字012 */
63  private function createCheckCode(){
64      $code="3456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
65      for($i=0; $i<$this->codeNum; $i++) {
66          $char = $code[rand(0,strlen($code)-1)];
67
68          $ascii .= $char;
69      }
70      return $ascii;
71  }
72

```



```
73      /* 内部使用的私有方法，设置干扰像素，向图像中输出不同颜色的点 */
74      private function setDisturbColor() {
75          for($i=0; $i <= $this->disturbColorNum; $i++) {
76              $color = imagecolorallocate($this->image, rand(0,255), rand(0,255), rand(0,255));
77              imagesetpixel($this->image,rand(1,$this->width-2),rand(1,$this->height-2),$color);
78          }
79
80          for($i=0; $i<10; $i++){
81              $color=imagecolorallocate($this->image,rand(0,255),rand(0,255),rand(0,255));
82              imagearc($this->image,rand(-10,$this->width),rand(-10,$this->height),rand(30,300),
83                      rand(20,200),55,44,$color);
84          }
85
86      /* 内部使用的私有方法，随机颜色、随机摆放、随机字符串向图像中输出 */
87      private function outputText() {
88          for ($i=0; $i<=$this->codeNum; $i++) {
89              $fontcolor = imagecolorallocate($this->image, rand(0,128), rand(0,128), rand(0,128));
90              $fontSize = rand(3,5);
91              $x = floor($this->width/$this->codeNum)*$i+3;
92              $y = rand(0,$this->height-imagefontheight($fontSize));
93              imagechar($this->image, $fontSize, $x, $y, $this->checkCode{$i}, $fontcolor);
94          }
95      }
96
97      /* 内部使用的私有方法，自动检测GD支持的图像类型，并输出图像 */
98      private function outputImage(){
99          if(imagetypes() & IMG_GIF){
100              header("Content-type: image/gif");
101              imagegif($this->image);
102          }elseif(imagetypes() & IMG_JPG){
103              header("Content-type: image/jpeg");
104              imagejpeg($this->image, "", 0.5);
105          }elseif(imagetypes() & IMG_PNG){
106              header("Content-type: image/png");
107              imagepng($this->image);
108          }elseif(imagetypes() & IMG_WBMP){
109              header("Content-type: image/vnd.wap.wbmp");
110              imagewbmp($this->image);
111          }else{
112              die("PHP不支持图像创建！");
113          }
114      }
115
116      /* 析构方法，在对象结束之前自动销毁图像资源释放内存 */
117      function __destruct(){
118          imagedestroy($this->image);
119      }
120  }
```

在上面的脚本中，虽然声明验证码类 Vcode 的代码比较多，但细节都被封装在类中，只要直接输出对象，就可以向客户端浏览器中输出一幅图片，并可以在浏览器表单中使用。另外本类自动获取验证码图片中的字符串，保存在服务的\$_SESSION["code"]中。在提交表单时，只有当用户在表单中输入验证码图片上显示的文字，并和服务端中保留的验证码字符串完全相同时，表单才可以提交成功。

注意：验证码在服务器端保存在\$_SESSION["code"]中，所以必须开启 session 会话才能使用该类，另外在服务器端存储时已经自动将验证码的内容全部转成了大写，所以在匹配时也要将客户端提交的验

验证码转成大写，以达到匹配时不区分大小写的目的。

16.2.2 应用验证码类的实例对象

在下面的脚本 `imgcode.php` 中，使用 `session_start()` 开启了用户会话控制（本书后面的章节有详细介绍），然后包含验证码类 `Vcode` 所在文件 `vcode.class.php`，创建该类对象并直接输出。就可以将随机生成的验证码图片发送出去，同时会自动将这个验证码字符串保存在服务器中一份。代码如下所示：

```
1 <?php
2     /**
3      * file:imgcode.php
4      * 用于请求时，通过验证码类的对象向客户端输出图片
5      */
6     session_start();                //开启SESSION,会使用$_SESSION["code"]在服务器中保存验证码
7
8     require_once('vcode.class.php'); //包含验证码所在的类文件
9     echo new Vcode();               //创建验证码对象，并直接被输出自动调用魔术__toString()方法
```

16.2.3 表单中应用验证码

在下面的脚本 `image.php` 中，包含用户输入表单和匹配验证码两部分。在表单中获取并显示验证码图片，如果验证码上的字符串看不清楚，还可以通过单击它重新获取一张。在表单中，按照验证码图片中显示的文字输出以后，提交时还会转到该脚本中验证。从客户端接收到的验证码，如果和服务中保留的验证码相同，则提交成功。代码如下所示：



```
1 <?php
2  /** file:image.php 用于输出用户操作表单和验证用户的输入 */
3  session_start(); //开启SESSION
4  if(isset($_POST['submit'])){ //判断用户提交后执行
5      /** 判断用户在表单中输入的字符串和验证码图片中的字符串是否相同 */
6      if(strtoupper(trim($_POST["code"])) == $_SESSION['code']){ //如果验证码输入成功
7          echo '验证码输入成功<br>'; //输出成功的提示信息
8      }else{ //如果验证码输入失败
9          echo '<font color="red">验证码输入错误! ! </font><br>'; //输出失败的输入信息
10     }
11 }
12 ?>
13 <html>
14     <head>
15         <title>Image</title>
16         <meta http-equiv="content-type" content="text/html; charset=utf-8" />
17         <script>
18             /** 定义一个JavaScript函数，当单击验证码时被调用，将重新请求并获取一个新的图片 */
19             function newgdcode(obj,url) {
20                 /** 后面传递一个随机参数，否则在IE7和火狐下，不刷新图片 */
21                 obj.src = url+ '?nowtime=' + new Date().getTime();
22             }
23         </script>
24     </head>
25     <body>
26         <!-- 在HTML中将PHP中动态生成的图片通过IMG标记输出，并添加了单击事件 -->
27         
29         <form method="POST" action="image.php">
30             <input type="text" size="4" name="code" />
31             <input type="submit" name="submit" value="提交">
32         </form>
33 </body>
34 </html>
```

16.2.4 实例演示

打开浏览器访问 image.php 脚本，就可以运行本例。图 16-5 为本例的演示结果，分别使用一次正确输入和一次错误的输入进行演示。



图 16-5 验证码实例演示

PHP 图片处理

像验证码或根据动态数据生成统计图表，以及前面介绍的一些 GD 库操作等都属于动态绘制图像。而在 Web 开发中，也会经常去处理服务中已存在的图片。例如，根据一些需求对图片进行缩放、加水印、裁剪、翻转和旋转等改图的操作。在 Web 应用中，经常使用的图片格式有 GIF、JPEG 和 PNG 中的一种或几种，当然 GD 库也可以处理其他格式的图片，但都很少用到。所以安装 GD 库时，至少安装 GIF、JPEG 或 PNG 三种格式中的一种，本书的图片处理也仅针对这三种图片格式进行介绍。

16.3.1 图片背景管理

在前面介绍的画布管理中，使用 `imagecreate()` 和 `imageCreateTrueColor()` 两个函数去创建画布资源。但如果需要对已有的图片进行处理，只要将这个图片作为画布资源即可，也就是我们所说的创建图片背景。可以通过下面介绍的几个函数，打开服务器或网络文件中已经存在的 GIF、JPEG 和 PNG 图像，返回一个图像标识符，代表了从给定的文件名取得的图像作为操作的背景资源。它们的原型如下所示，它们在失败时都会返回一个空字符串，并且输出一条错误信息。

resource imagecreatefromjpeg (string \$filename)	//从 JPEG 文件或 URL 新建一图像
resource imagecreatefrompng (string \$filename)	//从 PNG 文件或 URL 新建一图像
resource imagecreatefromgif (string \$filename)	//从 GIF 文件或 URL 新建一图像

不管使用哪个函数创建的图像资源，用完以后都需要使用 `imagedestroy()` 函数进行销毁。再有就是图片格式对应的问题，任何一种方式打开的图片资源都可以保存为同一种格式。例如，对于使用 `imagecreatefromjpeg()` 函数创建的图片资源，可以使用 `imagepng()` 函数以 PNG 格式将图像输出到浏览器或文件。当然最好是打开的是哪种格式的图片，就保存成对应的图片格式。如果要做到这一点，我们还需要先认识一下 `getimagesize()` 函数，通过图片名称就可以获取图片的类型、宽度和高度等。该函数的原型如下所示：

array getimagesize (string filename [, array &imageinfo])	//取得图片的大小和类型
--	--------------

如果不能访问 `filename` 指定的图像或者其不是有效的图像，该函数将返回 `FALSE` 并产生一条 `E_WARNING` 级的错误。如果不出错，`getimagesize()` 返回一个具有四个单元的数组，索引 0 包含图像宽度的像素值，索引 1 包含图像高度的像素值，索引 2 是图像类型的标记：1 = GIF，2 = JPG，3 = PNG，4 = SWF 等，索引 3 是文本字符串，内容为 “height=“yyy” width=“xxx””，可直接用于 `` 标记。如下所示：

```
1 <?php
2     list($width, $height, $type, $attr) = getimagesize("image/brophp.jpg");
3
4     echo ''
```

下面的例子声明一个 `image()` 函数，可以打开 GIF、JPG 和 PNG 中任意一种格式的图片，并在图片的中间加上一个字符串后，保存成原来格式（文字水印）。在以后的开发中，如果需要同样的操作（打开的是哪种格式的图片，也保存成对应格式的文件），可以参与本例的模式，代码如下所示：



```
1 <?php
2 /**
3  * 向不同格式的图片中间画一个字符串（也是文字水印）
4  * @param string $filename 图片的名称字符串，如果不是当前目录下的图片，请指明路径
5  * @param string $string 水印文字字符串，如果使用中文请使用utf-8字符串
6  */
7 function image($filename, $string) {
8     /* 获取图片的属性， 第一个宽度， 第二个高度， 类型1=>gif, 2=>jpeg, 3=>png */
9     list($width, $height, $type) = getimagesize($filename);
10    /* 可以处理的图片类型 */
11    $types = array(1=>"gif", 2=>"jpeg", 3=>"png");
12    /* 通过图片类型去组合，可以创建对应图片格式的，创建图片资源的GD库函数 */
13    $createfrom = "imagecreatefrom".$types[$type];
14    /* 通过“变量函数”去打对应的函数去创建图片的资源 */
15    $image = $createfrom($filename);
16    /* 设置居中字体的x轴作标位置 */
17    $x = ($width - imagefontwidth(5)*strlen($string)) / 2;
18    /* 设置居中字体的y轴作标位置 */
19    $y = ($height - imagefontheight(5)) / 2;
20    /* 设置字体的颜色为红色 */
21    $textcolor = imagecolorallocate($image, 255, 0, 0);
22    /* 向图片上画一个指定的字符串 */
23    imagestring($image, 5, $x, $y, $string, $textcolor);
24    /* 通过图片类型去组合保存对应格式的图片函数 */
25    $output = "image".$types[$type];
26    /* 通过变量函数去保存对应格式的图片 */
27    $output($image, $filename);
28    /* 销毁图像资源 */
29    imagedestroy($image);
30 }
31
32 image("brophp.gif", "GIF"); //向brophp.gif格式为gif的图片中央画一个字符串GIF
33 image("brophp.jpg", "JPEG"); //向brophp.jpg格式为jpeg的图片中央画一个字符串JPEG
34 image("brophp.png", "PNG"); //向brophp.png格式为png的图片中央画一个字符串PNG
```

演示结果如图 16-6 所示。



图 16-6 演示打开和保存对应格式的图片

16.3.2 图片缩放

网站优化不能只盯在代码上，内容也是网站最需要优化的对象之一，而图像又是网站中最主要的内容。图像的优化最需要处理的就是将所有上传到网站中的大图片自动缩放成小图（在网页中大小够用就行），以减少 N 倍的存储空间，并提高下载浏览的速度。所以图片缩放已经成为一个动态网站必须要处理的任务，经常和文件上传绑定在一起工作，能在上传图片的同时就调整其大小。当然有时也需要单独

处理图片缩放,例如在做图片列表时,如果直接用大图而在显示时才将其缩放成小图,这样做不仅下载速度会很慢,也会降低页面响应时间。通常遇到这样的应用都是在上传图片时,再为图片缩放出一个专门用来做列表的小图标,当单击这个小图标时,才会去下载大图浏览。

使用 GD 库处理图片缩放,通常使用 `imagecopyresized()`和 `imagecopyresampled()`两个函数中的一个,而使用 `imagecopyresampled()`函数处理后质量会更好一些。这里只介绍一下 `imagecopyresampled()`函数的使用方法。该函数的原型如下所示:

```
bool imagecopyresampled ( resource dst_image, resource src_image, int dst_x, int dst_y, int src_x, int src_y, int dst_w, int dst_h, int src_w, int src_h )
```

该函数将一幅图像中的一块正方形区域复制到另一个图像中,平滑地插入像素值,因此,减小了图像的大小而仍然保持了极高的清晰度。如果成功,则返回 TRUE,失败则返回 FALSE。参数 `dst_image` 和 `src_image` 分别是目标图像和源图像的标识符。如果源和目标的宽度和高度不同,则会进行相应的图像收缩和拉伸,坐标指的是左上角。本函数可用来在同一幅图内部复制(如果 `dst_image` 和 `src_image` 相同的话)区域,但如果区域交迭,则结果不可预知。在下面的示例中,以 JPEG 图片格式为例,编写一个图像缩放的函数 `thumb()`,代码如下所示:

```
1 <?php
2 /**
3  * 用于对图片进行缩放
4  * @param string $filename 图片的URL
5  * @width int $width 设置图片缩放的最大宽度
6  * @height int $height 设置图片缩放的最大高度
7  */
8 function thumb($filename, $width=200, $height=200) {
9     /* 获取原图像$filename的宽度$width_orig和高度$height_orig */
10    list($width_orig, $height_orig) = getimagesize($filename);
11
12    /* 根据参数$width和$height值,换算出等比例缩放的高度和宽度 */
13    if ($width && ($width_orig < $height_orig)) {
14        $width = ($height / $height_orig) * $width_orig;
15    } else {
16        $height = ($width / $width_orig) * $height_orig;
17    }
18
19    /* 将原图缩放到这个新创建的图片资源中 */
20    $image_p = imagecreatetruecolor($width, $height);
21    /* 获取原图的图像资源 */
22    $image = imagecreatefromjpeg($filename);
23
24    /*使用imagecopyresampled()函数进行缩放设置 */
25    imagecopyresampled($image_p, $image, 0, 0, 0, 0, $width, $height, $width_orig, $height_orig);
26
27    /* 将缩放后的图片$image_p保存, 100 (最佳质量, 文件最大) */
28    imagejpeg($image_p, $filename, 100);
29
30    imagedestroy($image_p); //销毁图片资源$image_p
31    imagedestroy($image); //销毁图片资源$image
32 }
33
34 thumb("brophp.jpg", 100,100); //将brophp.jpg图片缩放成100x100的小图
35 /* thumb("brophp.jpg", 200,2000); //如果按一边进行等比例缩放, 只需要将另一边给个无限大的值 */
```

在上例声明的 `thumb()`函数中,第一个参数 `$filename` 是要处理缩放图片的名称,也可以是图片位置



的 URL，第二个参数\$width 和第三个参数\$height，分别指定图片缩放的目标宽度和高度。本例使用了等比例缩放的算法，如果只需要通过宽度来约束图片的缩放，则高度设置一个无限大的值即可，反之亦然。上例将图片 brophp.jpg 缩放成宽度不超过 100 像素，高度也不能超过 100 像素的图片。演示结果如图 16-7 所示。

原图 brophp.jpg (300 × 300)

缩放后图 brophp.jpg (100 × 100)



图 16-7 缩放图片演示结果

16.3.3 图片裁剪

图片裁剪是指在一个大的背景图片中剪切出一张指定区域的图片，常见的应用是在用户设置个人头像时，可以从上传的图片中，裁剪出一个合适的区域作为自己的个人头像图片。图片裁剪和图片缩放的原理相似，所以也是借助 imagecopyresampled()函数去实现这个功能。同样也是以 JPEG 图片格式为例，声明一个图像裁剪函数 cut()，代码如下所示：


```

1 <?php
2 /**
3  在一个大的背景图片中剪裁出指定区域的图片，以jpeg图片格式为例
4  @param string $filename 需要剪裁的背景图片
5  @param int $x 剪裁图片左边开始的位置
6  @param int $y 剪裁图片顶部开始的位置
7  @param int $width 图片剪裁的宽度
8  @param int $height 图片剪裁的高度
9  */
10 function cut($filename, $x, $y, $width, $height){
11     /* 创建背景图片的资源 */
12     $back = imagecreatefromjpeg($filename);
13     /* 创建一个可以保存裁剪后图片的资源 */
14     $cutimg = imagecreatetruecolor($width, $height);
15
16     /* 使用imagecopyresampled()函数对图片进行裁剪 */
17     imagecopyresampled($cutimg, $back, 0, 0, $x, $y, $width, $height, $width, $height);
18
19     /* 保存裁剪后的图片，如果不想覆盖原图片，可以为裁剪后的图片加上前缀 */
20     imagejpeg($cutimg, $filename);
21
22     imagedestroy($cutimg); //销毁图像资源$cutimg
23     imagedestroy($back); //销毁图像资源$back
24 }
25
26 /* 调用cut()函数去裁剪brophp.jpg图片，从50, 50开始裁出宽度和高度都为200像素的图片 */
27 cut("brophp.jpg", 50, 50, 200, 200);

```

在上列声明的图片裁剪函数 cut()中，可以从第一个参数\$filename 传入的图片上，左部以第二个参数\$x 和顶部以第三个参数\$y 位置开始，裁剪出大小通过第四个参数\$width 指定的宽度和第五个参数\$height 指定的高度图片。上例在图片 brophp.jpg 中，左部和顶部都是从 50 像素位置开始，裁剪出宽度和高度都是 200 像素的图片。演示结果如图 16-8 所示。

原图 brophp.jpg

裁剪后图 brophp.jpg



图 16-8 裁剪图片演示结果

16.3.4 添加图片水印

为图片添加水印也是图像处理中常见的功能。因为只要在页面中见到的图片都可以很轻松地拿到，



你辛辛苦苦编辑的图片不想被别人不费吹灰之力拿走就用，所以为图片添加水印以确定版权，防止图片被盗用。制作水印可以使用文字（公司名称加网址），也可以使用图片（公司 LOGO），图片水印效果会更好一些，因为可以通过一些做图软件进行美化。

使用文字做水印，只需要在图片上画上一些文字即可。如果制作图片水印，就需要先了解一下 GD 库中的 `imagecopy()` 函数，能复制图像的一部分。该函数的原型如下所示：

```
bool imagecopy ( resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h )
```

该函数的作用是将 `src_im` 图像中坐标从 `src_x`, `src_y` 开始，宽度为 `src_w`，高度为 `src_h` 的一部分复制到 `dst_im` 图像中坐标为 `dst_x` 和 `dst_y` 的位置上。以 JPEG 格式的图片为例，编写一个为图片添加水印的函数 `watermark()`，代码如下所示：

```
1 <?php
2 /**
3  为背景图片添加图片水印（位置随机），背景图片格式为jpeg，水印图片格式为gif
4  @param string $filename 需要添加水印的背景图片
5  @param string $water 水印图片
6  */
7 function watermark($filename, $water){
8     /* 获取背景图片的宽度和高度 */
9     list($b_w, $b_h) = getimagesize($filename);
10
11     /* 获取水印图片的宽度和高度 */
12     list($w_w, $w_h) = getimagesize($water);
13
14     /* 在背景图片中放水印图片的随机起始位置 */
15     $posX = rand(0, ($b_w - $w_w));
16     $posY = rand(0, ($b_h - $w_h));
17
18     $back = imagecreatefromjpeg($filename); //创建背景图片的资源
19     $water = imagecreatefromgif($water); //创建水印图片的资源
20
21     /* 使用imagecopy()函数将水印图片复制到背景图片指定的位置中 */
22     imagecopy($back, $water, $posX, $posY, 0, 0, $w_w, $w_h);
23
24     /* 保存带有水印图片的背景图片 */
25     imagejpeg($back, $filename);
26
27     imagedestroy($back); //销毁背景图片资源$back
28     imagedestroy($water); //销毁水印图片资源$water
29 }
30
31 /* 调用watermark()函数，为背景JPEG格式的图片brophp.jpg，添加GIF格式的水印图片logo.gif */
32 watermark("brophp.jpg", "logo.gif");
```

上例声明的 `watermark()` 函数，第一个参数 `$filename` 为背景图片的 URL，第二个参数 `$water` 为水印图片的 URL。上例调用 `watermark()` 函数，将水印图片 `logo.gif` 添加到背景图片 `brophp.jpg` 中，位置在背景图片中随机。演示结果如图 16-9 所示。

原图 brophp.jpg

水印添加后图 brophp.jpg



图 16-9 为图片添加水印的演示结果

16.3.5 图片旋转和翻转

图片的旋转和翻转也是 Web 项目中比较常见的功能，但这是两个不同的概念，图片的旋转是指按特定的角度来转动图片，而图片的翻转则是将图片的内容按特定的方向对调。图片翻转需要自己编写函数来实现，而旋转图片则可以直接借助 GD 库中提供的 `imagerotate()` 函数完成。该函数的原型如下所示：

```
resource imagerotate ( resource src_im, float angle, int bgd_color [, int ignore_transparent] )
```

该函数可以将 `src_im` 图像用给定的 `angle` 角度旋转，`bgd_color` 指定了旋转后没有覆盖到的部分的颜色。旋转的中心是图像的中心，旋转后的图像会按比例缩小以适合目标图像的大小（边缘不会被剪去）。如果 `ignore_transparent` 被设为非零值，则透明色会被忽略（否则会被保留）。下面以 JPEG 格式的图片为例，声明一个可以旋转图片的函数 `rotate()`，代码如下所示：

```
1 <?php
2 /**
3  * 用给定角度旋转图像，以jpeg图处格式为例
4  * @param string $filename 要旋转的图片名称
5  * @param int $degrees 指定旋转的角度
6  */
7 function rotate($filename, $degrees) {
8     /* 创建图像资源，以jpeg格式为例 */
9     $source = imagecreatefromjpeg($filename);
10    /* 使用imagerotate()函数按指定的角度旋转 */
11    $rotate = imagerotate($source, $degrees, 0);
12    /* 将旋转后的图片保存 */
13    imagejpeg($rotate, $filename);
14 }
15
16 /* 将把一幅图像brophp.jpg旋转 180 度,即上下颠倒 */
17 rotate("brophp.jpg", 180);
```

上例声明的 `rotate()` 函数需要 2 个参数，第一个参数 `$filename` 指定一个图片的 URL，第二个参数 `$degrees` 则是指定图片旋转的角度。上例调用 `rotate()` 函数，将图片 `brophp.jpg` 旋转 180° ，即图片上下颠倒。

图片的翻转并不能随意指定角度，只能设置两个方向：沿 Y 轴水平翻转或沿 X 轴垂直翻转。如果是沿 Y 轴翻转，就是将原图从右向左（或从左向右）按一个像素宽度，及图片自身的高度循环复制到



新资源中，保存的新资源就是沿 Y 轴翻转后的图片。以 JPEG 格式图片为例，声明一个可以沿 Y 轴翻转的图片函数 turn_y(), 代码如下所示：

```
1 <?php
2 /**
3  * 图片沿y轴翻转，以jpeg格式为例
4  * @param string $filename 图片名称
5  */
6 function trun_y($filename){
7     /* 创建图片背景资源，以jpeg格式为例 */
8     $back = imagecreatefromjpeg($filename);
9
10    $width = imagesx($back); //获取图片的宽度
11    $height = imagesy($back); //获取图片的高度
12
13    /* 创建一个新的图片资源，用来保存沿y轴翻转后的图片 */
14    $new = imagecreatetruecolor($width, $height);
15    /* 沿y轴翻转就是将原图从右向左按一个像素宽度向新资源中逐个复制 */
16    for($x=0; $x < $width; $x++){
17        /* 逐条复制图片本身高度，1个像素宽度的图片到新资源中 */
18        imagecopy($new, $back, $width-$x-1, 0, $x, 0, 1, $height);
19    }
20
21    /* 保存翻转后的图片资源 */
22    imagejpeg($new, $filename);
23
24    imagedestroy($back); //销毁原背景图像资源
25    imagedestroy($new); //销毁新的图片资源
26 }
27
28 /* 图片沿y轴翻转*/
29 trun_y("brophp.jpg");
```

本例声明的 turn_y() 函数只需要一个参数，就是要处理的图片 URL。本例调用 turn_y() 函数将 brophp.jpg 图片沿 Y 轴进行翻转。如果是沿 X 轴翻转，就是将原图从上向下（或下左向上）按一个像素高度，以及图片自身的宽度循环复制到新资源中，保存的新资源就是沿 X 轴翻转后的图片。也是以 JPEG 格式图片为例，声明一个可以沿 X 轴翻转的图片函数 turn_x(), 代码如下所示：

```
1 <?php
2 /**
3  * 图片沿x轴翻转，以jpeg格式为例
4  * @param string $filename 图片名称
5  */
6 function trun_x($filename){
7     /* 创建图片背景资源，以jpeg格式为例 */
8     $back = imagecreatefromjpeg($filename);
9
10    $width = imagesx($back); //获取图片的宽度
11    $height = imagesy($back); //获取图片的高度
12
13    /* 创建一个新的图片资源，用来保存沿x轴翻转后的图片 */
14    $new = imagecreatetruecolor($width, $height);
15
16    /* 沿x轴翻转就是将原图从上向下按一个像素高度向新资源中逐个复制 */
17    for($y=0; $y < $height; $y++){
18        /* 逐条复制图片本身宽度，1个像素高度的图片到新资源中 */
```

```

19     imagecopy($new, $back, 0, $height-$y-1, 0, $y, $width, 1);
20 }
21
22 /* 保存翻转后的图片资源 */
23 imagejpeg($new, $filename);
24
25 imagedestroy($back); //销毁原背景图像资源
26 imagedestroy($new); //销毁新的图片资源
27 }
28
29 /* 将图片brophp.jpg沿x轴翻转 */
30 turn_x("brophp.jpg");

```

本例声明的 `turn_x()` 函数和 `turn_y()` 函数用法很相似，也只需要一个参数，就是要处理的图片 URL。本例调用 `turn_x()` 函数将 `brophp.jpg` 图片沿 *X* 轴进行翻转。这几个例子的演示结果如图 16-10 所示。



图 16-10 图片的旋转和翻转运行结果演示

16.4

设计经典的图像处理类

图像处理是网站中比较常见的功能，虽然前面也介绍了一些图片处理的方法，但都是以 JPEG 一种格式为例，并且都是处理当前目录下的图片，也没有考虑图片处理前后的区分对待。又因为 GD 库的应用还是比较烦琐的，为了能简化每次开发中处理图片的难度，也为了节省开发时间，也能让一些对 GD 库应用不熟练的开发人员，可以轻松操作图像，我们将项目中常的图像处理功能封装到一个类中。本例就是要完成一个图像处理类，帮助开发者在以后的开发中，通过编写几条简单代码的调用就可以完成对图像的处理。和上一章介绍的文件上传类一样，对于基础薄弱的读者只要会使用本类即可，而对一些喜欢挑战的朋友，可以尝试去读懂它，并能开发一个属于自己的图像处理类。

16.4.1 需求分析

要求自定义的图像处理类，即在使用非常简便的前提下，又可以完成以下几项功能：

- (1) 支持图片等比例缩放
- (2) 支持加图片水印
- (3) 支持图片裁剪

说明：要求可以设置图片存储的路径，并支持对 GIF、JPEG 和 PNG 三种格式的图片处理，处理后

的图片都可以通过指定前缀与原图进行区分。

16.4.2 程序设计

根据程序的需求，我们可以为图像处理类，声明一个构造方法和三个可见的成员方法。构造方法用来指定图片存放的路径，三个可见的成员方法分别用来对图片进行缩放、加图片水印和裁剪，其他方法都是为这几个方法提供服务的私有方法。该类的成员方法设计如表 16-1 所示。

表 16-1 图像处理类中设计的 4 个可见的成员方法

成员方法	描 述
__construct()	构造方法，用来在实例化对象时，设置图片存放的路径，该方法 只有一个可选参数 ，即图片存放的位置，如果不传值，则默认为当前目录
thumb()	该方法可以按等比例对图像进行缩放的方法，需要 4 个参数，返回缩放后的图片名称，参数分别介绍如下： 第一个参数 ：指定缩放图片名称（会到构造方法设置的路径中查找图片），支持 GIF、JPEG 和 PNG 三种格式图片 第二个参数 ：图片缩放的目标宽度 第三个参数 ：图片缩放的目标高度 第四个参数 ：可选的，指定缩放的图片名称前缀，默认为“th_”，如果覆盖原图片，将该参数值设置为空字符串即可
watermark()	该方法可以为一张背景图片按指定的位置添加图片水印，也需要 4 个参数，返回添加水印后图片的名称，参数分别使用介绍如下： 第一个参数 ：指定添加水印的背景图片名称（会到构造方法设置的路径中查找该图片），支持 GIF、JPEG 和 PNG 三种格式图片 第二个参数 ：指定水印图片名称，如果传入的水印图片没有指定路径，则默认去查找和背景图片相同的路径，否则到指定的路径下去找水印图片。也支持 GIF、JPEG 和 PNG 三种格式图片 第三个参数 ：指定水印图片在背景中添加的位置，有 10 种状态，使用一整数参数，0 为随机位置，其他位置如下： 1 为顶端居左，2 为顶端居中，3 为顶端居右； 4 为中部居左，5 为中部居中，6 为中部居右； 7 为底端居左，8 为底端居中，9 为底端居右； 第四个参数 ：可选的，指定添加水印后的图片名称前缀，默认为“wa_”，如果覆盖原图片，将该参数值设置为空字符串即可
cut()	该方法可以在一张背景图片中裁剪出一块指定区域的图片，共需要 6 个参数，返回裁剪出来的图片名称，指定的裁剪区域如果超出背景图片的大小范围则会裁减失败。参数分别介绍如下： 第一个参数 ：指定一张被裁剪的背景图片名称（会到构造方法设置的路径中查找该图片），同样支持 GIF、JPEG 和 PNG 三种格式图片 第二个参数 ：裁剪图片时指定相对于背景图片左部开始的位置 第三个参数 ：裁剪图片时指定相对于背景图片顶部开始的位置 第四个参数 ：指定裁剪图片的宽度 第五个参数 ：指定裁剪图片的高度 第六个参数 ：可选的，指定裁剪出来的图片名称前缀，默认为“cu_”，如果覆盖原图片，参数值设置为空字符串

16.4.3 图像处理类代码实现

本类可以完成对图片的缩放、加水印和裁剪的功能，前面也介绍过些类功能的实现方法。但在本类中会将其功能更加完善，例如支持多种图片类型的处理，缩放时进行优化等。除了在上一节中提供的可以操作的 4 个成员方法以外，编写一个图像类当然还需要更多的成员，但其他的成员方法只需要内部使用，并不需要用户在对象外部操作，所以只要声明为 **private**（私有）封装在对象内部即可。编写图像处理类 **Image** 并声明在 **image.class.php** 文件中，代码如下所示：

```

1 <?php
2 /**
3  * file: image.class.php 类名为Image
4  * 图像处理类，可以完成对各种类型的图像进行缩放、加图片水印和剪裁的操作。
5  */
6 class Image {
7     /* 图片保存的路径 */
8     private $path;
9
10    /**
11     * 实例图像对象时传递图像的一个路径，默认值是当前目录
12     * @param string $path 可以指定处理图片的路径
13     */
14    function __construct($path="./"){
15        $this->path = rtrim($path, "/")."/";
16    }
17
18    /**
19     * 对指定的图像进行缩放
20     * @param string $name 是需要处理的图片名称
21     * @param int $width 缩放后的宽度
22     * @param int $height 缩放后的高度
23     * @param string $sqz 是新图片的前缀
24     * @return mixed 是缩放后的图片名称,失败返回false;
25     */
26    function thumb($name, $width, $height, $sqz="th_"){
27        /* 获取图片宽度、高度、及类型信息 */
28        $imgInfo = $this->getInfo($name);
29        /* 获取背景图片的资源 */
30        $srcImg = $this->getImg($name, $imgInfo);

```



```
31      /* 获取新图片尺寸 */
32      $size = $this->getNewSize($name,$width, $height,$imgInfo);
33      /* 获取新的图片资源 */
34      $newImg = $this->kidOfImage($srcImg, $size,$imgInfo);
35      /* 通过本类的私有方法，保存缩略图并返回新缩略图的名称，以"th_"为前缀 */
36      return $this->createNewImage($newImg, $qz.$name,$imgInfo);
37  }
38
39  /**
40  * 为图片添加水印
41  * @param string $groundName 背景图片，即需要加水印的图片，暂只支持GIF,JPG,PNG格式
42  * @param string $waterName 图片水印，即作为水印的图片，暂只支持GIF,JPG,PNG格式
43  * @param int $waterPos 水印位置，有10种状态，0为随机位置：
44  *      1为顶端居左，2为顶端居中，3为顶端居右；
45  *      4为中部居左，5为中部居中，6为中部居右；
46  *      7为底端居左，8为底端居中，9为底端居右；
47  * @param string $qz 加水印后的图片的文件名在原文件名前面加上这个前缀
48  * @return mixed 是生成水印后的图片名称，失败返回false
49  */
50  function waterMark($groundName, $waterName, $waterPos=0, $qz="wa_"){
51      /*获取水印图片是当前路径，还是指定了路径*/
52      $curpath = rtrim($this->path,"/")."/";
53      $dir = dirname($waterName);
54      if($dir == "."){
55          $wpath = $curpath;
56      }else{
57          $wpath = $dir."/";
58          $waterName = basename($waterName);
59      }
60
61      /*水印图片和背景图片必须都要存在*/
62      if(file_exists($curpath.$groundName) && file_exists($wpath.$waterName)){
63          $groundInfo = $this->getInfo($groundName); //获取背景信息
64          $waterInfo = $this->getInfo($waterName, $dir); //获取水印图片信息
65          /*如果背景比水印图片还小，就会被水印全部盖住*/
66          if(!$pos = $this->position($groundInfo, $waterInfo, $waterPos)){
67              echo '水印不应该比背景图片小！';
68              return false;
69          }
70
71          $groundImg = $this->getImg($groundName, $groundInfo); //获取背景图像资源
72          $waterImg = $this->getImg($waterName, $waterInfo, $dir); //获取水印图片资源
73
74          /* 调用私有方法将水印图像按指定位置复制到背景图片中 */
75          $groundImg = $this->copyImage($groundImg, $waterImg, $pos, $waterInfo);
76          /* 通过本类的私有方法，保存加水图片并返回新图片的名称，默认以"wa_"为前缀 */
77          return $this->createNewImage($groundImg, $qz.$groundName, $groundInfo);
78      }else{
79          echo '图片或水印图片不存在！';
80          return false;
81      }
82  }
83
84
85  /**
86  * 在一个大的背景图片中剪裁出指定区域的图片
87  * @param string $name 需要剪裁的背景图片
88  * @param int $x 剪裁图片左边开始的位置
89  * @param int $y 剪裁图片顶部开始的位置
90  * @param int $width 图片剪裁的宽度
```

```

91  * @param int $height      图片剪裁的高度
92  * @param string $qz        新图片的名称前缀
93  * @return mixed           裁剪后的图片名称,失败返回false;
94  */
95  function cut($name, $x, $y, $width, $height, $qz="cu_"){
96      $imgInfo=$this->getInfo($name);          //获取图片信息
97      /* 裁剪的位置不能超出背景图片范围 */
98      if( (($x+$width) > $imgInfo['width']) || (($y+$height) > $imgInfo['height'])) {
99          echo "裁剪的位置超出了背景图片范围!";
100         return false;
101     }
102
103     $back = $this->getImg($name, $imgInfo);      //获取图片资源
104     /* 创建一个可以保存裁剪后图片的资源 */
105     $cutimg = imagecreatetruecolor($width, $height);
106     /* 使用imagecopyresampled()函数对图片进行裁剪 */
107     imagecopyresampled($cutimg, $back, 0, 0, $x, $y, $width, $height, $width, $height);
108     imagedestroy($back);
109     /* 通过本类的私有方法,保存剪切图并返回新图片的名称,默认以"cu_"为前缀 */
110     return $this->createNewImage($cutimg, $qz.$name,$imgInfo);
111 }
112
113 /* 内部使用的私有方法,用来确定水印图片的位置 */
114 private function position($groundInfo, $waterInfo, $waterPos){
115     /* 需要加水印的图片的长度或宽度比水印还小,无法生成水印 */
116     if( ($groundInfo["width"]<$waterInfo["width"]) ||
117         ($groundInfo["height"]<$waterInfo["height"]) ) {
118         return false;
119     }
120     switch($waterPos) {
121         case 1:          //1为顶端居左
122             $posX = 0;
123             $posY = 0;
124             break;
125         case 2:          //2为顶端居中
126             $posX = ($groundInfo["width"] - $waterInfo["width"]) / 2;
127             $posY = 0;
128             break;
129         case 3:          //3为顶端居右
130             $posX = $groundInfo["width"] - $waterInfo["width"];
131             $posY = 0;
132             break;
133         case 4:          //4为中部居左
134             $posX = 0;
135             $posY = ($groundInfo["height"] - $waterInfo["height"]) / 2;
136             break;
137         case 5:          //5为中部居中
138             $posX = ($groundInfo["width"] - $waterInfo["width"]) / 2;
139             $posY = ($groundInfo["height"] - $waterInfo["height"]) / 2;
140             break;
141         case 6:          //6为中部居右
142             $posX = $groundInfo["width"] - $waterInfo["width"];
143             $posY = ($groundInfo["height"] - $waterInfo["height"]) / 2;
144             break;
145         case 7:          //7为底端居左
146             $posX = 0;
147             $posY = $groundInfo["height"] - $waterInfo["height"];
148             break;
149         case 8:          //8为底端居中
150             $posX = ($groundInfo["width"] - $waterInfo["width"]) / 2;

```



```
151         $posY = $groundInfo["height"] - $waterInfo["height"];
152         break;
153     case 9:          //9为底端居右
154         $posX = $groundInfo["width"] - $waterInfo["width"];
155         $posY = $groundInfo["height"] - $waterInfo["height"];
156         break;
157     case 0:
158     default:        //随机
159         $posX = rand(0, ($groundInfo["width"] - $waterInfo["width"]));
160         $posY = rand(0, ($groundInfo["height"] - $waterInfo["height"]));
161         break;
162     }
163     return array("posX"=>$posX, "posY"=>$posY);
164 }
165
166
167 /* 内部使用的私有方法，用于获取图片的属性信息（宽度、高度和类型） */
168 private function getInfo($name, $path=".") {
169     $spath = $path=="." ? rtrim($this->path, "/"). "/" : $path.'/' ;
170
171     $data = getimagesize($spath.$name);
172     $imgInfo["width"] = $data[0];
173     $imgInfo["height"] = $data[1];
174     $imgInfo["type"] = $data[2];
175
176     return $imgInfo;
177 }
178
179 /*内部使用的私有方法， 用于创建支持各种图片格式（jpg,gif,png三种）资源 */
180 private function getImg($name, $imgInfo, $path='.'){
181
182     $spath = $path=="." ? rtrim($this->path, "/"). "/" : $path.'/' ;
183     $srcPic = $spath.$name;
184
185     switch ($imgInfo["type"]) {
186     case 1:          //gif
187         $img = imagecreatefromgif($srcPic);
188         break;
189     case 2:          //jpg
190         $img = imagecreatefromjpeg($srcPic);
191         break;
192     case 3:          //png
193         $img = imagecreatefrompng($srcPic);
194         break;
195     default:
196         return false;
197         break;
198     }
199     return $img;
200 }
201
202 /* 内部使用的私有方法，返回等比例缩放图片宽度和高度，如果原图比缩放后的还小保持不变 */
203 private function getNewSize($name, $width, $height, $imgInfo){
204     $size["width"] = $imgInfo["width"]; //原图片的宽度
205     $size["height"] = $imgInfo["height"]; //原图片的高度
206
207     if($width < $imgInfo["width"]){
208         $size["width"]=$width; //缩放的宽度如果比原图小才重新设置宽度
209     }
210 }
```

```

211     if($height < $imgInfo["height"]){
212         $size["height"] = $height;           //缩放的高度如果比原图小才重新设置高度
213     }
214     /* 等比例缩放的算法 */
215     if($imgInfo["width"]*$size["width"] > $imgInfo["height"] * $size["height"]){
216         $size["height"] = round($imgInfo["height"]*$size["width"]/$imgInfo["width"]);
217     }else{
218         $size["width"] = round($imgInfo["width"]*$size["height"]/$imgInfo["height"]);
219     }
220
221     return $size;
222 }
223
224 /* 内部使用的私有方法, 用于保存图像, 并保留原有图片格式 */
225 private function createNewImage($newImg, $newName, $imgInfo){
226     $this->path = rtrim($this->path, "/")."/";
227     switch ($imgInfo["type"]) {
228         case 1:           //gif
229             $result = imageGIF($newImg, $this->path.$newName);
230             break;
231         case 2:           //jpg
232             $result = imageJPEG($newImg, $this->path.$newName);
233             break;
234         case 3:           //png
235             $result = imagePng($newImg, $this->path.$newName);
236             break;
237     }
238     imagedestroy($newImg);
239     return $newName;
240 }
241
242 /* 内部使用的私有方法, 用于加水印时复制图像 */
243 private function copyImage($groundImg, $waterImg, $pos, $waterInfo){
244     imagecopy($groundImg, $waterImg, $pos["posX"], $pos["posY"], 0, 0, $waterInfo["width"],
245         $waterInfo["height"]);
246     imagedestroy($waterImg);
247     return $groundImg;
248 }
249
250 /* 内部使用的私有方法, 处理带有透明度的图片保持原样 */
251 private function kidOfImage($srcImg, $size, $imgInfo){
252     $newImg = imagecreatetruecolor($size["width"], $size["height"]);
253     $otsc = imagecolortransparent($srcImg);
254     if( $otsc >= 0 && $otsc < imagecolorstotal($srcImg)) {
255         $transparentcolor = imagecolorsforindex( $srcImg, $otsc );
256         $newtransparentcolor = imagecolorallocate(
257             $newImg,
258             $transparentcolor['red'],
259             $transparentcolor['green'],
260             $transparentcolor['blue']
261         );
262         imagefill( $newImg, 0, 0, $newtransparentcolor );
263         imagecolortransparent( $newImg, $newtransparentcolor );
264     }
265     imagecopyresized( $newImg, $srcImg, 0, 0, 0, 0, $size["width"], $size["height"], $imgInfo
266         ["width"], $imgInfo["height"] );
267     imagedestroy($srcImg);
268     return $newImg;
269 }

```




16.4.4 图像处理类的应用过程

图像处理类提供的缩放、加图片水印及裁剪的功能，是三个互相不干扰的功能，所以在项目开发中这三个功能很少绑定在一起使用。这里独立介绍一下每个方法的详细应用。下例是使用图像处理类 Image 实现图片缩放的示例，首先必须先加载图像处理类 Image 所在的文件 image.class.php，然后实例化 Image 类的对象，再通过对象中的 thumb() 方法实现对图片的缩放。代码如下所示：

```
1 <?php
2  /* 加载图像处理类所在的文件 */
3  include "image.class.php";
4  /* 实例化图像处理类对象，通过构造方法的参数指定图片所在路径 */
5  $img = new Image('./image/');
6
7  /* 将上传到服务器的大图控制在500X500以内，最后一个参数使用了'',将原来图片覆盖 */
8  $filename = $img -> thumb("brophp.jpg", 500, 500, '');
9
10 /* 另存为一张250X250的中图，返回的图片名称会默认加上th_前缀 */
11 $midname = $img -> thumb($filename, 250, 250);
12 /* 另存为一张80X80的小图标，返回的图片名称前使用指定的icon_作为前缀 */
13 $icon = $img -> thumb($filename, 80, 80, 'icon_');
14
15 echo $filename.'<br>'; //缩放成功输出brophp.jpg
16 echo $midname.'<br>'; //缩放成功输出th_brophp.jpg
17 echo $icon.'<br>'; //缩放成功输出icon_brophp.jpg
```

在上例的第 8 行，将图片 brophp.jpg 缩放至宽度和高度都不超过 500 个像素。在最后一个设置前缀的参数中使用了空字符串作为值，这样原图片就被这个缩放后的图片给覆盖掉了。这种用法通常和文件上传一起使用，将用户上传的图片设置成自动这样处理，就可以优化用户上传图片的尺寸了。第 11 行和第 13 行，则分别使用默认的前缀“th_”和指定的前缀“icon_”，创建出两张缩放后的新图片。

下例演示了使用 Image 类添加图片水印的用法，和上面处理图片缩放的例子一样，都需要先加载类文件并实例化 Image 类的对象。下面的例子演示了使用对象中的 watermark() 方法添加图片水印的各种操作。代码如下所示：

```

1 <?php
2  /* 加载图像处理类所在的文件 */
3  include "image.class.php";
4  /* 实例化图像处理类对象，没有通过参数指定图片所在路径，所以默认为当前路径 */
5  $img = new Image();
6
7  /* 为图片brophp.jpg，添加一个image目录下的logo.gif图片水印，第三个参数使用1，水印位置顶部居左*/
8  echo $img -> watermark('brophp.jpg', './image/logo.gif', 1, 'wa1_'); //输出wa1_brophp.jpg
9  echo $img -> watermark('brophp.jpg', './image/logo.gif', 2, 'wa2_'); //输出wa2_brophp.jpg
10 echo $img -> watermark('brophp.jpg', './image/logo.gif', 3, 'wa3_'); //输出wa3_brophp.jpg
11 echo $img -> watermark('brophp.jpg', './image/logo.gif', 4, 'wa4_'); //输出wa4_brophp.jpg
12 echo $img -> watermark('brophp.jpg', './image/logo.gif', 5, 'wa5_'); //输出wa5_brophp.jpg
13 echo $img -> watermark('brophp.jpg', './image/logo.gif', 6, 'wa6_'); //输出wa6_brophp.jpg
14 echo $img -> watermark('brophp.jpg', './image/logo.gif', 7, 'wa7_'); //输出wa7_brophp.jpg
15 echo $img -> watermark('brophp.jpg', './image/logo.gif', 8, 'wa8_'); //输出wa8_brophp.jpg
16 echo $img -> watermark('brophp.jpg', './image/logo.gif', 9, 'wa9_'); //输出wa9_brophp.jpg
17
18 /* 没有指定第四个参数（名称前缀），使用默认的名称前缀"wa_" */
19 echo $img -> watermark('brophp.jpg', './image/logo.gif', 0); //输出wa_brophp.jpg
20 /* 第四个参数（名称前缀）设置空(''), 就会将原来brophp.jpg图片覆盖掉 */
21 echo $img -> watermark('brophp.jpg', './image/logo.gif', 0, ''); //输出brophp.jpg
22 /* 第二个参数如果没有指定路径，则logo.gif图片和brophp.jpg图片在同一个目录下 */
23 echo $img -> watermark('brophp.jpg', 'logo.gif', 0, 'wa0_'); //输出wa0_brophp.jpg

```

在本例中，实例化 Image 类的对象时，并没有通过构造方法设置图片保存的路径，所以处理的图片默认都在当前路径下。在通过第二个参数指定图片水印时，如果不带路径，则水印图片和背景图片在相同的目录下。下例同样使用 Image 类的实例对象，并通过对象中的 cut()方法对图片进行裁剪，示例代码如下所示：

```

1 <?php
2  /* 加载图像处理类所在的文件 */
3  include "image.class.php";
4  /* 实例化图像处理类对象，通过构造方法的参数指定图片所在路径 */
5  $img = new Image('./image/');
6
7  /* 在图片brophp.jpg中，从50x50开始，裁剪出120x120的图片，返回带默认前缀"cu_"的图片名称 */
8  $img -> cut("brophp.jpg", 50, 50, 120, 120); //cu_brophp.jpg
9  /* 可以通过第6个参数，为裁剪出来的图片指定名称前缀，实现同一张背景图片中裁剪出多张图片 */
10 $img -> cut("brophp.jpg", 50, 50, 120, 120, 'user_'); //user_brophp.jpg
11 /* 如果第6个参数设置为'', 则是使用裁剪出来的图片将原图覆盖掉 */
12 $img -> cut("brophp.jpg", 50, 50, 120, 120, ''); //brophp.jpg

```



无兄弟，不编程

16.5

小结

本章必须掌握的知识点

- 画布管理和设置颜色
- 绘制和生成图像
- 图片的一些常见的操作（缩放、加水印、裁剪、旋转和翻转）
- 验证码类 Vcode 的使用
- 图片处理类 Image 的使用

本章需要了解的内容

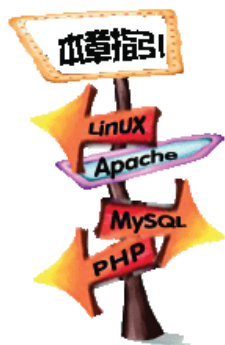
- 验证码类 Vcode 的编写
- 图片处理类 Image 的编写

本章需要拓展的内容

- 本章没有介绍到的其他 GD 库函数
- 找一些使用 GD 编写的插件去应用（例如，动态统计图）

第26章

超轻量级 PHP 框架 BroPHP



BroPHP 是一个免费开源的轻量级 PHP 框架（学习型），允许你把基于 BroPHP 框架开发的应用去开源或发布、销售商业产品。BroPHP 框架完全采用面向对象的设计思想，并且是基于 MVC 的三层设计模式，具有部署和应用及为简单、效率高、速度快，扩展性和可维护性都很好等特点，可以稳定地用于商业及门户的开发。BroPHP 框架包括单入口文件、MVC 模式、目录组织结构、类自动加载、强大基础类、URL 处理、输入处理、错误处理、缓存机制、扩展类等功能。是专门为《细说 PHP》的读者及 LAMP 兄弟连全体学员提供的“学习型 PHP 框架”。当然，任何 PHP 应用开发爱好者都可以从 BroPHP 框架的简单和快速的特性中受益。另外，BroPHP 框架的应用不仅使 WEB 开发变得更简单、更快捷，最主要的目的是让 PHP 学习者，通过使用本框架从而去了解 PHP 框架、再去研究框架，最后达到开发自己的框架的目的。

26.1

BroPHP 框架概述

BroPHP 是“学习型”的超轻量级框架（文件很小，对 CPU 和内存消耗极低），目前版本为 BroPHP 1.0。虽然第一版功能不算很多，但具备了一个框架构成最少应该有的全部功能（包括：MVC 模式、目录组织结构、类自动加载、基类、URL 处理、输入处理、错误处理、扩展类等）。本框架在已有的功能上，不管从组织结构上，还是从代码质量上，以及运行效率上都做到了单服务器最佳的效果。使用 BroPHP 框架适合开发 BBS、电子商城、SNS、CMS、Blog、企业门户等中小型系统。另外，本框架特别适合学习 PHP 使用，可以让你认识框架、分析框架内幕，从而达到编写自己框架的目的，并通过 BroPHP 框架改版，直接作为公司内部的开发框架使用。

26.1.1 系统特点

BroPHP 框架的编码结构尽量实现各模块功能独立，并将《细说 PHP》中各章节知识点整合在了一起。当你在分析框架源码时，PHP 的技术点可以参考本书前面的各个章节，也会将你了解的零散的 PHP



知识点组织在一起。BroPHP 框架部分特点如下。

- (1) 第一次访问时为用户自动创建了项目所需要的全部目录结构，用户无须再对组织项目的目录结构而烦恼。
- (2) 本框架采用模块和操作的方式来执行，简单易用，功能适中，更符合中国 Web 程序员的开发习惯。
- (3) 通过本框架编写的项目是完全采用 PHP 面向对象的思想，符合人类的思维模式，具有独立性、通用性、灵活性，有利于对项目的维护和调试。
- (4) 基于 MVC 的开发模式，将视图层和业务层分离，达到快速的部署，具有很好的可维护性，以及高重用性和可适用性，特别有利于软件工程化管理。
- (5) 内建丰富的 SQL 查询机制，操作灵活，简单易用。
- (6) 采用了目前业界最著名的 PHP 模板引擎 Smarty，对于熟悉 Smarty 的程序员而言具有很好的模板开发优势。
- (7) 使用 memcached 对 SQL 和 session 进行缓存，也可以使用 Smarty 缓存技术进行页面静态化，提升效率，减少运行消耗。
- (8) 本框架提供一些常用的扩展类，直接使用即可完成一些常见的功能。例如，文件上传、图像处理、分页实现及验证码类。
- (9) 本框架支持自定义扩展类库和扩展函数的使用，可以无限地实现功能扩展。
- (10) 采用人性化的调试模式，可以了解项目的运行过程，也可以快速解决项目开发时遇到的错误和异常。
- (11) 框架源码简单明了，结构清晰，方便在工作中根据当前项目的需求对框架进行改造。

可以在本书配套光盘中找到 BroPHP 框架源码，也可以到 <http://www.brophp.com> 或 <http://bbs.lampbrother.net> (LAMP 兄弟连) 网站中下载 BroPHP 框架最新版本和最新的帮助文档。

26.1.2 环境要求

操作系统：支持 Linux/Windows 服务器, 可以跨平台应用

WEB 服务器：可运行于 Apache、IIS 和 nginx 中

PHP 环境：PHP5.0 以上版本，需要安装 XML、mysqli 或 PDO、GD 库、MemCache 扩展模块

注意：对于 PHP 新手，推荐使用集成开发环境 AppServ 或 WAMP 对 BroPHP 进行本地开发和测试。

26.1.3 BroPHP 框架源码的目录结构

下例为 BroPHP 框架的系统目录，在项目开发时直接将 brophp 目录及子目录的所有文件复制到项目根目录中即可，并不需要对这个框架源文件做任何修改。但在 Linux 操作中需要注意，要将这个本框架目录及子目录的权限，设置运行 PHP 的用户有读的权限。

-- brophp 目录	#BroPHP 框架目录
-- bases 目录	#BroPHP 框架基础类存放目录
-- classes 目录	#BroPHP 框架扩展类存放目录

-- commons 目录	#BroPHP 框架通用函数和资源存放目录
-- libs 目录	#Smarty 模板引擎源文件存放目录
-- brophp.php 文件	#BroPHP 框架的公共入口文件



26.2 单一入口

在使用 PHP 过程化编程时，每个 PHP 文件都能独立访问并运行，就像一个体育场有多个入口一样，需要在每个入口都进行检票和安全检查。而采用单一入口模式进行项目部署和访问，无论完成什么功能，一个项目只有一个统一（但不一定是唯一）的入口，就像一个体育场如果只能从一个入口入场（程序是抽象的，一个入口和多个入口效率是一样的）控制起来则更灵活，几乎没有什么缺点。使用主入口文件部署项目的优点如下。

➤ 加载文件方便

在编写和阅读过程化程序代码时，经常会遇到文件之间互相包含，其中包括 PHP 使用 `include` 包括函数库和公共资源文件，也包括在 HTML 中使用 `<link>` 和 `<script>` 加载 CSS 和 JavaScript 文件。项目越大，文件越多，越让人感觉头疼，就像一张大网一样将文件交织在了一起，不容易找到头绪。而使用单一入口则解决这个难题，在项目应用中用到的任何一个文件，只要相对于单一入口文件的位置查找即可。

➤ 权限验证容易

如果每个 PHP 文件都可以独立访问，在做用户权限验证时就需要对每个文件进行判断。而采用单一入口，则只需要在一个位置进行判断即可。

➤ URL 重写简单

如果每个 PHP 文件，以及不同目录下的 PHP 文件都可以独立访问，则在 Web 服务器中对 URL 进行重新编写时，就需要编写很多条规则。而采用单一入口则在 URL 重写时只需要简单的几条规则即可。

26.2.1 基于 BroPHP 框架的单一入口编写规则

例如，在项目的根目录下，声明 `index.php` 文件作为当前项目应用的单一入口文件，和 BroPHP 框架库文件目录同级。编写的单一入口文件 `index.php` 的内容示例如下所示：

```
1 <?php
2 /**
3      file: index.php    声明基于BroPHP框架开发的项目，单一入口文件示例
4  */
5  define("BROPHP", "./brophp");      #定义BroPHP框架所在路径（相对于入口文件，不要加"/"）
6  define("APP", ".");                #定义项目的应用路径（"/"可加可不加）
7  define(BROPHP."./brophp.php");     #加载BroPHP框架目录下的入口文件
```

基于 BroPHP 框架项目的单一入口文件，可以自己定义名称。如 `index.php`、`admin.php`、`blog.php` 等之类命令都可以。但在入口文件中至少要编写两行代码：必须指定项目的应用路径，也就是在主入口文件中声明 APP 常量；再就是必须包含框目录下的入口文件 `brophp.php`。具体说明如下。

- 在上例的第 6 行中，是定义项目的应用路径，即自己写的项目应用放到哪个目录下就指定哪个目录，常量名（APP）是固定的不能改变。例如，`define("APP", ".")`，如果想将项目写在当前项目路径下的 `admin` 目录下则可以 `define("APP", "./admin/")`。另外，这个路径最后结尾的 “/” 可加可不加。
- 在上例的第 7 行中，是加载 BroPHP 框架库文件目录下的入口文件，是固定的写法。可以将上例

的第 5 行和第 7 行代码结合为一行，如 `require("../brophp/brophp.php")`。

26.3

部署项目应用目录

下例提供的是项目应用目录，只是默认的方式。项目的应用目录结构并不需要开发人员手动创建，只需要定义好项目的入口文件之后，系统会在第一次执行的时候自动生成项目必需的所有目录结构。访问上一节声明的单一入口文件，自动生成目录结构及说明如下所示：

-- brophp 目录	#BroPHP 框架库文件所在的目录
-- index.php 文件	#主入口文件（可以使用其他名称，也可以放在其他位置）
-- config.inc.php 文件	#项目的配置文件
-- controls 目录	#声明控制器类的目录
-- common.class.php 文件	#默认控制器的基类（用于写权限）
-- index.class.php 文件	#默认控制器（提供参考）
-- models 目录	#声明业务模型类的目录
-- views 目录	#声明视图的目录（Smarty 模板存放目录）
-- default 目录	#默认模板存入目录（可以为项目提供多套模板）
-- xxx 目录	#特定模块自己创建的目录（xxx 和模块同名）
-- xxx.tpl 文件	#为特定的操作自己定义的模板文件（xxx 和动作同名）
-- public 目录	#同一应用中公用模板存放目录
-- success.tpl 文件	#同一应用页面跳转提示模板
-- resource 目录	#当前项目模板的资源目录
-- css 目录	#当前项目模板的样式目录
-- js 目录	#当前项目模板的 JavaScript 目录
-- images 目录	#当前项目模板的图片目录
-- classes 目录	#用户自定义的扩展类目录
-- commons 目录	#用户自定义的扩展函数目录
-- functions.inc.php 文件	#用户将自定义的扩展函数都必须写在这个文件中
-- public 目录	#项目的所有应用公用的资源目录
-- css 目录	#项目的所有应用公用的 CSS 目录
-- js 目录	#项目的所有应用公用的 JavaScript 目录
-- images 目录	#项目的所有应用公用的图片
-- uploads 目录	#项目的所有应用公用的文件上传目录
-- runtime 目录	#项目运行时自动生成文件存放目录（可以随时删除）
-- comps 目录	#Smarty 模板编译文件存放目录
-- cache 目录	#Smarty 页面静态缓存目录
-- data 目录	#项目使用的数据库表结构缓存目录
-- controls 目录	#控制器缓存目录
-- models 目录	#业务模型缓存目录
-- _index.php 文件	#文件锁，如果目录结构存在则不再重新生成

上例只是以入口文件 `index.php` 为例，并且应用目录和框架目录在同一级时，默认生成的目录结构。具体的每个目录和文件的作用，在应用时可以参与后面部分的详细介绍。

注意：在 Linux 操作系统中，开发阶段需要让运行 PHP 用户有可写的权限，而当项目上线运行时，只需要 `runtime` 目录及子目录和 `public/uploads` 目录需要运行 PHP 用户有可写的权限，其他目录只要让运行 PHP 用户具有可读权限即可。



26.3.1 项目部署方式

在部署项目时，项目的目录结构往往由不同项目的应用 6 决定。使用 BroPHP 框架时，项目的应用目录（controls、models、views）和入口文件的位置，可以由不同项目的应用自己决定存放位置，而其他公用资源目录和配置文件（classes、commons、public、runtime、config.inc.php）必须同框架目录 brophp 在同一级。这里推荐几种部署应用目录结构的方法：

1. 如果项目只有一个应用（推荐两种方式）

第一种：入口文件和应用目录与框架在同级目录。这种方式比较简单，只需要在入口文件的第二行将应用目录常量 APP 的值改成“.”或“./”，然后直接访问入口文件即可生成所有目录结构。

入口文件名命名：index.php（可以改为其他名称）

```
1 <?php
2 /**
3      file: index.php  声明基于BroPHP框架开发的项目，单一入口文件示例
4 */
5 define("APP", ".");           #定义项目的应用路径('./'可加可不加)
6 define("./brophp/brophp.php"); #加载BroPHP框架目录下的入口文件brophp.php
```

自动生成的应用目录结构（都是同级的）

-- brophp 目录	#BroPHP 框架目录
-- index.php 文件	#主入口文件（可以使用其他名称，也可以放在其他位置）
-- config.inc.php 文件	#项目的配置文件
-- controls 目录	#声明控制器类的目录
-- models 目录	#声明业务模型类的目录
-- views 目录	#声明视图的目录（Smarty 模板存放目录）
-- classes 目录	#用户自定义的扩展类目录
-- commons 目录	#用户自定义的扩展函数目录
-- public 目录	#项目的所有应用公用的资源目录
-- runtime 目录	#项目运行时自动生成文件存放目录（可以随时删除）

第二种：项目的应用目录放到自己定义的目录下。例如，声明一个应用目录名为“home”，主入口文件和其他资源及框架同级。只需要在主入口文件的第二行将应用目录常量 APP 的值改成“./home”或“./home/”，然后直接访问主入口文件，即可生成所有目录结构。

入口文件名命名：index.php（可以改为其他名称）

```
1 <?php
2 /**
3      file: index.php  声明基于BroPHP框架开发的项目，单一入口文件示例
4 */
5 define("APP", "./home/");       #定义项目的应用路径在home下('./'可加可不加)
6 define("./brophp/brophp.php");  #加载BroPHP框架目录下的入口文件brophp.php
```

自动生成的应用目录结构（controls、models 和 views 在 home 目录下）

-- brophp 目录	#BroPHP 框架目录
-- index.php 文件	#主入口文件（可以使用其他名称，也可以放在其他位置）
-- config.inc.php 文件	#项目的配置文件
-- home 目录	#自定义的项目应用目录
-- controls 目录	#声明控制器类的目录

-- models 目录	#声明业务模型类的目录
-- views 目录	#声明视图的目录（Smarty 模板文件存放目录）
-- classes 目录	#用户自定义的扩展类目录
-- commons 目录	#用户自定义的扩展函数目录
-- public 目录	#项目的所有应用公用的资源目录
-- runtime 目录	#项目运行时自动生成文件存放目录（可以随时删除）

2. 如果项目分为前台和后台两个应用（推荐三种方式）

第一种：前台和后台的入口文件都和框架目录 brophp 在同一级目录中，后台的应用目录定义在 admin（可以改为其他名称）下。然后分别访问两个入口文件即可生成所有目录结构。

前台入口文件名命名：index.php（可以改为其他名称）

```

1 <?php
2 /**
3      file: index.php  声明基于BroPHP框架开发的项目，单一入口文件示例
4 */
5 define("APP", ".");           #定义项目的应用路径（'/'可加可不加）
6 define("./brophp/brophp.php"); #加载BroPHP框架目录下的入口文件brophp.php

```

后台入口文件名命名：admin.php（可以改为其他名称）

```

1 <?php
2 /**
3      file: index.php  声明基于BroPHP框架开发的项目，单一入口文件示例
4 */
5 define("APP", ".admin/");     #定义项目的应用路径（'/'可加可不加）
6 define("./brophp/brophp.php"); #加载BroPHP框架目录下的入口文件brophp.php

```

自动生成的应用目录结构（后台的 controls、models 和 views 在 admin 目录下）

-- brophp 目录	#BroPHP 框架目录
-- index.php 文件	#前台主入口文件（可以使用其他名称）
-- controls 目录	#声明控制器类的目录（前台控制器目录）
-- models 目录	#声明业务模型类的目录（前台模型目录）
-- views 目录	#声明视图的目录（前台视图目录）
-- admin.php 文件	#后台主入口文件（可以使用其他名称）
-- admin 目录	#自定义的后台项目应用目录
-- controls 目录	#声明控制器类的目录（后台控制器目录）
-- models 目录	#声明业务模型类的目录（后台模型目录）
-- views 目录	#声明视图的目录（后台视图目录）
-- config.inc.php 文件	#项目的配置文件
-- classes 目录	#用户自定义的扩展类目录
-- commons 目录	#用户自定义的扩展函数目录
-- public 目录	#项目的所有应用公用的资源目录
-- runtime 目录	#项目运行时自动生成文件存放目录（可以随时删除）

第二种：前台入口文件和前台应用与框架目录 brophp 在同一级目录中，后台的入口文件和应用目录定义在 admin（可以改为其他名称）下。然后分别访问两个入口文件，即可生成所有目录结构。

第三种：前台和后台入口文件与框架目录 brophp 在同一级目录中，前台和后台的应用目录分别定义在 home（可以改为其他名称）和 admin（可以改为其他名称）目录下。然后分别访问两个入口文件，即可生成所有目录结构。



3. 项目有多个应用

如果项目有多个应用。例如，除了有前台和后台还有博客和论坛，每个应用都需要有独立的入口文件和自己的应用目录。可以让所有入口文件在同一级（例如，与框架在同级目录，但名称不能相同，可以和应用目录名称相同），也可以将每个入口文件放在自己的应用目录中（入口名称就可以统一命名为：index.php）。

26.3.2 URL 访问

BroPHP 框架的 URL 都是使用 PATHINFO 模式（index.php/index/index/），应用的访问方式都是采用单一入口的访问方式，所以访问一个应用中的具体模块及模块中的某个操作，都需要在 URL 中通过入口文件后的参数来访问和执行。这样一来，所有访问都会变成由 URL 的参数来统一解析和调度。格式如下：

`http://www.brophp.com/入口文件/模块名/操作名/参数 1/值 1` #URL 统一解析和调度的 PATHINFO 模式

例如，项目应用代码直接放到主机为 www.brophp.com 的 Web 服务器的文档根目录下，入口文件名为 index.php，访问用户模块（user），再去执行添加（add）的操作，则 URL 的格式如下：

`http://www.brophp.com/index.php/user/add` #通过 URL 统一解析和调度

如果还需要其他参数，例如，在上例添加数据时，需要将数据加到类别 ID 为 5 的类别（cid=5）中，则可以在上例 URL 的操作名后继续加多个参数。则 URL 的格式如下：

`http://www.brophp.com/index.php/user/add/cid/5` #也可以有更多的参数

如果访问某个应用的入口时没有给出需要访问的模块和操作，则默认访问模块为 index，默认访问的操作为 index。下面几个 URL 的访问结果是相同的。

`http://www.brophp.com/
http://www.brophp.com/index.php
http://www.brophp.com/index.php/
http://www.brophp.com/index.php/index
http://www.brophp.com/index.php/index/
http://www.brophp.com/index.php/index/index
http://www.brophp.com/index.php/index/index/`

如果访问某个应用的入口时只给出访问的模块名，没有给出访问模块中的动作名，则默认访问这个模块中的 index 操作。像如访问用户模块（user），下面几个 URL 的访问结果也是相同的。

`http://www.brophp.com/index.php/user
http://www.brophp.com/index.php/user/
http://www.brophp.com/index.php/user/index
http://www.brophp.com/index.php/user/index/`

如果在 URL 访问中除了模块和操作，还需要其他参数，就必须给出模块名和动作名（包括默认的模块 Index 和默认的操作 Index）全格式，再加上多个参数。

`http://www.brophp.com/index.php/index/index/cid/5/page/6` #追加两个参数 cid=5 和 page=6

PATHINFO 模式对以往的编程方式没有影响，GET 和 POST 方式传值依然有效，因为在框架中会

对 PATHINFO 方式进行自动处理。例如，上面 URL 地址中的 cid 的值，在每个操作中还可以通过 \$_GET['cid']的方式正常获取到。

26.4 BroPHP 框架的基本设置

在 BroPHP 框架中，自动开启了一些常用选项，如编码、时区等，用户不需要自己设置。另外，项目需要的配置文件也是自动生成的，并在框架中提供了几个常用的函数。当然，你也可以根据自己的需要，在框架中为具体的项目添加更多的内容。

26.4.1 默认开启

在自定义的入口文件中，最后一行“require(BROPHP.'/brophp.php')”加载了 BroPHP 框架目录下的入口文件 brophp.php。在这个框架入口文件中有一些为整个应用默认开启的功能，所以在项目应用时就不需要再去设置了。除非很有必要，否则默认的设置都不需要进行修改，如表 26-1 所示。

表 26-1 BroPHP 默认开启的功能和描述

默认开启功能	描 述
输出字符集（utf-8）	utf8 字符集是网站和 MySQL 数据库的最佳选择，没有必要做其他改变
设置时区（PRC）	将 PHP 环境中的默认时间区改为中国时区
自动加载项目的配置文件（config.inc.php）	整个项目只有一个配置文件“config.inc.php”，在项目中被自动包含，在用到配置文件中的所有选项时，都可以直接使用
自动包括类库和函数库	在应用中用到的所有类和函数都是自动包含的，进行项目开发时只要按规范去编写，都不需要去特意包含
自动开启 Session	自动开启会话控制，如果启用 memcached，则将用户的会话信息写入到 memcached 服务器，否则使用默认的写入方式

26.4.2 配置文件

Web 项目几乎都需要有配置文件，这样才能更灵活地对项目进行管理和维护。BroPHP 框架在第一次访问时，为整个项目自动创建了一个配置文件 config.inc.php（仅一个），存放在与框架目录同级的目录中，并且默认被包含在程序中，所以在项目开发时配置文件中的选项都可以直接应用。另外，除了配置文件中默认选项可以直接使用以外，还可以自定义添加一些选项在这个文件中，自定义的选项可以是常量，也可以是变量和数组等，如果添加的是变量或数组，则在所有自定义的函数和类中需要使用 global 包含这些全局变量。系统自动创建的配置文件默认选项介绍如表 26-2 所示。

表 26-2 BroPHP 框架配置文件的选项和描述

配置选项	描 述
define("DEBUG", 1);	设置是否开启调试模式（1 开启，0 关闭），建议在开发时使用 1 值开启调试模式，上线运行则使用 0 值将其关闭。默认值为 1 开启



续表

配置选项	描 述
<code>define("DRIVER","pdo");</code>	设置数据库的驱动选项，本系统支持 pdo（默认）和 mysqli 两种驱动方式。在开启 mysqli 时需要 PHP 环境安装 mysqli 扩展模块，在使用 PDO 选项时，除了需要 PHP 环境中安装 PDO 的扩展模块外，还需要安装相应数据库的驱动
<code>//define("DSN","mysql:host=localhost; dbname=xsphp");</code>	当上面的 DRIVER 选项设置为 pdo 时，则可开启这个 PDO 的数据源设置。如果设置了此选项，则可以不用再去设置以下的 HOST、USER、PASS 和 DBNAME 选项
<code>define("HOST","localhost");</code>	数据库系统的主机设置选项，默认为 localhost
<code>define("USER","root");</code>	数据库系统用户名，默认为 root
<code>define("PASS","123456");</code>	数据库系统用户密码，默认为空
<code>define("DBNAME","brophp");</code>	应用的数据库名称，默认为 brophp
<code>define("TABPREFIX","bro_");</code>	设置数据表名的前缀，防止在相同的数据库中保存两个以上 BroPHP 框架开发项目的数据表。另外，这个选项同时也作为 Memcache 的键前缀，作用同表名前缀一样，防止同一个 Memcache 服务器有两个以上的 BroPHP 应用
<code>define("CSTART",0);</code>	这个选项用来设置 Smarty 的缓存，项目开发阶段使用 0 关闭缓存，在项目上线运行时设置 1 将其开启。默认为 0
<code>define("CTIME",60*60*24*7);</code>	这个选项设置 Smarty 模板的缓存时间，同时也是 Session 在 Memcache 中的生存时间。默认值为一周
<code>define("TPLPREFIX","tpl");</code>	Smarty 模板文件的后缀名，视图中所有 Smarty 模板的后缀名都要和这个相同，默认后缀名为 tpl
<code>define("TPLSTYLE","default");</code>	这个选项设置项目使用的模板风格。可以为一个项目开发多套模板风格，使用这个选项进行切换。默认使用的模板风格为 default
<code>// \$memServers = array("localhost", 11211); // \$memServers = array(array("www.lampbrother.net",'11211'), array("www.brophp.com", '11211'), ...);</code>	这个选项用来设置 Memcache 服务器的主机和端口。如果是一个一维数组，则连接一个 Memcache 服务器，也可以是一个二维数组同时连接多个 Memcache 服务器。另外，如果注释没有打开，则没有开启 Memcache 服务器，建议安装 Memcache 服务器并将其开启

26.4.3 内置函数

在 BroPHP 框架中，提供了几个常用的快捷操作的全局函数，在任何位置都可以直接调用这几个函数。包括 P()、D()和 toSize()三个内置函数（当然你可以定义更多常用的函数放在框架中），详细的功能介绍和用法如下。

- **函数 P()：**按照特定格式打印输出一个或多个任意类型（数组、对象、字符串等）的变量或数据，打印的值供程序员作为开发程序时的参考使用，只用于开发阶段的程序调试和排错。使用方式如下所示：

```

$arr=array(1,2,3,4,5,6);
P($arr);                                //可以打印输出 PHP 数组
$object=new Object();
P($object);                            //可以打印输出 PHP 对象
$string="this is a string";
P($string);                            //可以打印输出 PHP 自己串

```

```
$other=其他类型;
P($other);           //可以打印输出 PHP 的任何类型
P($arr, $object, $string, $other); //可以同时打印输出多个 PHP 变量
```

- **函数 D():** 快速实例化 Model 类的对象，实例化 Model 类也只能用这个函数。而且这个函数不仅可以实例化已声明的 Model 类，也可以实例化没有定义的 Model 类（只要参数对应的表名存在即可）。另外，不仅可以声明自己应用中的 Model 类，也可以初例化其他应用中的 Model 类对象。该函数大量用于控制器中，使用方式如下所示：

```
$book = D();           #不用参数，实例化的对象不需要对表进行操作
$book = D("book");     #自定义的 Book 类对象，book 表必须存在
$book = D("book", "admin") #如果有第二个参数，可以实例化 admin 应用下的 Book 类对象
```

- **函数 toSize():** 就是一个普通的功能函数，将字节大小根据范围转成对应的单位（KB、MB、GB 和 TB 等），该函数只有一个参数，就是字节数。

```
toSize(10240);           #结果返回 10KB
```

26.5 声明控制器（Control）

BroPHP 框架是以模块和操作的方式来执行的，一个项目应用中会有多个模块，每个模块又需要单独去调度，所以控制器是一个模块的核心，**建议为每个模块单独声明一个控制器类。**

26.5.1 控制器的声明（模块）

系统会自动到主入口文件指定的应用中，找 controls 目录下面对应的类，如果没有找到，则输出错误报告。例如，一个网上书店中有用户管理（user）、类别管理（cat）和图书管理（book）三个模块，则需要创建三个控制器 User 类、Cat 类和 Book 类和三个模块对应。访问一个模块中的控制器和控制器的操作都需要通过入口文件完成，控制器会管理整个用户的执行过程，负责模块的调度和操作的执行。另外，任何一个 Web 行为都可以认为是一个模块的某个操作，也需要通过入口文件来执行，BroPHP 框架中会根据当前的 URL 来分析要执行的模块和操作。在 URL 访问一节中介绍了 BroPHP 框架的 URL 访问格式。如下所示：

```
http://www.brophp.com/入口文件/模块名/操作名/参数 1/值 1           #URL 统一解析和调度的 PATHINFO 模式
```

例如：

```
http://www.brophp.com/index.php/user/mod/id/5           #URL 访问格式
```

上例是获取当前需要执行项目的入口文件（index.php）、模块（user）和操作（mod），如果有其他的 PATHINFO 参数（/id/5），会转成 get 请求（\$_GET["id"]=5）的格式。在这个例子中，用户访问的是 User 模块，就需要为这个模块定义一个控制器 User 类才能被调度。为模块的控制器在 BroPHP 中有专门的声明位置，声明在当前项目应用目录下的 controls 目录中，类名必须和模块名相同，这个例子中使用 user 模块，就需要创建一个 User 类（每个单词的首字母要大写）保存在 user.class.php 文件中（文件



名和类名相同，所有 BroPHP 中声明的类都要以.class.php 为后缀名)。如下所示：

```
1 <?php
2 /**
3      file: user.class.php 用户模块控制器，必须定义在当前应用的controls目录下
4 */
5 class User {
6     //声明控制器的操作
7 }
```

在自定义的控制器类 User 中，通常不需要去继承其他的类，如果写继承，也只能继承 BroPHP 框架中的基础类 Action，不能其他的继承方式。如下所示：

```
1 <?php
2 /**
3      file: user.class.php 用户模块控制器，如果写继承也只能去继承BroPHP框架中的Action类
4 */
5 class User extends Action{
6     //声明控制器的操作
7 }
```

在自定义的控制器类 User 中，如果不去继承系统中的 Action 类，则默认会继承控制器的通用类 Common。Common 类声明在 common.class.php 文件中，是部署项目应用时自动创建的一个文件，也保存在当前应用的 controls 目录下。Common 类的默认格式如下所示：

```
1 <?php
2 /**
3      file:common.class.php 所有控制器的默认父类，自动生成并定义在当前应用controls目录下
4 */
5 class Common extends Action {
6     function init() {
7         //所有的操作都会行执行这个方法
8         //通常用于设置用户登录
9     }
10
11     //可以自定义一些方法， 作为所有控制器的公用操作方法
12 }
```

用户自定义的控制器类（User）自动继承了 Common 类，而 Common 类又继承了 BroPHP 框架基础类中的 Action 类。所以在 User 类中就可以直接使用从 Action 类中继承过来的所有属性和方法。Common 类存在的目的有两个：

（1）在 Common 类中有一个默认的方法 init()，如果自动继承该类，每个模块中的操作在执行前都会自动调用 init()方法。所以可以在这个方法中完成像用户登录和权限控制等操作。

（2）在 Common 类中也可以自定义一些方法，作为自动继承该类的控制器的公用操作。

26.5.2 操作的声明

在上例的 URL 访问中，除了需要声明控制器 User 类之外，还需要 User 模板定义用户的操作。每一个操作都对应当前模块控制器中的一个方法。例如，上例访问的模块是 user（对应 User 类），而执行的动作 mod（对应 User 类中的 mod 方法），如果后面还有其他 PATHINFO 参数，则将以 GET 方式传给这个方法。如下所示：


```

1 <?php
2  /** file: user.class.php 在controls目录下, 默认继承Common及Action */
3  class User {
4      /* 控制器中默认的方法, 用于获取用户默认的操作, 例如输出用户列表信息 */
5      function index() {
6          // 这个方法的URL访问如下两种方式, 可以不写操作名 (index)
7          // http://www.brophp.com/index.php/user/
8          // http://www.brophp.com/index.php/user/index
9      }
10
11     /* 控制器中声明的方法, 用于获取添加用户界面的操作*/
12     function add() {
13         // 这个操作的访问如下, 模块 (user), 操作 (add)
14         // http://www.brophp.com/index.php/user/add
15     }
16
17     /* 控制器中声明的方法, 用于修改用户的操作*/
18     function mod() {
19         // 这个操作的访问如下, 模块 (user), 操作 (add)
20         // http://www.brophp.com/index.php/user/mod/id/5
21         p( $_GET["5"] ); //输出结果: Array( "id">=5 );
22     }
23
24     /* 控制器中声明的私有的其他方法, 不是一个操作, 最好写到Model中 */
25     private function upload() {
26         //这个用于上传用户头像, 不是操作则不能用URL访问
27     }
28 }

```

26.5.3 页面跳转

自定义的控制器类直接或间接地继承 BroPHP 系统中的基类 Action, 所以 Action 类内置了一些方法, 并可以在每个控制器的方法中直接使用 \$this 进行访问。例如, 开发中经常会遇到一些带有提示信息的跳转页面, 操作成功或者操作错误需要自动跳转到另外一个目标页面。页面跳转在 Action 类中提供了 success() 和 error() 两个方法, 详细的使用方法如下所示。

1. 成功操作跳转 success()

在进行添加或修改等的一些操作时, 如果操作成功, 通常都会自动跳转到一个提示页面, 然后再自动跳转到一个目标页面。Success() 方法是系统 Aaction 类内置的方法, 用在自定义控制器的方法中。这个方法的格式如下所示:

Success(提示消息, [跳转时间], [目标位置])

这个方法有三个参数, 并且都是可选的。其中第一个参数用于在提示页面中输出成功消息, 默认消息就是简单的“操作成功”的提示字样。第二个参数用于设置提示页面的停留时间, 默认为 1 秒 (时间很短, 成功提示没有必要停留时间过长), 可以通过传递一个整数重新设置这个时间 (单位: 秒)。第三个参数是自动跳转的目标位置 (这个位置必须是 PATHINFO 的格式), 如果只有一个字符串 (index) 指定目标方法, 则表示自动跳转到同一个模块的这个方法中。如果使用 “/” 分开的字符串 (模块/操作, 如 user/index), 则表示跳转到其他模块指定的操作中, 也可以在这个参数中使用其他的参数将一些数据带到新的目标操作方法中。如果没有提供第三个参数, 则默认返回 (window.history.back())。常见的用



法如下所示：

<code>\$this->success();</code>	<code>//默认方式</code>
<code>\$this->success("添加成功");</code>	<code>//只有第一个参数</code>
<code>\$this->success("添加成功", 3);</code>	<code>//使用两个参数</code>
<code>\$this->success("添加成功", 3, "user/index");</code>	<code>//使用三个参数</code>
<code>\$this->success("添加成功", 3, "user/index/cid/5");</code>	<code>//可以附加资源</code>

成功的提示界面如图 26-1 所示，可以根据自己的爱好对界面进行修改。在提示界面中，有停止跳转的操作，也可以手动“单击”跳转。

2. 失败操作跳转 error()

在进行添加或修改等操作时，如果操作失败，则需要自动跳转到一个提示页面，查看出错原因，然后再自动跳转到一个目标页面。`error()`方法也是系统 `Aaction` 类内置的方法，也用在自定义控制器的方法中。这个方法的使用方式和 `success()`方法完全相同，只是提示界面和默认的提示消息及跳转时间不同而已。错误的提示界面如图 26-2 所示。



图 26-1 success()方法成功提示界面



图 26-2 error()方法失败提示界面

26.5.4 重定向

如果某个操作（控制器中的方法）执行完成以后，也需要直接转向到其他的操作中，但并不需要一些提示，并且也需要将当前操作中的一些数据带到新的操作中。可以使用从系统基类 `Action` 中继承过来的 `redirect()`方法实现，重定向后会改变当前的 URL 地址。例如，在 `User` 模块的控制器中，执行 `del` 操作成功删除用户后，重定向到自己模块的 `index` 操作中。如下所示：

```

1 <?php
2 /** file: user.class.php 在controls目录下, 默认继承Common及Action */
3 class User {
4     /* 控制器中默认的方法, 用于获取用户默认的操作 */
5     function index() {
6         //默认的操作方法
7     }
8
9     /* 控制器中声明的方法, 用于删除用户的操作 */
10    function del() {
11        //创建用户对象
12        $user = D("user");
13        //使用用户对象中的delete删除某指定的一个用户
14        if( $user->delete($_GET["id"]) ) {
15            //如果删除成功就重定向到本模板的默认操作index中
16            $this->redirect("index");
17        } else {
18            //如果删除失败就跳转到提示界面, 并返回
19            $this->error("删除用户失败");
20        }
21    }
22 }

```

redirect()方法的其他应用

<code>\$this->redirect("模块/动作");</code>	#如果有使用 "/" 分成模块和操作
<code>\$this->redirect("book/add");</code>	#重定向到 book 模块的 add 操作中（例）

如果在重定向到其他操作中时, 还需要带一些参数过去, 还可使用第二个参数以 PATHINFO 的形式将数据传过去。如下所示:

<code>\$this->redirect("模块/动作", "参数");</code>	#使用第二个参数, 传数据 (PATHINFO 格式)
<code>\$this->redirect("book/index", "cid/5/page/3");</code>	#传了 cid 和 page 两个参数 (例)

上例在 redirect()中使用了第二个参数, 在重定向到 book 模块的 index()方法中的同时, 也将 cid=5 和 page=3 两个参数传到了 book 模块的 index()方法中, 在 index()方法中可以直接使用\$_GET 进行接收。

26.6 设计视图 (View)

视图 (View) 是用户看到并与之交互的界面, 对 Web 应用程序来说, 视图扮演着重要的角色。View 层用于与用户的交互, Controller 层是 Model 与 View 之间沟通的桥梁, 它可以分派用户的请求并选择恰当的视图用于显示。BroPHP 框架内置最流行的 Smarty 模板引擎, 所有的视图界面都是 Smarty 编写的模板 (参考前面的 Smarty 章节)。

26.6.1 视图与控制器之间的交互

向视图中分配动态数据并显示输出, 都是在控制器类的某个操作方法中完成的。我们自定义的控制器类都间接地继承了 Smarty 类, 所以在每个控制器类中, 都可以直接使用\$this 访问从 Smarty 类中继承过来的成员。在每个模块控制器的操作中常用的 Smarty 成员如下所示:



```
1 <?php
2 /** file: user.class.php 定义一个控制器类User */
3 class User {
4     /* 控制器中默认的方法 */
5     function index() {
6         //向模板中分配变量
7         $this->assign("data", $data);
8         //输出模板（这个方法在BroPHP框架中改写了）
9         $this->display();
10        //判断模板是否已经被缓存
11        $this->isCached();
12        //消除单个模板缓存
13        $this->clearCache();
14        //消除所有缓存的模板
15        $this->clearAllCache();
16    }
17 }
```

使用\$this 就相当于在使用 Smarty 对象，可以通过\$this->assign()方法向模板（视图）分配变量，并通过\$this->display()方法加载并显示对应的模板。所有使用 Smarty 对象可以完成的操作，这里也都可以实现。

26.6.2 切换模板风格

当前应用下的所有视图，都要将模板声明在当前项目应用的 views 目录下。因为可以为同一个应用程序编写多套模板，所以在 views 目录下声明的每个目录，都是为当前的应用创建的一套独立的模板风格，默认的风格声明在 default 目录下。如果为一个应用编写了几套风格模板，只要修改配置文件中的“TPLSTYLE”选项即可（选项值和目录名对应）。如下所示：

```
/* 修改配置文件 config.inc.php */
define("TPLSTYLE", "default"); //找 views/default/下面的模板风格显示
//define("TPLSTYLE", "home1"); //找 views/home1/下面的模板风格显示
//define("TPLSTYLE", "home2"); //找 views/home2/下面的模板风格显示
```

如果项目中有两个或多个应用，因为 BroPHP 框架只为项目提供一个配置文件，例如，项目分为前台和后台两个应用，所以如果在配置文件中将 TPLSTYLE 改变，则前后台都要有对应的模板。如果只想前台有多套模板风格切换使用，而后台只要一套默认的模板风格不变，就需要将上例的选项“define("TPLSTYLE", "default");”写在每个应用的主入口文件的最上面，因为每个应用的入口文件也可以充当当前应用的子配置文件。

26.6.3 模板文件的声明规则

在每套模板目录下有两个默认的目录 public 和 resource，public 目录下声明的是当前风格的公用模板文件。例如，header.tpl 模板、footer.tpl 模板等，默认有一个 success.tpl 模板，用来显示提示消息框（在控制器中的 success()和 error()两个方法使用）。resource 目录是当前模板风格使用资源目录，包括模板中用到的 CSS、JS 和 Image 等。

在 BroPHP 框架中，对父类 Smarty 中的 display()方法重新改写过，所以声明模板的位置和模板文件

名要按一定的规则。一个项目应用通常都会有多个模块，一个模块又对应一个控制器类，也需要在对应的风格模板目录下，为每个模块单独创建一个目录（目录名和控制类名相同，但全部为小写）。然后，在这个目录下创建和控制器中的操作方法同名的模板文件，模板文件的后缀名由配置文件 `config.inc.php` 中的“`TPLPREFIX`”选择决定，默认是“`.tpl`”，可以修改为`.html`或`.htm`及其他的后缀名。例如，需要在 `user` 模板中的 `add` 操作中输出模板视图，就需要在当前模板风格目录中（`view/default` 目录下），创建一个 `user` 目录，并在 `user` 目录下创建一个模板文件 `add.tpl`。

26.6.4 display()用新用法

`display()`方法重载了父类 `Smarty` 中的方法，其他的参数都没有变化，只是将第一参数的用法改写了。在控制器的操作中，`display()`方法的多种应用形式如下所示：

```

1 <?php
2  /** file: user.class.php  定义一个控制器类User */
3  class User {
4      /** 控制器中默认的方法 */
5      function index() {
6          /**
7           * 如果没有提供参数，默认找和当前模块相同目录名(user)下的
8           * 默认模板文件名为当前操作名(index)，后缀名为tpl（配置文件中可改）
9           * 例如： view/default/user/index.tpl 模板文件
10         */
11         $this->display();
12
13         /**
14          * 如果提供参数没有"/"，默认找和当前模块相同目录名(user)下的
15          * 模板文件名为参数名add，后缀名为tpl
16          * 例如： view/default/user/add.tpl 模板文件
17         */
18         $this->display("add");
19
20         /**
21          * 如果提供参数有"/"，找和"/"前模块同名目录（shop）下的
22          * 模板文件名为参数名add，后缀名为tpl（配置文件可改）
23          * 例如： view/default/shop/add.tpl 模板文件
24         */
25         $this->display("shop/add");
26     }
27 }

```

26.6.5 在模板中的几个常用变量应用

在编写模板文件时，经常会用到链接地址、图片的位置、CSS 文件的地址或是 JS 文件的地址。如果直接写 URL，不仅非常烦琐，而且当域名或主入口文件有改变时，所有 URL 都需要重新修改。所以在控制器的操作中时将一些常用的 URL（和服务器对应）自动分配到了模板中，并可以在模板中直接使用。

```

/* 例如，在 add.tpl 模板中（项目声明在 shop 目录下，入口文件为 admin.php,模块为 index） */
<{$root}>;           //到项目应用的根目录           /shop
<{$app}>;            //到项目应用的主入口文件       /shop/admin.php

```




<code><{\$url}></code> ;	//到访问的模块	<code>/shop/admin.php/index</code>
<code><{\$public}></code> ;	//所有应用的共用资源 <code>public</code>	<code>/shop/public</code>
<code><{\$res}></code> ;	//到模板风格下的 <code>resource</code> 目录	<code>/shop/views/default/resource</code>

例如：

```
<a href="<{$url}>/mod/id/5">修改</a>
<script src="<{$res}>/js/jquery.js"></script>
```

上例会自动解析为：

```
<a href="/shop/admin.php/index/mod/id/5">修改</a>
<script src="/shop/views/default/resource/js/jquery.js"></script>
```

26.6.6 在 PHP 程序中定义资源位置

虽然 BroPHP 框架对所有的类库和函数都是自动包含的，但如果需要在控制器或模型中加载自定义 PHP 某个文件，或是操作一些服务器中的文件，以及设置上传文件目录等，可以使用相对于主入口文件的相对位置。也可以通过 `PROJECT_PATH` 和 `APP_PATH` 两个路径完成。如下所示：

- **PROJECT_PATH** 代表项目所在的根路径，即与框架所在的目录同级
- **APP_PATH** 代表项目中当前应用目录（在入口文件中指定的应用路径）

另外，除了在模板中可以直接使用 `<{$root}>`、`<{$app}>`、`<{$url}>`、`<{$public}>`、`<{$res}>` 等路径，如果不是使用模板文件，而是在 PHP 中直接去访问前台文件（JS、CSS、HTML、图片等），则可以使用 BroPHP 框架中的几个常量或几个全局 `$GLOBALS` 变量，都是从 Web 服务器根目录开始的绝对路径。如下所示：

- `B_ROOT` 或 `$GLOBALS["root"]` //Web 服务器根到项目的根
- `B_APP` 或 `$GLOBALS["app"]` //当前应用脚本文件
- `B_URL` 或 `$GLOBALS["url"]` //访问到当前模块
- `B_PUBLIC` 或 `$GLOBALS["public"]` //项目的全局资源目录
- `B_URL` 或 `$GLOBALS["res"]` //当前应用模板的资源

还有就是可以在每个模板的操作中，通过 `$_GET["m"]` 获取当前访问的模块名称，也可以通过 `$_GET["a"]` 访问当前的操作名称。

26.7

应用模型（Model）

模型（Model）就是业务流程/状态的处理及业务规则的制定。业务流程的处理过程对其他层来说是黑箱操作，模型接受从控制器请求的数据，并返回最终的处理结果。业务模型的设计可以说是 MVC 最主要的核心。在 BroPHP 中基础的模型类就是内置的 `DB` 类，该类完成了基本的数据表增、删、改、查、连贯操作和统计查询，一些高级特性都被封装到模型的基类中。

26.7.1 BroPHP 数据库操作接口的特性

编写程序的业务逻辑最烦琐的地方，就是对不同数据表的反复编写、执行及处理 SQL 语句（增、删、改、查）。降低网站性能的最大开销是在程序中执行大量的 SQL 查询，攻击网站最常见的方式是使用 SQL 注入。但在 BroPHP 框架中解决了这些问题，系统模型基类的一些基本特性如下所示。

1. 重用性

BroPHP 内置了抽象数据库访问层，把不同的数据库操作封装起来，而使用了统一的操作接口。只需要使用公共的 DB 类进行 SQL 操作，而无须针对不同的数据表写重复的代码和底层实现。

2. 高效性

BroPHP 框架的 Model 中，所有的 SQL 语句都是通过 `prepare()` 和 `execute()` 方法去准备和执行的，效率要比直接使用 `query()` 方法高得多。另外，最主要的是在 BroPHP 中所有的查询结果都使用 `memcached` 进行缓存，所以只要获取一次结果集，同样的查询下次不管再执行多少次，都不需要再重新连接数据库了，而是直接从 `memcached` 中获取数据，这样可以大大提高网站的性能。并且如果有执行对表有影响的 SQL 语句，就会清除该表的缓存，所以还可以达到动态更新的效果。

3. 安全性

每个 SQL 语句都是使用 PDO 或 `mysqli` 中的预处理方式，并通过“？”参数绑定的形式先将语句在服务器中准备好，再为这个“？”绑定的任何“值”都不会再重新编译 SQL 语句，所以 BroPHP 框架没有 SQL 注入的可能。

4. 简易性

BroPHP 框架为所有自定义 Model 类的实例化提供了统一的内置函数 `D()` 来实现，简化了 Model 类的对象创建过程。而且所有的 SQL 查询都可以采用连贯操作方式，并使用系统中内置的方法就可以以最简单的方式完成对数据表的操作。

5. 扩展性

BroPHP 框架中 Model 类之间的继承关系简单明了，很容易通过自己定义的 Model 类对系统中内置 DB 类的功能进行扩展，完成特定的功能。

6. 维护性

BroPHP 框架中所有和 SQL 语句相关的执行都汇总到了一个操作中，并有统一的处理方式。这样就可以大大提高 Model 类的可维护性。

26.7.2 切换数据库驱动

BroPHP 框架支持 `mysqli` 和 PDO 两种连接方式的驱动，并且都是使用它们的“预处理”方式来处理 SQL 语句，这样不仅效率高，而且能防止 SQL 注入。默认是使用 PDO 的连接方式（推荐使用 PDO，除了可以连接 MySQL 数据库，还可以连接其他数据库）。不管使用哪种连接方式，在使用前要先安装 PHP 扩展库，PDO 还需要安装对应的数据库驱动。切换的方式也很容易，只要修改配置文件（和框架



在相同目录的 config.inc.php 文件) 中的一个参数, DB 类就会自动调用相应的数据库适配器来处理。如下所示:

```
/*项目的配置文件 Config.inc.php (和框架同级目录下)*/
define("DRIVER","pdo"); //pdo(默认)和可改成 mysqli
```

在 Model 中如果能正确地连接数据库, 除了在配置文件中设置上例的选择数据库驱动方式外, 还需要配置数据库的连接用户和密码, 以及数据库的库名。如下所示:

```
/* 正确的配置数据库的连接 */
define("HOST","localhost"); // 数据库服务器的主机位置
define("USER","root"); // 数据库服务器的登录用户
define("PASS",""); // 数据库登录用户的密码
define("DBNAME","brophp"); // 数据库的库名
define("TABPREFIX","bro_"); // 数据表的名称前缀
```

如果选择 PDO 来连接数据库, 还可以使用 DSN (数据源名) 的方式来配置数据库的连接, 这样的配置不仅可以使用 PDO 连接 MySQL 数据库, 还可以连接其他数据库。也是通过在配置文件中修改配置, 如下所示:

```
/*使用 DSN 的方式配置 PDO 连接数据库*/
define("DSN","mysql:host=localhost;dbname=brophp"); //DSN 方式
```

如果使用 DSN 的方式配置数据库连接, 则不用再去配置 HOST 和 DBNAME 两个选项了, 因为在 DSN 中都有设置了。

26.7.3 声明和实例化 Model

所有对数据表的操作都需要使用 BroPHP 的 Model 完成, 而不管是自定义 Model 类(DB 类的子类), 还是直接使用系统内置的数据库操作类, 都需要使用内置的 D()方法来实例化一个 Model 类对象。

1. 声明自定义的 Model 类

在配置文件中配置好与数据库连接有关的选项以后, 就可以为数据表声明一个对应的 Model 类来处理它了, 自定义的 Model 类名必须和数据表名相同(BroPHP 采用的是通过类名找对应的表进行处理)。例如, 数据库中有三张表 bro_books、bro_users 和 bro_articles (其中 bro_为表名前缀, 会自动处理), 就需要在当前应用的 models 目录下创建 books.class.php、users.class.php 和 articles.class.php 三个文件(类名不用加表名前缀名)。在每个文件中只声明一个对应的类, 如果不去写继承, 会自动继承系统中的 DB 类, 就可以直接使用从 DB 类中继承过来的内置方法操作数据表了。自定义的 Model 类 Users 如下所示:

```

1 <?php
2 /**
3     file: users.class.php 自定义处理users表的Model类, 声明在当前应用下的models目录下
4     默认继承系统内置DB类, 可以直接使用所有从DB类中继承过来的方法
5 */
6 class Users {
7     /* 声明一个用户登录的方法 */
8     function isLogin() {
9         //方法体
10    }
11
12    /* 声明一个退出系统的方法 */
13    function isLogout() {
14        //方法体
15    }
16 }

```

如果有两个 Model 类需要声明一个父子类, 用于构建共用的属性和方法, 系统也直接支持使用 `extends` 继承一个自定义的一个公用父类, 但主动继承的父类也会自动继承系统内置的 **DB** 类。所以自定义的 Model 类还是间接地继承了 DB 类, 这样除了可以直接使用自定义父类中的成员, 还可以直接使用系统内置类 DB 中的成员。自定义的 Model 类 Books 继承自定义的 Demo 类, 如下所示:

```

1 <?php
2 /**
3     file: Demo.class.php 在models目录下, 将来会自动继承DB类作为多个Model的父类
4 */
5 class Demo {
6     //声明一个功能方法
7     function fun() {
8         //多个子类可以共用这个方法
9     }
10 }

```

自定义的 Model 类 Books 主动使用 `extends` 继承上例中的 Demo 类, 如下所示:

```

1 <?php
2 /**
3     file: books.class.php 声明一个类Books主动继承Demo类, 间接继承了DB类
4 */
5 class Books extends Demo {
6     //在这个类中可以使用Demo类和系统内置DB中的所有继承过来的成员
7 }

```

在声明好一个 Model 类之后, 就可以在当前项目应用的控制器中, 使用系统内置的 `D()` 函数去实例化这个 Model 类的对象, 再通过这个对象就可以直接对业务进行处理了。在使用 `D()` 函数时, 需要提供一个参数, 参数必须是自定义的 Model 类名 (也是要处理的表名称)。例如, 声明好了一个 User 模型类后, 在 User 控制器中的使用过程如下所示:



```
1 <?php
2 /**
3      file: user.class.php 在controls目录下，声明用户模块控制器，默认继承Common及Action
4 */
5 class User {
6     /* 控制器中默认的方法，用于获取用户默认的操作 */
7     function index() {
8         //创建用户对象， 参数user找模型中的User类创建
9         $user = D("user");
10        $data = $user->select(); //调用父类中的方法查询表中所有记录
11        // $user -> isLogin(); //也可以调用User类中声明的方法
12    }
13 }
```

在使用 D()函数实例化模型类时，系统会自动通过参数字符串找到对应的数据表。如果这个数据表是第一次操作，则系统会自动获取表结构并缓存一起来，以后的每次操作都是从缓存中直接获取表结构，不会每次都重新连接数据库反复获取表结构。

2. 直接使用内置 DB 类

如果只需使用系统内置 DB 类中的功能就可以完成对业务的处理，则没有必要单独声明一个空（没有成员）的 Model 类（只有需要对某个表有特定的操作，DB 类没有提供的功能，才去自定义 Model 完成一些特定的处理），也是使用 D()函数实现。例如，没有声明对用户表（user）操作的模型类时，使用 D()直接传表名（不用加前缀）作为参数（用于获取表结构），就可以实例化一个 DB 类的对象，完成对 user 表的操作。如下所示：

```
1 <?php
2 /**
3      file: user.class.php 在controls目录下，声明用户模块控制器
4 */
5 class User {
6     /* 控制器中默认的方法，用于获取用户默认的操作 */
7     function index() {
8         $user = D("user"); //模型User类不存在， 参数为表名
9         $data = $user -> select(); //调用系统DB类中的方法查询表中所有记录
10    }
11 }
```

3. 使用跨应用的 Model 类

如果项目中有前台和后台两个应用（也可以有更多的应用），是否需要各自定义一个业务模型对同一个表进行操作呢？例如，在后台应用（admin）中的 model 目录下声明一个 User 类，类中声明了处理用户登录和退出的方法，如果在前台应用中的 model 目录下也声明一个 User 类，在类中再写一次处理用户登录和退出的方法，就会发生代码重复编写的情况。所以在 BroPHP 框架中对同一个项目有多个应用时，相同表的处理可以使用同一个 Model 类来完成。当然也是使用系统内置的 D()函数完成，只不过除了使用第一个参数传一个类名外（或是表名），还需要使用第二个参数传另一个应用的目录名（入口文件中声明的应用目录名同名）。例如，在前台应用的控制器 Index 类中的 index()方法中，使用后台应用（在 admin 目录下）中的模型 User 类处理 user 表。D()函数的使用如下所示：


```
1 <?php
2 /**
3  * file: index.class.php 在前台controls目录下声明的主控制器
4  */
5 class Index {
6     /* 控制器中默认的方法 */
7     function index() {
8         /* 创建后台admin目录中models目录下的User类对象 */
9         $user = D("user", "admin");
10        $user -> isLogin(); //在前台调用后台User类中的登录方法
11    }
12 }
```

4. 没有为 D()方法提供参数

如果在使用 D()方法时没有提供参数，则也可以创建 Model 类对象，但不能对数据表进行操作，只能完成一些非表操作的功能，例如获取数据库的使用大小、获取数据库系统的版本、事务处理等。D()函数的使用如下所示：

```
1 <?php
2 /**
3  * file: index.class.php 在前台controls目录下声明的主控制器
4  */
5 class Index {
6     /* 控制器中默认的方法 */
7     function index() {
8         //如果没有传表名或类名，则直拉创建DB对象，但不能对表进行操作
9         $db = D(); //可以访问DB对象中非表的操作方法
10
11        $db -> dbSize(); //获取数据库的空间使用信息
12        $db -> dbVersion(); //获取数据库系统的版本
13        $db -> beginTransaction(); //开启事务
14        $db -> commit(); //提交事务
15        $db -> rollback(); //回滚事务
16    }
17 }
```

26.7.4 数据库的统一操作接口

BroPHP 框架中为所有对表的操作提供了统一的接口，这样不仅可以省去编写 SQL 语句的烦恼，也不用考虑 SQL 语句的执行效率和 SQL 优化及 SQL 注入等安全问题，因为所有 SQL 语句都已经在框架中封装好了。并且这些接口操作简单，符合程序员的开发习惯。在 BroPHP 框架中提供的数据库操作接口和描述如表 26-3 所示。

表 26-3 数据库的操作接口及描述

方 法 名	描 述
insert()	向表中新增数据，返回最后插入的自动增长 ID
update()	更新表中的数据，返回更新的影响行数
delete()	删除表中的数据，返回删除的影响行数
field()	连贯操作时使用，设置查询的字段，返回对象\$this
where()	连贯操作时使用，设置查询条件，返回对象\$this
order()	连贯操作时使用，设置 SQL 的排序方式，返回对象\$this



limit()	连贯操作时使用，设置获取的记录数，返回对象\$this
group()	连贯操作时使用，设置 SQL 的分组条件，返回对象\$this
having()	连贯操作时使用，设置分组时的查询条件，返回对象\$this
total()	获取符合条件的记录总数
find()	获取数据表的单条记录，返回一维数组
select()	获取数据表的多条记录，返回二维数组
r_select()	关联查询，从有关联的多个表中获取数据
r_delete()	关联删除，一起删除多个表中的有关联的记录
query()	任意的 SQL 语句都可以使用该方法执行
beginTransaction()	开启事务处理操作
commit()	提交事务
rollback()	事务回滚
dbSize()	获取数据库使用大小
dbVersion()	获取数据库的版本
setMsg()	设置提示信息，该方法设置的消息可以通过 getMsg()方法获取
getMsg()	获取一些验证信息，提示给用户使用，可以一起获取多条，以字符串返回

以上的每个方法在使用时都不用提供表名，因为在使用这些方法时要先为数据表创建对应的 Model 类对象，在使用 D()函数创建对象时已经传递了表名，也自动获取了表结构。每个方法的详细使用如下所示。

1. insert([array \$post][, mixed filter][,bool validata])

该方法是向数据表中新增一条记录，只要为该函数提供正确的新增所需要的数据（是一个数组），就可以直接插入到表中。通常都是在控制器中接收表单提交过来的数据，再在控制器中调用 Model 类中的这个方法完成添加数据。在向表中新增数据时需要注意以下两点。

（1）所有 Form 表单的提交方法最好使用“post”方式。

（2）每个表单项的名称一定要和数据表的字段名相同，只有相互对应的项才能加入到表中。

该函数有三个可选参数，如果没有提供第一个参数，则默认是将表单提交过来的数组\$_POST 作为第一个参数。也可以直接将\$_POST 数组作为第一个参数传递，当然也可以根据自己的需要组合一个数组后再传递给第一个参数。例如，bro_users 表结构如下所示：

```

Create table bro_users(
    id int not null auto_increment,
    name varchar(30) not null default "",
    age int not null default 0,
    sex char(4) not null default '男',
    ptime int not null default 0,
    email varchar(60) not null default "",
    primary key(id)
);

```

#表名为 bro_users
#用户编号 ID
#用户名
#用户年龄
#用户性别
#用户注册时间
#用户电子邮箱

表单提交过来的数组\$_POST 如下所示：

```

$_POST=array(
    "name"=>"admin",
    #<input name="name">

```

```

"age"=>"22",
"sex"=>"男",
"email"=>"gaolf@php.net",
"sub"=>"注册"
);
#<input name="age">
#<input name="sex">
#<input name="email">
#<input name="sub" type="submit">

```

从\$_POST 数组中可以看到表单中提交过来的数组，没有提交 id（表中是自动增长的）和 ptime（注册时间需要从 PHP 服务器自动获取）。而和 bro_users 表字段不一样的还多了一个名称为 sub 的提交按钮。在控制器的方法中的使用如下所示：

```

1 <?php
2 /**
3  * file: user.class.php 在controls目录下，声明用户模块控制器
4  */
5 class User {
6     /* 控制器中添加方法 */
7     function add() {
8         $user = D("user"); //创建用户实例对象
9
10        $_POST["ptime"] = time(); //向$_POST数组中添加用户注册时间
11        $id = $user->insert(); //默认使用$_POST数组作为参数
12        // $id = $user -> insert($_POST); //也可以直接传递$_POST参数
13    }
14 }

```

按上例 insert()方法的使用，内部将组合成一个准备好的语句。如下所示：

SQL: "INSERT INTO bro_users(name,age,sex,ptime,email) values(?,?,?,?)"
 对应的数组绑定？参数 array("admin", "22", "男", "123322122", "gaolf@brophp.net");

在 insert()方法内部有一个处理，会将传递过来的\$_POST 数组下标和表字段名称进行匹配，如果有匹配成功的，说明表单项的名称和数据表的字段名称相同。例如“sub=>注册”中，下标“sub”就不是表的字段名，所以在组合 SQL 语句时将其去掉。

insert()方法需要的第二个参数\$filter，默认值是 1（只要是“真”值都可以），这个参数决定是否对表单传递过来的数据进行过滤。因为表单是黑客攻击网站的主要入口，所以为了防止用户在表单中输出一些不允许的 HTML 标记或恶意的 JavaScript 代码，在 insert()中使用 PHP 中内置的两个方法 stripslashes()和 htmlspecialchars()进行了处理，不仅能将 HTML 标记转为了 HTML 实体，同时也去掉了在表单中输入的单引号或双引号自动添加的转义符号。特定情况下可以使用 0 值（只要是“假”值都可以）关闭这个过滤功能。如果使用一个数组作为参数，数组中的元素为表单名称，则也可以部分关闭。

insert()函数也可以提供第三个参数\$validata，默认值是 0（只要是“假”值都可以），这个参数决定是否需要使用 XML 对数据进行自动验证，“假”值是不需要验证的。

insert()方法执行成功返回最后自动增长的 ID，失败返回 false，如果数据表没有自动增长的字段，成功返回 true。

2. update([array \$array][, int filter] [, bool validata])

该方法用于更新数据表中的记录，有三个可选参数，第二个和第三个参数与 insert()方法中的两个参数一样，用于设置表单过滤功能和设置自动验证。该方法可以以主键为条件更新一条记录，也可以以自己设置的条件同时更新多条记录，还可以设置更新特定的字段。例如，数据表 bro_users 的结构同上，



update()方法的常用的几种使用方式如下所示。

第一种：最常用 update()方法更新一条数据，将修改表单提交过来的\$_POST 数组直接传给该函数的第一个参数（不需要修改的字段可以在\$_POST 数组中去掉），则会以\$_POST 数组中和表主键字段名称相同的元素下标，作为条件更新一条记录。例如，\$_POST 数组中的内容如下：

```
$_POST=array(
    "id"=>"5",
    "name"=>"admin",
    "age"=>"25",
    "sex"=>"男",
    "email"=>"gaolf@php.net",
    "sub"=>"修改"
);
```

```
#<input name="name" type="hidden">
#<input name="name">
#<input name="age">
#<input name="sex">
#<input name="email">
#<input name="sub" type="submit">
```

这里要注意，修改表单的名称中一定要有一个对应表的主键（本例是 ID，通常使用隐藏表单传递），将这个\$_POST 作为每一个参数传入 update()方法。使用和组合后的 SQL 语句如下：

```
1 <?php
2 /**
3     file: user.class.php 在controls目录下，声明用户模块控制器
4 */
5 class User {
6     /* 控制器中修改方法 */
7     function mod() {
8         $user = D("user"); //创建用户实例对象
9
10        $rows = $user -> update(); //默认使用$_POST数组作为参数
11        // $rows = $user -> update($_POST); //也可以直接传递$_POST参数
12    }
13 }
```

SQL: "UPDATE bro_users SET name=?,age=?,sex=?,email=? WHERE id=?";
对应的数组绑定？参数 array("admin", "25", "男", "gaolf@php.net", 5);

第二种：可以通过 where()方法（详见 where()方法）使用连贯操作，设置更新的条件去更新一条或多条记录。例如，使用 where()方法设置条件更新主键值为 1、2 和 3 的三条记录，使用和组合后的 SQL 语句如下：

```
1 <?php
2 /**
3     file: user.class.php 在controls目录下，声明用户模块控制器
4 */
5 class User {
6     /* 控制器中修改方法 */
7     function mod() {
8         $user = D("user"); //创建用户实例对象
9
10        //默认使用$_POST数组作为参数(可以默认)，加上where()连贯操作
11        $rows = $user -> where("1, 2, 3") -> update($_POST);
12    }
13 }
```

SQL: "UPDATE bro_users SET name=?,age=?,sex=?,email=? WHERE id in(?,?,?);"
对应的数组绑定？参数 array("admin", "25", "男", "gaolf@php.net", 1, 2, 3);

第三种：在前两种方式的基础上，还可以使用 `update()` 方法更新指定的字段。例如，计算一篇文章的访问数，访问一次访问数字段值就累加一次。本例设置 `bro_users` 表中 `id` 为 5 的记录中年龄字段(`age`)的值累加 1。也是使用 `update()` 方法的第一参数实现，只要在参数中使用一个字符串，这个字符串就是 SQL 语句中的 `SET` 后面的设置内容。使用和组合后的 SQL 语句如下：

```
D('users ')->where(array("id"=>5))->update("age=age+1");
```

SQL: "UPDATE bro_users SET age=age+1 WHERE id=?";
对应的数组绑定？参数 `array(5)`;

另外，`update()` 方法也可以和 `limit()` 及 `order()` 两个方法组合使用。例如，将最新添加的 5 条用户记录的性别 (`sex` 字段) 都改为“女”。就需要使用 `order()` 方法倒序排列，并使用 `limit()` 方法限制 5 条记录被修改。使用和组合后的 SQL 语句如下：

```
D('users ')->order("id desc")->limit(5)->update(array("sex"=>"女"));
```

SQL: "UPDATE bro_users SET sex=? ORDER BY id DESC LIMIT 5";
对应的数组绑定？参数 `array('女')`;

`update()` 方法执行成功后，返回影响记录的行数，没有行数影响可以作为 `false` 值使用。

3. delete()

该方法用于删除数据表中的记录，可以以主键为条件删除一条记录，也可以按自己设置的条件同时删除多条记录。其实 `delete()` 方法和 `where()` 方法（详见 `where()` 方法）的参数是一样的，可以任意设置条件删除记录。例如，数据表 `bro_users` 的结构同上，`delete()` 方法的常用的几种使用方式如下所示。

第一种：如果你想一次一条地单个记录删除，只要将主键（通常是 `id`）值作为参数传入即可。使用和组合后的 SQL 语句如下：

```
1 <?php
2 /**
3     file: user.class.php 在controls目录下，声明用户模块控制器
4 */
5 class User {
6     /* 控制器中删除的方法 */
7     function del() {
8         $user = D("user");           //创建用户实例对象
9
10        //使用$_GET数组传过来的主键作为参数删除一条， 例如$_GET['id'] = 5;
11        $rows = $user -> delete( $_GET['id'] );
12    }
13 }
```

SQL: "DELETE FROM bro_users WHERE id=?";
对应的数组绑定？参数 `array(5)`;

第二种：通常在用户列表中可以通过复选框选中多条记录以后一起删除，只要将多条记录的主键（像 `id`）组合成数组作为参数传入 `delete()` 方法即可。使用和组合后的 SQL 语句如下：



```

1 <?php
2 /**
3      file: user.class.php 在controls目录下，声明用户模块控制器
4  */
5  class User {
6      /* 控制器中删除的方法 */
7      function del() {
8          $user = D("user");           //创建用户实例对象
9
10
11      /*
12          $_POST数组中是传过来的多个主键（复选框中选中id为前5条）
13          例如: $_POST = array("id" => array(1, 2, 3, 4, 5));
14      */
15      $rows = $user -> delete( $_POST['id'] );
16  }

```

SQL: "DELETE FROM bro_users WHERE id IN(?,?,?,?);"

对应的数组绑定？参数 array(1,2,3,4,5);

当然，delete()方法也可以有其他用法，例如删除“id > 5”的所有记录，或是删除名称中包含“php”字符串的记录，也可以和 where()组成连贯操作一起使用，总之条件可以任意设置。如下所示：

```

D('users ')->delete(array("id ">=>5));           //删除 id > 5 的记录
或
D('users ')->where(array("id ">=>5))->delete();    //同上

```

SQL: "DELETE FROM bro_users WHERE id > ?";

对应的数组绑定？参数 array(5);

为了防止条件组合不成立时误删除表中全部记录，在使用 delete()方法时如果 where 条件为空或不成立，则不会删除任何记录。另外，delete()方法除了可以和 where()方法一起使用，也可以和 limit()及 order()两个方法组合使用。例如，删除最新添加的 5 条记录，使用和组合后的 SQL 语句如下：

```
D('users ')->order("id desc")->limit(5)->delete();
```

SQL: "DELETE FROM bro_users ORDER BY id DESC LIMIT 5";

delete()方法执行成功后，返回影响记录的行数没有行数影响可以作为 false 值使用。

4. find()

该方法用于从一个数据表中获取满足条件的一条记录，以一维数组的方式返回查找到的结果。经常用在修改数据时先通过该方法获取一条记录放到修改表单中，也会用在用户登录时获取当前用户信息。这个方法常用的使用方式有两种。

第一种：直接通过参数传入需要查找记录的主键（通常为 id），返回主键对应记录的一维数组，使用和组合后的 SQL 语句如下：

```

1 <?php
2 /**
3  * file: user.class.php 在controls目录下, 声明用户模块控制器
4  */
5 class User {
6     /* 控制器中修改用户的方法 */
7     function mod() {
8         $user = D("user"); //创建用户实例对象
9
10        $data = $user -> find( $_GET['id'] ); //$_GET['id'] = 5
11        p( $data ); //打印结果数组
12    }
13 }

```

SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE id=? LIMIT 1";

对应的数组绑定? 参数 array(5);

结果数组: Array("id"=>5, "name"=>"zs", "age"=>20, "sex"=>"男", "email"=>"a@b.c");

第二种: 可以通过 where()方法（详见 where()方法）和 field()方法（详见 field()方法）使用连贯操作，自己定义查询条件和查找指定的字段。例如，在用户登录时，通过用户提交的用户名和密码到数据库中查找用户注册过的信息。如下所示：

```

D('users') -> field('id,username')
-> where(array("username"=>$_POST['username'], "pass"=>$_POST['pass']))
-> find();

```

SQL: "SELECT id,username FROM bro_users WHERE username=? AND pass=? LIMIT 1";

对应的数组绑定? 参数 array("admin", "123456");

结果数组: Array("id"=>1, "username"=>"admin");

5. field()

该方法不能单独使用，需要和 find()或 select()方法一起使用，形成连贯操作去组合一个 SQL 语句，用于设置查询指定的字段。用法很简单，只要在 SQL 语句的“SELECT”和“表名”之间可以写的内容都可以写在这个方法的参数中。例如，和 find()方法一起使用，如下所示：

```
D('users')->field("id,name,sex")->find(5);
```

SQL: "SELECT id,name,sex FROM bro_users WHERE id = ? LIMIT 1";

对应的数组绑定? 参数 array(5);

或设置查找字段时为字段指定别名，如下所示：

```
D('users')->field("id as '编号',name '用户名',sex '性别'")->find(5);
```

SQL: "SELECT id as '编号',name '用户名',sex '性别' FROM bro_users WHERE id = ? LIMIT 1";

对应的数组绑定? 参数 array(5);

6. where()

该方法也不能单独使用，需要和 find()、select()、update()、total()或 delete()等方法之一一起使用，形成连贯操作去组合一个 SQL 语句，用于设置查询条件。例如，前面见过和 find()、update()及 delete()方法配合使用的方式。使用这个方法设置查询条件非常灵活，有很多种使用方式，基本上可以通过这个方法组合成任意的查询条件，这个方法常用的使用方式如下。



第一种：如果没有传递参数，或条件为空（例如："、0、false 等），则在 SQL 语句中不使用 where 条件。例如，从 bro_users 表中使用 select() 获取数据，但组合条件 where 条件时没有传参，如下所示：

```
D('users ')->where('')->select();           //where('')参数为空，或 0、false
SQL: "SELECT id,name,age,sex,email FROM bro_users";    //没有 where 条件
```

第二种：如果直接在这个方法的参数中传一个整数，则组合的 where 条件就是直接设置主键（通常为自动增长的 id）的值。例如，从 bro_users 表中查找 id（主键）为 5 的记录。如下所示：

```
D('users ')->where(5)->select();           //使用整数作为参数
SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE id=?";
对应的数组绑定？参数 array(5);
```

第三种：如果使用以逗号分隔的数字字符串或使用一维的索引数组作为参数，则组合的 SQL 语句通过 IN 关键字为主键设置多个查询的值。例如，从 bro_users 表中查找 id（主键）为“1,2,3”的三条记录。两种方式如下所示：

```
D('users ')->where('1,2,3')->select();      //使用数字字符串作为参数
D('users ')->where(array(1,2,3))->select();  //使用一维的索引数组作为参数
SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE id IN(?,?,?);
对应的数组绑定？参数 array(1,2,3);
```

第四种：如果是以一个关联数组作为参数，数组中的第一个元素还是一个数组（二维数组），则组合的 SQL 语句通过 IN 关键字设置多个查询的值，元素下标作为字段名。例如，从 bro_articles 表中查找 uid（非主键）为“1,2,3”的三条记录。如下所示：

```
D('users ')->where(array("uid"=>array(1,2,3)))->select();//二维数组参数
SQL: "SELECT id,title,content FROM bro_articles WHERE uid IN(?,?,?);
对应的数组绑定？参数 array(1,2,3);
```

第五种：如果是以一个关联数组作为参数，则数组的下标是数据表的字段名，数组的值是这个字段查询的值。例如，从 bro_users 表中查找性别（sex）为“男”所有记录。如下所示：

```
D('users ')->where(array("sex"=>"男"))->select();    //使用关联数组作为参数
SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE sex=?";
对应的数组绑定？参数 array("男");
```

第六种：如果还是以以一个关联数组作为参数，但在数组的值中使用两个百分号（“%值%”，则会组合成模糊查询的形式。例如，从 bro_users 表中查找名字（name）中包含字符串“feng”的所有记录。如下所示：

```
D('users ')->where(array("name"=>"%feng%"))->select(); //使用关联数组作为参数
SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE name LIKE ? ";
对应的数组绑定？参数 array("%feng%");
```

第七种：也是以一个关联数组作为参数，但关联数组的下标中使用“空格”分为两部分，空格前面是指定数据表的字段名，空格后面是指定的查询运算符。例如，从 `bro_users` 表中查找年龄（age）大于“20”岁的所有记录。如下所示：

```
D('users ')->where(array("age >"=>20))->select(); //使用关联数组作为参数
```

```
SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE age > ? ";
对应的数组绑定？参数 array(20);
```

第八种：如果参数的关联数组是由多个元素组成的，则是设置多个 `where` 条件，多个条件之间使用“and”隔开，是“逻辑与”的关系。例如，从 `bro_users` 表中查找年龄（age）大于“20”岁，并且性别（sex）为“男”的所有记录。如下所示：

```
D('users ')->where(array("age >"=>20, "sex"=>"男"))->select(); //数组中多个元素
```

```
SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE age >? AND sex=?";
对应的数组绑定？参数 array(20, "男");
```

第九种：如果参数是多个关联数组，则也是设置多个 `where` 条件，但多个条件之前使用“or”隔开，是“逻辑或”的关系。例如，从 `bro_users` 表中查找名字（name）中包含字符串“feng”的，或者性别（sex）为“男”的所有记录。如下所示：

```
D('users ')->where(array("name"=>"%feng%"), array("sex"=>"男"))->select();
```

```
SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE name LIKE ? OR sex=?";
对应的数组绑定？参数 array("%feng%", "男");
```

第十种：也是最后一种，如果直接以字符串作为参数，就像直接写 SQL 语句中 `where` 条件一样。如果能通过前面几种方式完成 `Where` 条件设置就尽量不使用这种方式，因为这种方式不能使用“?”参数，也就不能防止 SQL 注入。例如，从 `bro_users` 表中查找年龄（age）大于“20”岁，并且性别（sex）为“男”的所有记录。如下所示：

```
D('users ')->where("age > 20 AND sex='男' ")->select(); //直接使用字符串参数
```

```
SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE age >20 AND sex='男' ";
```

7. order()

该方法也不能单独使用，需要和 `select()`、`delete()`、`update()` 方法及其他连贯操作的方法一起使用，用于设置 SQL 的排序条件，默认所有表都是按主键（通常为 `Id`）正序排序。如果需要改变查询结果的排序方式，就可以通过这个方法实现。例如，从 `bro_users` 表中查找年龄（age）大于“20”岁的用户，并按年龄从大到小排序。如下所示：

```
D('users ')->where(array("age >"=>20))->order("age desc")->select();
```

```
SQL: "SELECT id,name,age,sex,email FROM bro_users WHERE age >20 ORDER age DESC";
```

//或删除年龄大于 20 岁的最后 5 条记录：

```
D('users ')->where(array("age >"=>20))->order("age desc")->limit(5)->delete();
```



```
SQL: "DELETE FROM bro_users WHERE age > 20 ORDER age DESC Limit 5";
```

8. limit()

该方法也不能单独使用，需要和 `select()`、`delete()`、`update()` 方法及其他连贯操作的方法一起使用，用于 SQL 语句限制查询记录的个数。可以使用的方式有两种：

第一种：直接使用一个整数作为参数，就是限制记录的个数，如下所示：

```
D('users')->limit(10)->select(); //取 10 条记录
```

```
SQL: "SELECT id,name,age,sex,email FROM bro_users LIMIT 10";
```

第二种：可以使用两个整数作为参数（也可以以逗号分隔开两个数字的字符串作为参数），分别设置从哪条记录开始查询和取多少条记录，如下所示：

```
D('users')->limit(30,10)->select(); //从 30 条开始取，取 10 条记录，两个数字参数
```

```
D('users')->limit('30,10')->select(); //从 30 条开始取，取 10 条记录，字符串参数
```

```
SQL: "SELECT id,name,age,sex,email FROM bro_users LIMIT 30,10";
```

9. group()

该方法也不能单独使用，需要和 `select()` 方法及其他连贯操作的方法一起使用，用于为数据表的查询记录设置分组条件。例如，在 `bro_users` 表中按性别（`sex`）统计男生和女生两组的总记录数。如下所示：

```
D('users')->field('sex, count(sex)')->group('sex')->select(); //按性别分组
```

```
SQL: "SELECT sex, count(sex) FROM bro_users GROUP BY sex";
```

10. having()

该方法也不能单独使用，需要和 `select()` 方法及其他连贯操作的方法一起使用，用于设置分组后的筛选条件设置，必须和 `group()` 方法一起使用。例如，统计 `bro_users` 表中平均年龄大于 20 岁的男生和女生数量。如下所示：

```
D('users')->field('sex, count(sex)')->group('sex')->having('avg(age)>20')->select();
```

```
SQL: "SELECT sex, count(sex) FROM bro_users GROUP BY sex HAVING avg(age)>20";
```

11. total()

获取满足条件的记录总数，通常用于计算分页。可以和 `where()` 方法连贯操作设置条件，也可以直接在参数中传递查询条件。如果没有指定参数，则获取表中所有记录总数。例如，统计 `bro_users` 表年龄（`age`）大于 20 的数量。如下所示：

```
$count = D('users')->total(array("age >=>20)); //直接使用
```

```
$count = D('users')->where(array("age >=>20"))->total(); //和 where() 方法一起使用
```

```
SQL: "SELECT COUNT(*) as count FROM bro_users WHERE age > ?";
```


对应的数组绑定？参数 array(20);

12. select()

从一个数据表中获取满足条件的一条或多条记录，返回二维数组。具体的连贯操作参考前面的 field()、where()、order()、limit()、group()、having()等方法。例如，从表 bro_users 中获取主键值为 1,2,3 的三条记录。使用和组合后的 SQL 语句如下：

```
1 <?php
2 /**
3  file: user.class.php 在controls目录下，声明用户模块控制器
4  */
5 class User {
6     /* 控制器中的默认操作方法 */
7     function index() {
8         $user = D("user"); //创建用户实例对象
9
10        $data = $user -> field('id, name, age') //设置查询字段
11                    ->where('1, 2, 3') //设置查询条件
12                    ->order('id desc') //设置排序条件
13                    ->select(); //获取满足条件的记录
14
15        P( $data ); //打印二维数组
16    }
17 }
```

SQL: "SELECT id,name,age FROM bro_users WHERE id in(?,?,?) ORDER id desc";
对应的数组绑定？参数 array(1,2,3);

返回的二维数组\$data 的格式：
\$data=array(
 [0]=>Array("id"=>3, "name"=>"wangwu", "age"=>30),
 [1]=>Array("id"=>2, "name"=>"lisi", "age"=>20),
 [2]=>Array("id"=>1, "name"=>"zhangsan", "age"=>10)
);

13. r_select()

目前使用的数据库系统都是关联数据库系统，关联关系则是指表与表之间存在一定的关联关系（在一个表中使用外键保存另一个表的主键），通常我们所说的关联关系包括下面三种。

- (1) 一对一关联 (1:1)：一个用户一个购物车（用户表中一条记录和购物车中一条记录关系）。
- (2) 一对多关联 (1:n)：一个类别中有多篇文章（类别表中一条记录和文章表中多条记录关系）。
- (3) 多对多关联 (n:m)：一个班级有多个学生，一个学生上多个班级的课。

r_select()方法用于关联查询，可以按关联关系从多个表中获取记录。该方法和 select()一样可以通过连贯操作获取指定的记录。这个方法的参数需要传递一个或多个数组，每个数组关联一个数据表。例如，需要和其他两个数据表进行关联查询，则需要一起传递两个数组，每个参数的数组的结构都是一样的，数组中每个元素的作用说明如表 26-4 所示。

表 26-4 r_select()方法每个参数的结构说明

数组中的元素位置	描 述
第一个元素	需要关联的表的名称
第二个元素	关联表的字段列表，如果使用 1 对 1 关联数组的方式（没有提供第四个元素时），关联的数据表字段名和主表的字



	段名是不能相同的（如果相同，则从表和主表重名的字段将自动加上表名前缀 <code>user_name</code> , <code>user</code> 为表名, <code>name</code> 为重名字段），就需要在这个元素中，为和主表同名的字段起个别名。在这个参数中使用空字符串或 <code>null</code> ，则获取关联表的所有字段
第三个元素	关联的外键
第四个元素	<p>这个元素可以是一个数组或一个字段名称字符串，是可选的。如果没有提供这个参数，则是以 1 对 1 的表关系返回记录列表（右关联）。如果提供了第四个元素，是一个字段名称字符串，则是自己指定主表中某个字段需要和关系的表外键关联的键（也是以 1:1 的关联，但记录以主表为主，是左关联）；当设置这个参数为一个数组时，就会以子数组形式进行关联查询（适合 1 对多的表关系）。在这个数组中也有四个可用的元素，分别介绍如下：</p> <p>元素一：为子数组的下标</p> <p>元素二：是子数组记录的排序方式（可选）</p> <p>元素三：是限制子数组记录个数（可选）</p> <p>元素四：是子数组查询的 <code>where</code> 条件（可选）</p>

例如：有 `bro_cats`（类别表）、`bro_articles`（文章表）、`bro_test`（测试表）三个表，表结构和记录内容如下所示：

```
# bro_cats(类别表):
Create table bro_cats(
    id int not null auto_increment,
    name varchar(60) not null default '',
    desn text not null default '',
    primary key(id)
);
```

#表名为 `bro_cats`
#类别编号 ID
#类别名称
#类别描述

在表中插入 3 条记录，如下所示：

```
INSERT INTO bro_cats(name, desn) values('php','php demo');
INSERT INTO bro_cats(name, desn) values('jsp','jsp demo');
INSERT INTO bro_cats(name, desn) values('asp','asp demo');
```

```
# bro_articles(文章表):
Create table bro_articles(
    id int not null auto_increment,
    cid int not null default 0,
    name varchar(60) not null default '',
    content text not null default '',
    primary key(id)
);
```

#表名为 `bro_articles`
#类别编号 ID
#关联 `cats` 表的外键
#文章名称
#文章内容

在表中插入 5 条记录，如下所示：

```
INSERT INTO bro_articles(cid, name, content)
values(1,'this article of php1','php content1');
INSERT INTO bro_articles(cid, name, content)
values(1,'this article of php2','php content2');
INSERT INTO bro_articles(cid, name, content)
values(2,'this article of jsp','jsp content');
INSERT INTO bro_articles(cid, name, content)
values(3,'this article of asp1','asp content1');
INSERT INTO bro_articles(cid, name, content)
values(3,'this article of asp2','asp content2');
```

#php 类中 cid=1
#php 类中 cid=1
#jsp 类中 cid=2
#asp 类中 cid=3
#asp 类中 cid=3

bro_tests(测试表):

```
Create table bro_tests(
    id int not null auto_increment,
    cid int not null default 0,
    test varchar(60) not null default '',
    primary key(id)
);
```

#表名为 bro_users
#类别编号 ID
#关联 cats 表的外键
#测试字段

在表中插入 4 条记录，如下所示：

```
INSERT INTO bro_tests(cid, test) values(1,'php data'); #cid=1
INSERT INTO bro_tests(cid, test) values(2,'jsp data'); #cid=2
INSERT INTO bro_tests(cid, test) values(3,'asp data'); #cid=3
```

例如，使用 `r_select()` 方法从 `bro_cats` 和 `bro_articles` 两个表中获取类别名称、文章名称和文章内容。使用和组合后的 SQL 语句如下：

```
1 <?php
2 /**
3  file: cat.class.php    声明的类别模块控制器
4  */
5  class Cat {
6      /* 控制器中默认的操作方法 */
7      function index() {
8          $cat = D("cats");
9
10         //分别从cats和articles两个表中获取数据(右关联)
11         $data = $cat -> field('id, name as cname')          //主键id必取
12             -> r_select(
13                 //数组 关联表名          字段列表          外键
14                 array('articles', 'id, name, content', 'cid')
15             );
16
17         P( $data );      //打印二维数组
18     }
19 }
```

SQL: SELECT id,name as cname FROM bro_cats ORDER BY id ASC

SQL: SELECT id,name,content,cid FROM bro_articles WHERE cid IN('1','2','3') ORDER BY id ASC

返回的二维数组 `$data` 的格式：

```
$data=Array (
    [0] => Array(
        [id] => 1
        [cname] => php
        [name] => this article of php1
        [content] => php content1
        [cid] => 1
    )
    [1] => Array (
        [id] => 1
        [cname] => php
        [name] => this article of php2
        [content] => php content2
        [cid] => 1
    )
)
```



```

)
[2] => Array (
    [id] => 2
    [cname] => jsp
    [name] => this article of jsp
    [content] => jsp content
    [cid] => 2
)
[3] => Array (
    [id] => 3
    [cname] => asp
    [name] => this article of asp1
    [content] => asp content1
    [cid] => 3
)
[4] => Array (
    [id] => 3
    [cname] => asp
    [name] => this article of asp2
    [content] => asp content2
    [cid] => 3
)
)

```

例如，还是使用 `r_select()` 方法从 `bro_cats` 和 `bro_articles` 两个表中获取类别名称、文章名称和文章内容，但要求让 `bro_articles` 表中的记录以子数组的形式和 `bro_cats` 记录对应显示，这时，就需要在参数的数组中使用第 4 个元素，这个元素可以是一个数组（有 4 个可以用的元素）。使用和组合后的 SQL 语句如下：

```

1 <?php
2 /**
3      file: cat.class.php      声明的类别模块控制器
4  */
5  class Cat {
6      /* 控制器中默认的操作方法 */
7      function index() {
8          $cat = D("cats");
9
10         $data = $cat -> field('id, name')           //不需要别名
11         ->r_select(
12             array('articles', 'id, name, content', 'cid', array('art', 'id desc', 5))
13         );
14
15         P( $data );
16     }
17 }

```

SQL: SELECT id,name as cname FROM bro_cats ORDER BY id ASC

SQL: SELECT id,name,content,cid FROM bro_articles WHERE cid IN('1','2','3') ORDER BY id ASC

返回的二维数组 `$data` 的格式：

```

$data= Array (
    [0] => Array(
        [id] => 1
        [name] => php
        [art] => Array(
            //子数组下标 art

```

```

        [0] => Array(
            [id] => 2      //order by id desc
            [name] => this article of php2
            [content] => php content2
            [cid] => 1
        )

        [1] => Array(
            [id] => 1
            [name] => this article of php1
            [content] => php content1
            [cid] => 1
        )
    )
)

[1] => Array (
    [id] => 2
    [name] => jsp
    [art] => Array(
        [0] => Array(
            [id] => 3
            [name] => this article of jsp
            [content] => jsp content
            [cid] => 2
        )
    )
)

[2] => Array (
    [id] => 3
    [name] => asp
    [art] => Array (
        [0] => Array (
            [id] => 5
            [name] => this article of asp2
            [content] => asp content2
            [cid] => 3
        )

        [1] => Array (
            [id] => 4
            [name] => this article of asp1
            [content] => asp content1
            [cid] => 3
        )
    )
)
)

```

如果从三个关联的表中获取数据（加上 `bro_tests` 表,关联的外键都是 `cid`），只要在 `r_select()`方法中多传入一个数组即可（可以是更多个关联的表）。使用和组合后的 SQL 语句如下：



```
1 <?php
2 /**
3      file: cat.class.php      声明的类别模块控制器
4  */
5  class Cat {
6      /* 控制器中默认的操作方法 */
7      function index() {
8          $cat = D("cats");
9
10         $data = $cat -> field('id, name')           //不需要别名
11                     -> r_select(
12                         array('articles', 'id, name, content', 'cid', array('art', 'id desc')),
13                         array('tests', 'id, test', 'cid', array('test'))
14                     );
15
16         P( $data );
17     }
18 }
```

SQL: SELECT id,name as cname FROM bro_cats ORDER BY id ASC

SQL: SELECT id,name,content,cid FROM bro_articles WHERE cid IN('1','2','3') ORDER BY id ASC

更多的 r_select()应用可以参考下面的例子。下例是从 4 个表中获取关联数据，几乎用到了 r_select() 方法的全部语法。是在后面 BroCMS 项目中应用的一条语句，获取首页面的所有栏目信息，包括子栏目、栏目图片，以及栏目下的符合条件的文章。

```
1 <?php
2 //获取并分配所有栏目
3 $column -> field("id, title, picid")->order("ord asc")->where(array("pid"=>0, "display"=>1))
4         -> r_select(
5             array("image", 'name as imgname', 'id', 'picid'),
6             array("column", 'id, title', 'pid', array("subcol", 'ord asc', '4', "display=1")),
7             array("article", 'id, title', 'pid', array('art', 'id desc', 10, "audit=1"))
8         );
```

14. r_delete()

该方法用于关联删除，可以按关联关系从多个表中删除关联的数据记录。和关联查询相似，只要在这个方法的参数中传递一个或多个数组，每个数组对应一个关联的数据表。参数数组中有三个元素，第一个元素为关联的表名，第二个元素为关联的外键，第三个元素是可选的附加条件，也是一个数组格式，用于补上一个删除的附加条件，和 where()方法用法一样。该方法删除成功后，返回全部的影响行数。例如，表结构同上，删除类别表 bro_cats 中 id 为 1, 2 的两条记录，同时删除 bro_articles 和 bro_tests 中和类别对应的记录，并限制删除 bro_tests 表时 test 字段中必须包含有“php”的内容。使用和组合后的 SQL 语句如下：

```

1 <?php
2 /**
3  * file: cat.class.php 声明的类别模块控制器
4  */
5 class Cat {
6     /* 控制器中删除的操作方法 */
7     function del() {
8         $cat = D("cats");
9
10        $data = $cat -> where('1, 2')
11                    -> r-delete(
12                        array('articles', 'cid'),
13                        array('tests', 'cid', array('test'=>'%php%'))
14                    );
15
16        P( $data ); // 返回全部的影响行数
17    }
18 }

```

SQL: DELETE FROM bro_articles WHERE cid IN('1','2')

SQL: DELETE FROM bro_tests WHERE cid IN('1','2') AND test like 'php'

SQL: DELETE FROM bro_cats WHERE id IN('1','2')

15. query()

SQL 语句的统一入口，任何用户自定义的 SQL 语句（不能通过前面方法完成的 SQL 语句），都可以通过这个方法完成。该方法有三个参数：第一个参数就是用户自定义 SQL 语句，是必选项，可以使用“？”参数，如果使用问号参数，就必须在该方法的第三个参数中使用数组为“？”参数绑定对应的值。该方法的第二个参数是指定 SQL 语句的操作类型，返回什么类型由这个参数决定，第二个参数可以使用的字符串如下。

- “select”：查询多条记录的操作，返回二维数组
- “find”：查询一条记录的操作，返回一维数组
- “total”：按条件查询数据表的总记录数
- “insert”：插入数据的操作，返回最后插入的 ID
- “update”：更新数据表的操作，返回影响的行数
- “delete”：删除数据表的操作，返回影响的行数

如果第二个参数为空或其他字符串，query()方法执行成功则返回 true，失败则返回 false。在自定义的 SQL 语句中，表名可以直接使用数据库对象的 \$tabName 属性获取，例如：“\$user->tabName”获取用户表的表名。使用的方式如下：

```

1 <?php
2 /**
3  * file: user.class.php 定义一个用户控制器类User
4  */
5 class User {
6     /* 控制器中默认操作方法， 获取用户表记录的总数和全部记录 */
7     function index(){
8         $user = D("users");
9
10        $total = $user->query('SELECT [内容任意] FROM bro_users', 'total'); // 获取总数
11        $data = $user->query('SELECT * FROM bro_users', 'select'); // 全部记录

```



```
12
13     p($total, $data); //打印二维数组
14 }
15
16 /* 自定义insert语句，使用?参数，用最后一个数组参数绑定值，$user->tabName代表表名 */
17 function add(){
18     $user = D('users');
19     //返回最后插入的ID
20     $id = $user->query('insert into {$user->tabName}(name, age, sex) values(?,?,?)',
21         'insert',
22         array('zhangsan',10, '男'));
23     p($id);
24 }
25
26 /* 自定义删除SQL语句，删除age > 20 */
27 function del(){
28     $user = D('users');
29     //返回影响的行数
30     $num = $user->query('DELETE FROM {$user->tabName} WHERE age > ?', 'delete', array(20));
31     p($num)
32 }
33
34 /*自定义update语句，更新id=2的数据 */
35 function mod(){
36     $user = D('users');
37     //返回影响的行数
38     $num = $user->query('UPDATE {$user->tabName} set name=?,age=?,sex=? WHERE id=?',
39         'update',
40         array('zhangsan', 15, '女', 2));
41     p($num);
42 }
43
44 /* 自定义创建表hello语句，在第二个参数使用空字符串 */
45 function create(){
46     $user = D('users');
47     //成功返回true
48     $user->query('CREATE TABLE IF NOT EXISTS hello(id INT, name VARCHAR(30))','');
49 }
50 }
```

16. beginTransaction()

用于事务处理，开启一个事务。

17. commit()

用于事务处理，提交事务。

18. rollback()

用于事务处理，回滚事务。

19. dbSize()

获取项目中所有数据表的使用大小。

20. dbVersion()

获取数据库的版本信息。

21. setMsg()

设置 Model 类中的提示消息，有一个参数。参数的类型可以是一个字符串，也可以是一个数组。该函数设置的提示消息可以通过 getMsg()方法获取。

22. getMsg()

获取 Model 类中的提示消息。例如，验证成功或失败返回的提示消息。

26.8

自动验证

BroPHP 中的自动验证是基于 XML 方式实现的，可以对所有表单在服务器端通过 PHP 实现自动验证。如果自己定义一个 JS 文件，通过处理 XML 文件可以同时实现在前台也自动使用 JavaScript 验证。使用方法是在当前应用的 models 目录下，创建一个和表名同名的 XML 文件。例如，对 bro_users 表进行自动验证，则在 models 下创建一个 users.xml 文件（一般都是对入库的数据进行验证，而入库又发生在添加或修改数据时，所以 XML 文件名必须和表名相同才能自动处理）。文件中的使用样例如下所示：

```
/* 在 models 目录下，声明 users.xml，对添加或修改 bro_users 表的表单进行自动验证 */
<?xml version="1.0" encoding="utf-8"?>
<form>
  <input name="name" type="notnull" action="both" msg="有问题" />
  <input name="email" type="email" msg="不是正确的 EMAIL 格式" />
  <input name="price" type="currency" msg="价格必须是金钱格式" />
  <input name="code" type="vcode" msg="验证码输入错误!" />
  <input name="name" type="regex" value="/^abc/i" msg="不能匹配!" />
</form>
```

在上例的 XML 文件中，最外层标记<form>和每个子标记<input>，其实是可以任意命名的标记（上例的命名类似表单），如果不是正确的 XML 文件格式，也会在调试模式下提示（XML 文件每个标记必须有关闭，所有属性值都要使用双引号，第一行是固定写法）。但每个<input>标记中的属性名必须按规范设置，也可以对同一个表单进行多次不同形式的验证。例如，年龄不能为空和年龄必须是整数等，只要连续写两个<input name="age">标记即可。属性的设置分别介绍如下。

1. name 属性

该属性是必需的属性，和提交的表单项 name 属性是对应的，表示对那个表单项进行验证。

2. action 属性

该属性是可选的，用于设置验证的时间，可以有三个值 add（添加数据时进行验证）、mod（修改数据时进行验证）、both（添加和修改数据时都进行验证）。如果不加这个属性，默认值是 both。

3. msg 属性

该属性也是必须提供的属性，用于在验证没通过时的提示消息。

4. value 属性

该属性也是可选的，不过该属性是否使用和设置的值都由 type 属性的值决定。



5. type 属性

这是一个可选的属性，用于设置验证的形式，如果没有提供这个属性，默认值是“**regex**”（使用正则表达式进行验证，需要在 **value** 的属性中给出正则表达式）。该属性可以使用的值及使用方法如下所示。

regex：使用正则表达式进行验证，需要和 **value** 属性一起使用，在 **value** 中给出自定义的正则表达式，这也是默认的方式。例如：

```
<input name="name" type="regex" value="/^php/i" msg="名字不是以 PHP 开始的！" />
```

unique：唯一性校验，检查提交过来的值在数据表中是否已经存在，例如：

```
<input name="name" type="unique" msg="这个用户名已经存在！" />
```

notnull：验证表单提交的内容是否为空。例如，只在添加数据时验证：

```
<input name="name" type="notnull" action="add" msg="用户名不能为空！" />
```

email：验证是否是正确的电子邮件格式。例如：

```
<input name="email" type="email" msg="不是正确的 EMAIL 格式！" />
```

url：验证是否是正确的 URL 格式。例如：

```
<input name="url" type="url" msg="不是正解的 URL 格式！" />
```

number：验证是否是数字格式。例如：

```
<input name="age" type="number" msg="年龄必须输出数字！" />
```

currency：验证是否为金钱格式。例如：

```
<input name="price" type="currency" msg="商品价格的录入格式不正确！" />
```

confirm：检查两次输入的密码是否一致，需要使用 **value** 属性指定另一个表单（第一个密码字段）名称。例如：

```
<input name="repassword" type="confirm" value="password" msg="两次密码输入不一致！" />
```

in：检查值是否在指定范围之内，需要使用 **value** 属性指定范围，有多种用法。例如：

```
<input name="num" type="in" value="2" msg="输出的值必须是 2！" />
<input name="num" type="in" value="2-9" msg="输出的值必须在 2 和 9 之间！" />
<input name="num" type="in" value="1, 3, 5, 7" msg="必须是 1,3,5,7 中的一个！" />
```

Length：检查值的长度是否在指定的范围之内，需要使用 **value** 属性指定范围，例如：

```
<input name="username" type="length" value="3" msg="用户名的长度必须为 3 个字节！" />
<input name="username" type="length" value="3," msg="用户名的长度必须在 3 个以上！" />
<input name="username" type="length" value="3-" msg="用户名的长度必须在 3 个以上！" />
<input name="username" type="length" value="3,20" msg="用户名的长度必须在 3-20 之间！" />
<input name="username" type="length" value="3-20" msg="用户名的长度必须在 3-20 之间！" />
```


callback: 使用自定义的函数，通过回调的方式验证表单，需要通过 **value** 属性指定回调用函数的名称。例如，使用自定义的函数 **myfun** 验证用户名，如下所示：

```
<input name="name" type="callback" value="myfun" msg="名子不是以 PHP 开始！" />
```

在使用框架中的添加和修改方法时，必须使用第三个参数开启自动验证，例如“**insert(\$_POST, 1, 1)**”和“**update(null, 1, 1)**”第三个参数都使用一个“真”值。并使用 **DB** 对象中的 **getMsg()** 方法获取 XML 标中的 **msg** 属性值，提示用户定义的错误报告。在控制器中的简单应用如下所示：

```
1 <?php
2 /**
3  file: user.class.php  定义一个用户控制器类User
4  */
5  class User {
6      /* 处理用户从添加表单提交过来的数据，加入到数据表users中，并设置通过users.xml自动验证 */
7      function insert(0 {
8          $user = D("users");
9
10         if($user -> insert($_POST, 1, 1)) {
11             $this -> success("添加用户成功", 1, 'index'); //第三个参数"1"，开启XML验证
12         } else {
13             $this -> error($user->getMsg(), 3, 'add'); //使用getMsg()获取XML中的提示信息
14         }
15     }
16
17     /* 处理用户从修改表单提交过来的数据，修改数据表users一条记录，并设置通过users.xml自动验证 */
18     function update() {
19         $user = D("users");
20
21         if($user -> update(null, 1, 1)) {
22             $this -> success("修改用户成功", 2, 'index'); //第三个参数"1"，开启XML验证
23         } else {
24             $this -> error($user->getMsg(), 3, 'mod'); //使用getMsg()获取XML中的提示信息
25         }
26     }
27 }
```

另外，如果使用 BroPHP 中提供的 **Vcode** 类输出验证码，只要表单中输入验证的选项名称 **name** 值为“**code**”，并且 XML 文件存在，就会自动验证。

26.9 缓存设置

在 BroPHP 框架中提供了两种缓存机制，可以同时使用。一种是基于 **memcached** 将 **session** 会话数据和数据表的结果集缓存在服务器的内存中；另一种是使用 **Smarty** 的缓存机制实现页面静态化。建议在开发阶段不要开启任何缓存，上线运行一定要设置缓存。

26.9.1 基于 memcached 缓存设置

BroPHP 框架的 **memcached** 缓存设置比容易，只要 **memcached** 服务器安装成功（可以有多台），并



为 PHP 安装好了 memcached 的扩展应用。在配置文件 config.inc.php 中设置一个或多个 memcache 服务器地址和端口即可。BroPHP 框架就会自动将 session 信息和从数据库获取的结果集缓存到 memcached 中，如果用户执行了添加、修改或删除等影响表行数的操作，则会重新将数据表的结果数据缓存。配置文件中启用 memcached 如下所示：

```
//使用单一 memcached 服务器
$memServers = array("localhost", 11211);

//如果有多台 memcache 服务器可以使用二维数组
$memServers = array(
    array("www.lampbrother.net", '11211'),
    array("www.brophp.com", '11211'),
    ...
);
```

另外，在使用 BroPHP 框架开发的多个项目中，使用同一个 memcached 服务器时，实现了独立缓存，不会发生冲突。

26.9.2 基于 Smarty 的缓存机制

这种缓存设置和 Smarty 的使用方式是完全一样的，在 BroPHP 框架中也是通过配置文件 config.inc.php 去设置的缓存。

```
//在配置文件 config.inc.php 中开启 smarty 缓存设置
define("CSTART", 1); //缓存开关 1 开启，0 为关闭
define("CTIME", 60*60*24*7); //设置缓存时间
```

除开启了缓存设置以外，还需要在控制器类中进行一些设置，同 Smarty 的应用一样，如果开启页面缓存，就需要消除 PHP 和数据库间的处理开销。如下所示：

```
1 <?php
2 /**
3  file: user.class.php  定义一个用户控制器类User
4  */
5  class User {
6      /* 控制器中的默认操作方法 */
7      function index() {
8          /* 如果有对应的缓存文件则不再去连接数据库和执行SQL查询，使用缓存Smarty的isCached()方法判断 */
9          if( !$this -> isCached(null, $_SERVER["REQUEST_URI"]) ) {
10              //连接了数据库，读取表的数据
11              $user = D('users');
12              $this -> assign('data', $user -> select());
13          }
14
15          $this -> display(null, $_SERVER['REQUEST_URI']); //使用URI作为缓存ID
16      }
17  }
```

在 BroPHP 框架中，设置模板局部缓存，以及清除单个和多个缓存模板文件，也是直接采用 Smarty 的操作方式。

26.10

调试模式

调试模式是为程序员在开发阶段提供的帮助工具，在项目上线运行后将其关闭即可。关闭和开启调试模式非常简单，只要在配置文件 `config.inc.php` 中设置“DEBUG”选项的值即可（上线后使用 0 值关闭，开发时使用 1 值开启）。如果在线上运行后，关闭了调试模式，则会将运行中产生的错误报告写到 `runtime` 目录下的 `error_log` 文件中，这样在运行后也可以通过查看这个文件对项目进行维护。调试模式中可供参考的信息包括：脚本运行时间、自动包含的类、运行中的异常、一些常见的提示、使用的 SQL 语句和表结构等，可以通过关闭按钮临时关闭掉输出的提示框。调试框的界面如图 26-3 所示。



图 26-3 BroPHP 框架的调试信息界面

如果在开发阶段，某个操作中并不需要显示调试模式的界面，则可以在当前的操作中加上一个开关（使用函数 `debug(0)` 或 `debug()`，也可以使用 `$GLOBALS["debug"]=0`）就不会输出高调试信息的提示界面了。如下所示：

```

1 <?php
2 /**
3  * file: index.class.php 定义一个控制器类Index
4  */
5 class Index {
6     /* 控制器中默认的操作方法 */
7     function index() {
8         //关闭调试模式的输出，或使用$GLOBALS['debug']=0 也可以
9         debug( 0 );
10    }
11 }

```

另外，调试模式的开关也是一些缓存的开关。项目上线将调试模式关闭以后，一些程序中的缓存也将自动开启。例如，缓存数据表的结构（开发阶段表结构并不缓存，程序员反复修改的表结构时，都在项目中立即更新）、不再去判断一些目录或文件是否存在等，可以提高程序的运行效率。



26.11

内置扩展类库

在 BroPHP 框架中，内置了几个常用的扩展类，不需要任何改动直接就可以使用，包括文件上传、图像处理、分页和验证码四个类，并都在框架中自动进行了包含设置，直接实例化对象即可使用。如果需要更多这样的类库去使用，可以在项目目录下的 `classes` 目录中，自定义一些操作的类去使用。这四个常用类已经在前面章节中有过详细介绍，所以这里只是简单介绍一下如何在框架中应用。

26.11.1 分页类 Page

分页功能在每个项目中都是很常见的，框架中的分页类可以帮助你快速实现分页功能。不仅功能强大，使用也非常容易，对本类的操作只需要一些简单的属性和函数调用。虽然不需要在程序中包含分页类文件，但需要先创建分页类的对象再去应用。该类的构造方法中有 4 个参数：第一个参数是必需的，提供数据表需要显示的总记录数；第二个参数是可选的，提供每页需要显示的记录总数，默认为 25 条；

第三个参数也是可选的，用来向下一个页面提供本页中的数据；第四个参数也是可选的，用来设置默认页，需要一个布尔值，默认为 `true`，如果为 `true` 值，则默认显示第一页，如果使用 `false` 值，则默认页为最后一页。简单的应用使用的方式如下所示：

```
1 <?php
2 /**
3      file: user.class.php  定义一个用户控制器类User
4  */
5  class User {
6      /* 控制器中的默认操作方法，以分页形式显示所有用户 */
7      function index() {
8          $user = D("users");
9          //不需要加载分页类，直接创建分页对象，只使用前两个参数， 每页显示5条数据
10         $page = new Page($user->total(), 5);
11         //获取每页数据， 使用分页类中的$limit属性， 获取limit限制
12         $data = $user -> limit($page->limit) -> select();
13         //将数据分配给模板
14         $this -> assign('data', $data);
15         //分配分页内容给模板， 使用分页类中的fpage()方法获取分页内容
16         $this -> assign('fpage', $page -> fpage());
17         //显示输出模板
18         $this -> display();
19     }
20 }
```

输出结果如下所示：

共 33 条记录 本页 5 条 本页从 16-20 条 4/7页 [首页](#) [上一页](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [下一页](#) [末页](#) [4](#) [GO](#)

1. 设置分页输出内容显示格式

如果想自定义输出分页信息，也可以通过分页对象中的 `set()` 方法连贯操作进行设置，可以设置一个也可以单独或连续设置多个（设置的值也可以使用图片）。使用方式如下：

```
$page -> set("head", "条图片") //设置分页显示单位
```

```

-> set("first", "<|")           //修改“首页”按钮
-> set("last", ">|")           //修改“末页”按钮
-> set("prev", "<<|")         //修改“上一页”按钮
-> set("next", ">>|");         //修改“下一页”按钮

```

输出结果如下所示：

共 33 条图片 本页 5 条 本页从 16-20 条 4/7页 < << 1 2 3 4 5 6 7 >> > 4 GO

2. 设置分页输出内容及显示顺序

如果需要在输出结果中显示自定义内容，也可以通过 Page 类中的 fpage()方法的参数指定。在输出的结果中共由 8 部分组成，可以通过在 fpage()的参数中传入“0-7”之间的整数，自定义输出内容和输出的顺序。fpage()的方法参数使用如下所示：

```
$this->assign("fpage", $page->fpage(4,5,6,0,3));
```

输出结果如下所示：

< << 1 2 3 4 5 6 7 >> > 共 33 条图片 4/7页

3. 跳转页面添加附加资源

如果从当前页需要转到下一页时，将本页的一些数据也带到下一个页面中去，就可以在创建 Page 对象时，通过设置第三个参数完成。例如，当前是分类“cid=5”下面的数据分页，转到下页时也要是“cid=5”类别下的数据。创建分页对象如下所示（可以传更多的数据过去，只要使用 PATHINFO 的格式）：

```
$page=new Page($total, NUM, "cid/5");
```

4. 可以获取的属性

除了使用分页类中的一些方法，还可以从分页对象中获取两个属性的值：一个是分页时使用的 limit，另一个则是当前正在访问的页面。如下所示：

```

$page->limit;           //用于 SQL 语句中
$page->page;            //获取当前所在的分页页码

```

26.11.2 验证码类 Vcode

验证码也是项目中很常见的应用，用于限制“人”而非机器操作。BroPHP 将一些实现的细节封装到 Vcode 类中，只留了一个最简单的操作接口，实例化一个对象输出即可。该类的构造方法中有三个参数：第一个参数是验证码图片的宽度，默认值是 80 像素；第二个参数是验证码图片的高度，默认值是 20 像素；第三个参数是设置验证码的个数，默认值是 4 个。使用非常简单，只要在控制中声明一个方法，并在这个方法中创建对象后直接输出，然后在表单中使用的 src 指定这个操作方法即可输出验证码（注意：表单 name 名称为“code”）。如下所示：

```

1 <?php
2     /**
3      * file: user.class.php 定义一个用户控制器类User
4      */
5     class User {

```




```

6      /* 控制器中的操作 */
7      function code() {
8          //直接输出验证码对象，使用默认参数
9          echo new Vcode();
10         //或 new Vcode(100, 25, 5);    //使用参数设置验证码样式
11     }
12 }

```

在 HTML 表单中使用获取动态生成的验证码图片，src 属性 为“<{\$url}>/code”。并使用<input>标记将用户输入和图片一致的验证码传给服务器，其中<input>中的 name 属性值为“code”。如使用方式如下：

<input type="text" name="code">	{* name 属性值为"code"*
	{* src 的值请求当前模块的 code 操作 *

如果看不清，可以单击图片换一张，要让用户感觉不区分大小写，可以借助一些 JavaScript 代码来实现。如下所示：

```

{* 不区分大小写，输入的小写字母全部显示大写形式 *}
<input type="text" name="code" onkeyup="if (this.value != this.value.toUpperCase()) this.value = this.value.toUpperCase();" />
{* 如果图片看不清楚，单击切换一张新的图片 *}
/code/'+Math.random()" />

```

输出结果：

如果使用 BroPHP 自动验证，则只要对应的模板 XML 文件存在，就会自动检查验证码（输出表单名称必须为“code”，因为在服务器中是使用\$_SESSION["code"]保存的验证码，并且在自动验证中也是使用 code 值调用对应的验证函数）。

26.11.3 图像处理类 Image

在项目开发时经常需要对上传的图片内容进行优化，最常见的操作是对图片进行缩放、加水印及裁剪操作，本类提供了这三个功能。创建对象后调用 thumb()方法对图片进行缩放、调用 waterMark()方法可以为图片加水印，调用 cut()方法就可以对图片中的指定区域进行裁剪（目前支持 GIF、JPEG，PNG 等图片格式）。如下所示。

1. 构造方法

该方法用来创建图像处理类的对象，只有一个参数并且是可选的，用来指定处理图片的位置。默认处理图片的目录是与框架同级目录中的 public/uploads/目录下，可以通过这个唯一参数自定义图片所在的目录位置。

2. 图片缩放方法 thumb()

该方法用来对图像进行缩放，需要 4 个参数，其中最后一个参数是可选的，缩放成功后返回图片的名称，如果缩放失败，则返回 false。第一个参数是需要处理的图片名称（图片所在位置由构造方法决定）；第二个参数是图片需要缩放的宽度；第三个参数是图片需要缩放的高度；第四个参数是可选的，指定缩放后图片新名的前缀，默认值为“th_”。使用方式如下所示：

```
//例如，创建图像类对象后，将图片 brophp.gif 缩放至 300 x 300，并加上 th_前缀名
$img = new Image(); //创建图片对象
$simgname = $img->thumb("brophp.gif", 300, 300, "th_"); //缩放图片，返回缩放后的图片名
```

3. 为图片添加水印方法 waterMark()

该方法用来为图像添加水印（只支持图片水印），也需要 4 个参数，其中最后一个参数也是可选的，成功后返回加水印后新图片的名称，如果失败，则返回 `false`。第一个参数是背景图片，即需要加水印的图片（图片所在位置也由构造方法决定）；第二个参数是图片水印，即作为水印的图片（图片所在位置也由构造方法决定）；第三个参数是水印图片在背景图片上添加的位置，共有 10 种状态，0 为随机位置（1 为顶端居左，2 为顶端居中，3 为顶端居右，4 为中部居左，5 为中部居中，6 为中部居右，7 为底端居左，8 为底端居中，9 为底端居右）；第四个参数是可选的，指图片新名的前缀，默认值为“wa_”。使用方式如下所示：

```
//例如，创建图像类对象后，将图片 brophp.gif 加上水印 php.gif
$img = new Image(); //创建图片对象
$simgname = $img->waterMark("brophp.gif", "php.gif", 5, "wa_"); //加水印中部居中
```

4. 图片裁剪方法 cut()

该方法可以在一个大的背景图片中剪裁出指定区域的图片，需要 6 个参数，其中最后一个参数也是可选的，成功后返回裁剪后的图片的名称，如果失败则返回 `false`。第一个参数是需要剪切的背景图片；第二个参数是剪切图片左边开始的位置；第三个参数是剪切图片顶部开始的位置；第四个参数是图片剪裁的宽度；第五个参数是图片剪裁的高度；第六个参数是可选的，指图片新名的前缀，默认值为“cu_”。使用方式如下所示：

```
//例如，创建图像类对象后，将图片 brophp.gif 从 50x50 的位置开始剪裁出 100 × 100 大小的图片
$img = new Image(); //创建图片对象
$simgname = $img->cut("brophp.gif", 50, 50, 100, 100, "cu_"); //剪裁出指定区域的内容
```

26.11.4 文件上传类 FileUpload

基本上每个项目都有文件上传功能，为了可以简化用户的上传工作，本类支持单个文件上传，也支持多个文件上传。另外，如果上传的是图片，还可以直接进行缩放和加水印的操作。也可以设置文件上传的尺寸、上传文件的类型和文件名称等。如下所示：

```
1 <?php
2 /**
3  * file: user.class.php  定义一个用户控制器类User
4  */
5 class User {
6  /* 控制器中添加用户的操作方法， 需要添加用户头像 */
7  function add() {
8      $user = D('users');
9      //使用返回的图片名称追加到$_POST数组中， 随表单的其他元素一起插入到数据库当中
10     $_POST['picname'] = $this->upload();
11     //将用户信息添加到数据表users中
12     $user -> insert($_POST);
13 }
```



```
14
15  /* 文件上传方法，这个方法最好不要声明在控制器中，最好定义到Model类中去 */
16  private function upload() {
17      //可以通过参数指定上传位置，也可以通过set()方法设置
18      $up = new FileUpload();
19      //pic为上传表单的名称，通过upload()方法实现
20      if($up->upload('pic')) {
21          //返回上传后的文件名，通过getFileName()方法
22          return $up -> getFileName();
23      } else {
24          //如果上传失败提示出错原因，通过getErrMsg()方法
25          $this -> error($up -> getErrMsg(), 3, 'index');
26      }
27  }
28  }
```

在 FileUpload 类中有几个可以使用的方法：创建对象以后，通过 upload()方法上传文件，参数为<input type=“file” name=“pic”>的 name 值；如果上传成功，可以通过该对象中的 getFileName()方法获取上传的文件（默认为随机文件名，可以设置）；如果上传失败，也可以通过 getErrMsg()方法获取出错信息；还可以通过 set()方法进行连贯操作，限制上传文件的尺寸、类型和是否启用随机文件名，也可以通过 set()方法对上传的图片缩放和加水印。如下所示：

```
1 <?php
2 /* 文件上传方法，这个方法最好不要声明在控制器，最好定义到Model类中去 */
3 private function upload() {
4     //可以通过参数指定上传位置，也可通过set()方法设置
5     $up = new FileUpload();
6
7     //设置上传文件存方位置
8     $up -> set('path', '/usr/www/uploads')
9     //设置上传文件允许的大小，单位为字节
10    -> set('maxSize', 1000000)
11    //设置允许上传的文件类型
12    -> set('allowType', array('gif', 'jpg', 'png'))
13    //设置启有上传后随机文件名， true启用（默认）， false使用原文件名
14    -> set('israndname', true)
15    //设置上传图片的缩放大小， 还可以通过prefix指定新名前缀
16    -> set('thumb', array('width'=>300, 'height'=>200))
17    //设置为上传图片加水印， 也可以通过prefix指定新名前缀
18    -> set('watermark', array('water'=>'php.gif', 'position'=>5));
19
20    //pic为上传表单的名称， 通过upload()方法实现
21    if($up->upload('pic')) {
22        //返回上传后的文件名， 通过getFileName()方法
23        return $up -> getFileName();
24    } else {
25        //如果上传失败提示出错原因， 通过getErrMsg()方法
26        $this -> error($up -> getErrMsg(), 3, 'index');
27    }
28 }
```

上传多个文件和单个文件上传的方法一致，但 getFileName()方法返回一个数组，为上传成功的图片名称。如果上传失败，getErrMsg()方法也返回一个数组，是每个出错的信息。

26.12

自定义功能扩展

除了使用 BroPHP 框架内置的功能外，还可以为框架自定义一些扩展功能。框架中提供了两种扩展方式：如果是一个比较小的功能，可以仅定义函数，例如获取客户端的 IP 地址；而如果需要一些比较复杂的功能，就需要声明功能类放到框架中去使用。

26.12.1 自定义扩展类库

使用 BroPHP 框架除了自定义控制器类和业务模型类，还可以自定义一些扩展功能类。只要将类声明在 `classes` 目录下（以 `.class.php` 为后缀名，文件名全部小写），并以类名作为文件名，一个文件中存放一个类。如果按这些规范编写，则所有自定义的类都会被 BroPHP 框架用到时自动加载，在任何位置都可以直接创建对象并使用，包括通过类名直接调用的静态方法。

26.12.2 自定义扩展函数库

如果是一个很小的功能，就不需要通过编写类去实现，只要一个小函数就可以搞定。BroPHP 框架也提供了自定义函数的位置，只要将自定义的功能函数编写在 `commons` 目录下的 `functions.inc.php` 文件中，使全局函数在任何位置都可以直接调用。

26.13

小结

本章必须掌握的知识点

- BroPHP 框架对系统的要求
- 单一入口文件的作用与声明
- 项目的目录结构部署及应用
- BroPHP 框架的一些基本设置
- 控制器的声明与应用
- 视图的声明与应用
- 模型的声明与应用
- 应用缓存设置
- 内置扩展类的使用
- 自定义功能扩展



无兄弟，不编程

本章需要了解的内容

- BroPHP 框架的系统特性
- BroPHP 框架源码的目录结构

