

第20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。

20.1 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、电子商务、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。

20.1.1 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.lamp.brother.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。

20.1.2 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。

20.1.3 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

20.2 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。

20.2.1 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：

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

20.3 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。

20.3.1 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）

```
<?php
/**
    file: index.php      声明基于 BroPHP 框架开发的项目，单一入口文件示例
 */
define("APP", ".");           #定义项目的应用路径(' '可加可不加)
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）

```
<?php
/**
```

```

file: index.php      声明基于 BroPHP 框架开发的项目，单一入口文件示例
*/
define("APP", "./home/");          #定义项目的应用路径在 home 下(可加可不加)
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）

```

<?php
/**
file: index.php      声明基于 BroPHP 框架开发的项目，单一入口文件示例
*/
define("APP", ".");          #定义项目的应用路径(可加可不加)
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）

```

<?php
/**
file: index.php      声明基于 BroPHP 框架开发的项目，单一入口文件示例
*/
define("APP", "./admin/");    #定义项目的应用路径(可加可不加)
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`）。

20.3.2 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。

20.4 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。

20.4.1 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 20-1 所示。

表 20-1 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“ <code>config.inc.php</code> ”，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

续表

默认开启功能	描 述
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 <code>memcached</code> ，则将用户的会话信息写入到 <code>memcached</code> 服务器，否则使用默认的写入方式



20.4.2 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 20-2 所示。

表 20-2 BroPHP 框架配置文件的选项和描述

配置选项	描 述
<code>define("DEBUG", 1);</code>	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
<code>define("DRIVER", "pdo");</code>	设置数据库的驱动选项，本系统支持 <code>pdo</code> （默认）和 <code>mysqli</code> 两种驱动方式。在开启 <code>mysqli</code> 时需要 PHP 环境安装 <code>mysqli</code> 扩展模块，在使用 <code>PDO</code> 选项时，除了需要 PHP 环境中安装 <code>PDO</code> 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 <code>DRIVER</code> 选项设置为 <code>pdo</code> 时，则可开启这个 <code>PDO</code> 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 <code>HOST</code> 、 <code>USER</code> 、 <code>PASS</code> 和 <code>DBNAME</code> 选项
<code>define("HOST", "localhost");</code>	数据库系统的主机设置选项，默认为 <code>localhost</code>
<code>define("USER", "root");</code>	数据库系统用户名，默认为 <code>root</code>
<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 <code>brophp</code>
<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 <code>tpl</code>

续表

配置选项	描 述
<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 <code>default</code>
<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
----	--------------------------

20.4.3 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
```



20.5 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。**

20.5.1 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中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：

```
<?php
/**
    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
 */
Class User {
    //声明控制器的操作
}
```

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`，不能有其他的继承方式。如下所示：

```
<?php
/**
```

```

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
*/
Class User extends Action{
    //声明控制器的操作
}

```

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：

```

<?php
/**
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
*/
class Common extends Action {
    function init() {
        //所有的操作都会行执行这个方法
        //通常用于设置用户登录
    }

    //可以自定义一些方法， 作为所有控制器的公用操作方法
}

```

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。

20.5.2 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：

```

<?php
/** file: user.class.php 在 controls 目录下， 默认继承 Common 及 Action */
class User {
    /* 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 */
    function index() {
        // 这个方法的 URL 访问如下两种方式， 可以不写操作名 (index)
        // http://www.brophp.com/index.php/user/
        // http://www.brophp.com/index.php/user/index
    }

    /* 控制器中声明的方法，用于获取添加用户界面的操作*/
    function add() {
        // 这个操作的访问如下， 模块 (user)， 操作(add)
    }
}

```



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

/* 控制器中声明的方法，用于修改用户的操作*/
function mod() {
    // 这个操作的访问如下， 模块(user)， 操作(add)
    // http://www.brophp.com/index.php/user/mod/id/5
    p( $_GET["5"] );    //输出结果： Array( "id"=>5 );
}

/* 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 */
private function upload() {
    //这个用于上传用户头像， 不是操作则不能用 URL 访问
}
}
```

20.5.3 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：

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

成功的提示界面如图 20-1 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 20-2 所示。



图 20-1 success()方法成功提示界面



图 20-2 error()方法失败提示界面

20.5.4 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：

```
<?php
/** file: user.class.php 在 controls 目录下， 默认继承 Common 及 Action */
class User {
    /* 控制器中默认的方法，用于获取用户默认的操作 */
    function index() {
        //默认的操作方法
    }

    /* 控制器中声明的方法，用于删除用户的操作 */
    function del() {
        //创建用户对象
        $user = D("user");
        //使用用户对象中的 delete 删除某指定的一个用户
        if( $user->delete($_GET["id"]) ) {
            //如果删除成功就重定向到本模板的默认操作 index 中
            $this->redirect("index");
        } else {
            //如果删除失败就跳转到提示界面，并返回
            $this->error("删除用户失败");
        }
    }
}
```



redirect()方法的其他应用

```
$this->redirect("模块/动作");           #如果有使用 "/" 分成模块和操作
$this->redirect("book/add");           #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：

```
$this->redirect("模块/动作", "参数");     #使用第二个参数，传数据（PATHINFO 格式）
$this->redirect("book/index", "cid/5/page/3"); #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。

20.6 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）。

20.6.1 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：

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

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。

20.6.2 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。

20.6.3 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`。

20.6.4 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：



```
<?php
/** file: user.class.php 定义一个控制器类 User */
class User {
    /* 控制器中默认的方法 */
    function index() {
        /*
            如果没有提供参数，默认找和当前模块相同目录名(user)下的
            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）
            例如: view/default/user/index.tpl 模板文件
        */
        $this->display();

        /*
            如果提供参数没有"/"，默认找和当前模块相同目录名(user)下的
            模板文件名为参数名add，后缀名为tpl
            例如: view/default/user/add.tpl 模板文件
        */
        $this->display("add");

        /*
            如果提供参数有"/"，找和"/"前模块同名目录（shop）下的
            模板文件名为参数名add，后缀名为tpl（配置文件可改）
            例如: view/default/shop/add.tpl 模板文件
        */
        $this->display("shop/add");
    }
}
```

20.6.5 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
<{$url}>; //到访问的模块 /shop/admin.php/index
<{$public}>; //所有应用的共用资源 public /shop/public
<{$res}>; //到模板风格下的 resource 目录 /shop/views/default/resource
```

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>
```

20.6.6 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。

20.7 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。

20.7.1 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。

20.7.2 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。

20.7.3 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：

```
<?php
/**
 *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
 *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
 */
class Users {
    /* 声明一个用户登录的方法 */
    function isLogin() {
        //方法体
    }

    /* 声明一个退出系统的方法 */
    function isLogout() {
        //方法体
    }
}
```

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：

```
<?php
/**
 * 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
 */
class Demo {
    //声明一个功能方法
    function fun() {
        //多个子类可以共用这个方法
    }
}
```

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

```
<?php
/**
 * file: books.class.php 声明一个类 Books 主动继承 Demo 类，间接继承了 DB 类
 */
class Books extends Demo {
    //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
}
```

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：

```
<?php
/**
 * 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
 */
class User {
    /* 控制器中默认的方法，用于获取用户默认的操作 */
    function index() {
        //创建用户对象，参数 user 找模型中的 User 类创建
        $user = D("user");
        $data = $user->select(); //调用父类中的方法查询表中所有记录
        // $user -> isLogin(); //也可以调用 User 类中声明的方法
    }
}
```

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：

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

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：

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

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：

```
<?php
/**
 * file: index.class.php 在前台 controls 目录下声明的主控制器
 */
class Index {
    /* 控制器中默认的方法 */
```



```
function index() {  
    //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  
    $db = D(); //可以访问 DB 对象中非表的操作方法  
  
    $db -> dbSize(); //获取数据库的空间使用信息  
    $db -> dbVersion(); //获取数据库系统的版本  
    $db -> beginTransaction(); //开启事务  
    $db -> commit(); //提交事务  
    $db -> rollback(); //回滚事务  
}
```

20.7.4 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 20-3 所示。

表 20-3 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()	设置提示消息，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",
    "age"=>"22",
    "sex"=>"男",
    "email"=>"gaolf@php.net",
    "sub"=>"注册"
);
```

#<input name="name">
#<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：

```
<?php
/**
 *file: user.class.php 在 controls 目录下，声明用户模块控制器
 */
class User {
    /* 控制器中添加方法 */
    function add() {
        $user = D("user"); //创建用户实例对象

        $_POST["ptime"] = time(); //向$_POST 数组中添加用户注册时间
        $id = $user->insert(); //默认使用$_POST 数组作为参数
    }
}
```




```

        // $id = $user -> insert($_POST); //也可以直接传递$_POST 参数
    }
}

```

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：

```
<?php
/**
    file: user.class.php 在 controls 目录下，声明用户模块控制器
 */
class User {
    /* 控制器中修改方法 */
    function mod() {
        $user = D("user");           //创建用户实例对象

        $rows = $user -> update();    //默认使用$_POST 数组作为参数
        //$rows = $user -> update($_POST); //也可以直接传递$_POST 参数
    }
}
```

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：

```
<?php
/**
    file: user.class.php 在 controls 目录下，声明用户模块控制器
 */
class User {
    /* 控制器中修改方法 */
    function mod() {
        $user = D("user");           //创建用户实例对象

        //默认使用$_POST 数组作为参数(可以默认)，加上 where()连贯操作
        $rows = $user -> where("1, 2, 3") -> update($_POST);
    }
}
```

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：



无兄弟，不编程

```
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：

```
<?php
/**
    file: user.class.php 在 controls 目录下，声明用户模块控制器
 */
class User {
    /* 控制器中删除的方法 */
    function del() {
        $user = D("user"); //创建用户实例对象

        /*使用$_GET 数组传过来的主键作为参数删除一条， 例如$_GET['id'] = 5;
        $rows = $user -> delete( $_GET['id'] );
        */
    }
}
```

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：

```
<?php
/**
    file: user.class.php 在 controls 目录下，声明用户模块控制器
 */
class User {
    /* 控制器中删除的方法 */
    function del() {
        $user = D("user"); //创建用户实例对象

        /*
            $_POST 数组中是传过来的多个主键（复选框中选中id 为前5 条）
            例如： $_POST = array("id" => array(1, 2, 3, 4, 5));
        */
        $rows = $user -> delete( $_POST['id'] );
    }
}
```

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：

```
<?php
/**
 * file: user.class.php 在 controls 目录下，声明用户模块控制器
 */
class User {
    /* 控制器中修改用户的方法 */
    function mod() {
        $user = D("user");           //创建用户实例对象

        $data = $user->find( $_GET['id'] );   //$_GET['id'] = 5
        p( $data );                     //打印结果数组
    }
}
```

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:

```
<?php
/**
 *file: user.class.php 在 controls 目录下, 声明用户模块控制器
 */
class User {
    /* 控制器中的默认操作方法 */
    function index() {
        $user = D("user"); //创建用户实例对象

        $data = $user -> field('id, name, age') //设置查询字段
```




```
->where('1, 2, 3')           //设置查询条件
->order('id desc')           //设置排序条件
->select();                   //获取满足条件的记录

P( $data );                   //打印二维数组
}
}
```

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 20-4 所示。

表 20-4 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 user_name, user 为表名，name 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 null，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	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： 元素一：为子数组的下标

续表

数组中的元素位置	描 述
第四个元素	元素二：是子数组记录的排序方式（可选） 元素三：是限制子数组记录个数（可选） 元素四：是子数组查询的 where 条件（可选）

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(                                #表名为 bro_cats
    id int not null auto_increment,                    #类别编号 ID
    name varchar(60) not null default '',              #类别名称
    desn text not null default '',                     #类别描述
    primary key(id)
);
```

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(                             #表名为 bro_articles
    id int not null auto_increment,                    #类别编号 ID
    cid int not null default 0,                        #关联 cats 表的外键
    name varchar(60) not null default '',              #文章名称
    content text not null default '',                  #文章内容
    primary key(id)
);
```

在表中插入 5 条记录，如下所示：

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

```
# bro_tests(测试表):
Create table bro_tests(                                #表名为 bro_users
    id int not null auto_increment,                    #类别编号 ID
    cid int not null default 0,                        #关联 cats 表的外键
    test varchar(60) not null default '',              #测试字段
    primary key(id)
);
```



无兄弟，不编程

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：

```
<?php
/**
    file: cat.class.php    声明的类别模块控制器
*/
class Cat {
    /* 控制器中默认的操作方法 */
    function index() {
        $cat = D("cats");

        //分别从 cats 和 articles 两个表中获取数据(右关联)
        $data = $cat -> field('id, name as cname')           //主键 id 必取
                        ->r_select(
                            //数组 关联表名 字段列表      外键
                            array('articles', 'id, name, content', 'cid')
                        );

        P( $data );      //打印二维数组
    }
}
```

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：

```

<?php
/**
    file: cat.class.php 声明的类别模块控制器
*/
class Cat {
    /* 控制器中默认的操作方法 */
    function index() {
        $cat = D("cats");

        $data = $cat -> field('id, name') //不需要别名
            ->r_select(
                array('articles', 'id, name, content', 'cid', array('art', 'id desc', 5))
            );

        P( $data );
    }
}

```

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(

```



无兄弟，不编程

```
[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：

```
<?php
/**
    file: cat.class.php    声明的类别模块控制器
*/
class Cat {
    /* 控制器中默认的操作方法 */
    function index() {
        $cat = D("cats");

        $data = $cat -> field('id, name')    //不需要别名
```

```

        ->r_select(
            array('articles', 'id, name, content', 'cid', array('art', 'id desc')),
            array('tests', 'id, test', 'cid', array('test')))
        );

        P( $data );
    }
}

```

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。

```

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

```

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：

```

<?php
/**
 *file: cat.class.php  声明的类别模块控制器
 */
class Cat {
    /* 控制器中删除的操作方法 */
    function del() {
        $cat = D("cats");

        $data = $cat -> where('1, 2')
            -> r_delete(
                array('articles', 'cid'),
                array('tests', 'cid', array('test'=> '%php%'))
            );

        P( $data );    //返回全部的影响行数
    }
}

```



```
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：

```
<?php
/**
 *file: user.class.php 定义一个用户控制器类 User
 */
class User {
    /* 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 */
    function index(){
        $user = D("users");

        $total = $user->query('SELECT [内容任意] FROM bro_users', 'total'); //获取总数
        $data = $user->query('SELECT * FROM bro_users','select'); //全部记录

        P($total, $data); //打印二维数组
    }

    /* 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 */
    function add(){
        $user = D('users');
        //返回最后插入的ID
        $id = $user->query('insert into {$user->tabName}(name, age, sex) values(?,?,?)',
            'insert',
            array('zhangsan',10, '男'));

        p($id);
    }

    /* 自定义删除 SQL 语句， 删除 age > 20 */
    function del(){
        $user = D('users');
        //返回影响的行数
    }
}
```

```

$num = $user->query('DELETE FROM {$user->tabName} WHERE age > ?', 'delete',
array(20));
    p($num)
}

/*自定义 update 语句, 更新 id=2 的数据 */
function mod(){
    $user = D('users');
    //返回影响的行数
    $num = $user->query('UPDATE {$user->tabName} set name=?,age=?,sex=? WHERE id=?',
                        'update',
                        array('zhangsan', 15, '女', 2));
    p($num);
}

/* 自定义创建表 hello 语句, 在第二个参数使用空字符串 */
function create(){
    $user = D('users');
    //成功返回 true
    $user->query('CREATE TABLE IF NOT EXISTS hello(id INT, name VARCHAR(30))');
}
}

```

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。

20.8 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="名子不是以 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：

```
<?php
/**
 * file: user.class.php 定义一个用户控制器类 User
 */
class User {
    /* 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 */
    function insert() {
        $user = D("users");

        if($user -> insert($_POST, 1, 1)) {
            $this -> success("添加用户成功", 1, 'index'); //第三个参数 "1"，开启XML验证
        } else {
            $this -> error($user->getMsg(), 3, 'add'); //使用 getMsg() 获取 XML 中的提示信息
        }
    }

    /* 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 */
    function update() {
        $user = D("users");

        if($user -> update(null, 1, 1)) {
            $this -> success("修改用户成功", 2, 'index'); //第三个参数 "1"，开启XML验证
        } else {
            $this -> error($user->getMsg(), 3, 'mod'); //使用 getMsg() 获取 XML 中的提示信息
        }
    }
}
```

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。

20.9 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。

20.9.1 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。

20.9.2 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：

```
<?php
/**
 * file: user.class.php 定义一个用户控制器类 User
 */
class User {
    /* 控制器中的默认操作方法 */
    function index() {
        /* 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 */
        if( !$this->isCached(null, $_SERVER['REQUEST_URI']) ) {
            //连接了数据库，读取表的数据
            $user = D('users');
            $this->assign('data', $user->select());
        }

        $this->display(null, $_SERVER['REQUEST_URI']); //使用 URI 作为缓存 ID
    }
}
```

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。

20.10 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 20-3 所示。



图 20-3 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：

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

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。

20.11 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。

20.11.1 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：

```
<?php
/**
 *file: user.class.php 定义一个用户控制器类 User
 */
class User {
    /* 控制器中的默认操作方法, 以分页形式显示所有用户 */
    function index() {
        $user = D("users");
        //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
        $page = new Page($user->total(), 5);
        //获取每页数据, 使用分页类中的 $limit 属性, 获取 limit 限制
        $data = $user -> limit($page->limit) -> select();
        //将数据分配给模板
        $this -> assign('data', $data);
        //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
        $this -> assign('fpage', $page -> fpage());
        //显示输出模板
        $this -> display();
    }
}
```

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：



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
```

20.11.2 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：

```
<?php
/**
 * file: user.class.php 定义一个用户控制器类 User
 */
class User {
    /* 控制器中的操作 */
    function code() {
        //直接输出验证码对象，使用默认参数
        echo new Vcode();
        //或 new Vcode(100, 25, 5);    //使用参数设置验证码样式
    }
}
```

```
}
```

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）。

20.11.3 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
$imgname = $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

```
$imgname = $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

```
$imgname = $img->cut("brophp.gif", 50, 50, 100, 100, "cu_");
```

//剪裁出指定区域的内容

20.11.4 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：

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

    /* 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 */
    private function upload() {
        //可以通过参数指定上传位置， 也可以通过 set()方法设置
```

```

$up = new FileUpload();
//pic 为上传表单的名称， 通过 upload() 方法实现
if($up->upload('pic')) {
    //返回上传后的文件名， 通过 getFileName() 方法
    return $up -> getFileName();
} else {
    //如果上传失败提示出错原因， 通过 getErrorMsg() 方法
    $this -> error($up -> getErrorMsg(), 3, 'index');
}
}
}

```

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

```

<?php
/* 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 */
private function upload() {
    //可以通过参数指定上传位置，也可通过 set() 方法设置
    $up = new FileUpload();

    //设置上传文件存方位置
    $up -> set('path', '/usr/www/uploads')
    //设置上传文件允许的大小，单位为字节
    -> set('maxSize', 1000000)
    //设置允许上传的文件类型
    -> set('allowType', array('gif', 'jpg', 'png'))
    //设置启有上传后随机文件名， true 启用（默认）， false 使用原文件名
    -> set('israndname', true)
    //设置上传图片的缩放大小， 还可以通过 prefix 指定新名前缀
    -> set('thumb', array('width'=>300, 'height'=>200))
    //设置为上传图片加水印， 也可以通过 prefix 指定新名前缀
    -> set('watermark', array('water'=>'php.gif', 'position'=>5));

    //pic 为上传表单的名称， 通过 upload() 方法实现
    if($up->upload('pic')) {
        //返回上传后的文件名， 通过 getFileName() 方法
        return $up -> getFileName();
    } else {
        //如果上传失败提示出错原因， 通过 getErrorMsg() 方法
        $this -> error($up -> getErrorMsg(), 3, 'index');
    }
}
}

```

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，getErrorMsg() 方法也返回一个数组，是每个出错的信息。



20.12 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。

20.12.1 自定义扩展类库

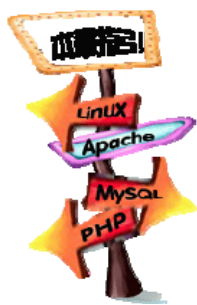
使用 BroPHP 框架除了自定义控制器类和业务模型类，还可以自定义一些扩展功能类。只要将类声明在 `classes` 目录下（以 `.class.php` 为后缀名，文件名全部小写），并以类名作为文件名，一个文件中存放一个类。如果按这些规范编写，则所有自定义的类都会被 BroPHP 框架用到时自动加载，在任何位置都可以直接创建对象并使用，包括通过类名直接调用的静态方法。

20.12.2 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。

第21章

B/S 结构软件开发流程



软件开发流程即软件设计思路和方法的一般过程，包括设计软件的功能和实现的算法和方法、软件的总体结构设计和数据库设计和模块设计、编程和调试、程序联调和测试及编写、提交程序。提高软件开发能力没有捷径可走，唯有走“规范化”之路。即制定适合于本企业的软件过程规范，并按照此规范执行。本规定对软件开发各个过程中的目的、要求、人员和职责、工作的内容及输入/输出、评审等进行规范。本规定主要的约束对象是 B/S 结构的应用软件开发，涉及的开发部仅指软件开发部，产品仅指狭义范围内的 B/S 结构程序或应用软件程序。

21.1

软件开发过程的划分

本规定对一个完整的开发过程按“软件过程改进方法和规范”把产品生命周期划分为 6 个阶段：包括产品概念阶段（记为 PH0）；产品定义阶段（记为 PH1）；产品开发阶段（记为 PH2）；产品测试阶段（记为 PH3）；用户验收阶段（记为 PH4）；产品维护阶段（记为 PH5）。软件项目的过程有三大类：项目管理过程、项目研发过程和机构支持过程。而这三类过程可以细分为 19 个主要过程域，分布在 PH0 到 PH5 的各个阶段。项目管理过程包含 6 个过程域，分别为：立项管理、结项管理、项目规划、项目监控、风险管理、需求管理。项目研发过程包含 8 个过程域，分别为：需求开发、技术预研、系统设计、实现与测试、系统测试、Beta 测试、客户验收、技术评审。机构支撑过程包含 5 个过程域，分别为：配置管理、质量保证、培训管理、外包与采购管理、服务与维护。建议用户（企业）根据自身情况（如发展战略、研发实力等）适当地修改使用。详细的划分如图 21-1 所示。

按照软件开发的流程规范，一个项目从策划到完成是由众多的过程规范和文档模板组成的，如表 21-1 所示。主要用于指导国内互联网企业持续地改进其软件过程能力。在项目经理领导的团队开发小组，在各种过程开发基础上，接受迭代开发和敏捷软件开发过程的思想，要不拘泥于传统的开发过程，建立公司的快速开发过程，随需而变，及时满足客户需求的变更。

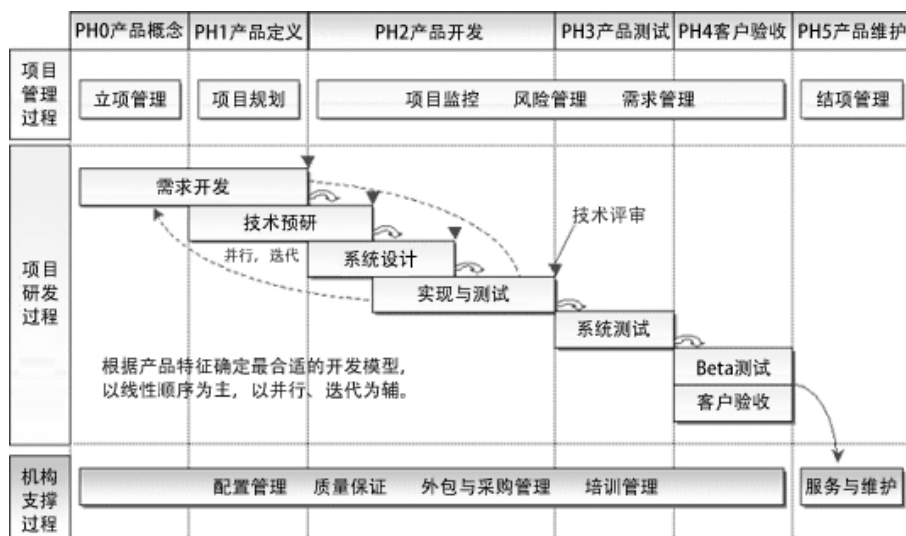


图 21-1 软件开发过程划分流程图

表 21-1 软件开发过程域遵循的标准文档

需求开发	《用户需求说明书》 《软件需求规格说明书》	技术预研	《技术预研计划》 《技术预研报告》
系统设计	《体系结构设计报告》 《用户界面设计报告》 《数据库设计报告》 《模块设计报告》	实现与测试	《实现与测试计划》 《编程文档》
系统测试	《系统测试计划》 《测试用例》 《测试报告》	Beta 测试	《Beta 测试协议》 《Beta 测试报告》
客户验收	《客户验收计划》 《客户验收报告》	技术评审	《技术评审计划》 《技术评审报告》 《技术评审检查表》

由于软件开发过程域遵循的标准文档众多，本书在后面几个章节中只提供了最常用的三个文档说明，供广大读者学习参考使用，包括《项目需求说明书》、《数据库设计说明书》及《程序设计说明书》。因为这三个文档和我们的项目设计有直接联系，也是项目开发流程中最主要的说明文档，其他的文档则需要在使用时自行整理。

21.2 需求开发

需求分析说明书的形成需要市场调研，技术和市场要结合才能体现最大价值。这个阶段需要出三样东西，用户视图、数据词典和用户操作手册。用户视图是该软件用户（包括终端用户和管理用户）所能看到的页面样式，其中包含了很多操作方面的流程和条件。数

据词典是指明数据逻辑关系并加以整理的东西，完成了数据词典，数据库的设计就完成了一半多。用户操作手册是指明了操作流程的说明书。请注意，用户操作流程和用户视图是由需求决定的，因此应该在软件设计之前完成，完成这些，就为程序研发提供了约束和准绳。很遗憾，太多公司都不是这样做的，因果颠倒，顺序不分，开发工作和实际需求往往因此产生隔阂脱节的现象。需求分析中，除了以上工作，作为项目设计者，应当完整地做出项目的性能需求说明书，因为往往性能需求只有懂技术的人才可能理解，这就需要技术专家和需求方（客户或公司市场部门）能够有真正的沟通和了解。

21.2.1 需求分析流程

需求分析流程图如图 21-2 所示。

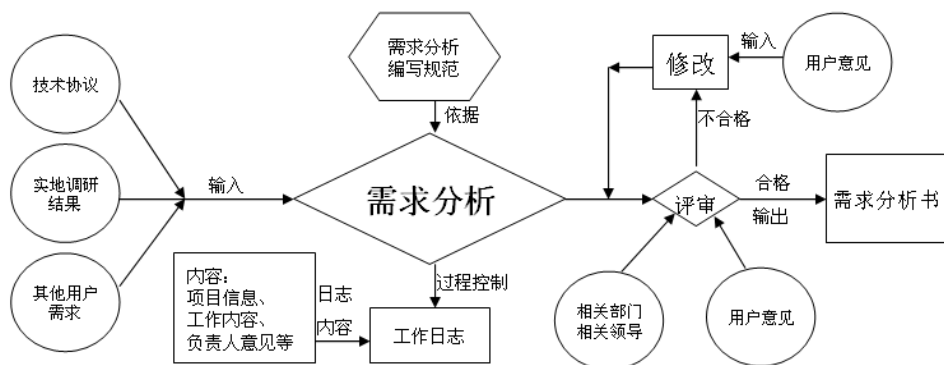


图 21-2 需求分析流程图

工作流程：市场部签订软件开发合同后，向开发部移交与之相关的资料，如合同书、技术协议等；开发部组织人员根据相关资料进行需求分析，并且要与用户进行技术交流，充分获取用户对软件开发的边界等具体问题的确认；需求分析编制完成后，经相关部门评审合格后即付诸实施。

责任部门：开发部。

相关部门：市场部、主管副总、用户。

相关资料：软件合同、技术协议、需求分析书、用户确认单、评审记录、日志。

相关规范：系统总体设计编制规范、系统详细设计编制规范。

21.3 系统设计

系统设计主要分为数据库设计、程序的概要设计及详细设计。概要设计是将系统功能模块进行初步划分，并给出合理的研发流程和资源要求。作为快速原型设计方法，完成概要设计就可以进入编码阶段了，通常采用这种方法是因为涉及的研发任务属于新领域，技术主管人员一上来无法给出明确的详细设计说明书，但是并不是说详细设计说明书不重



要。事实上快速原型法在完成原型代码后，根据评测结果和经验教训的总结，还要重新进行详细设计的步骤。详细设计则是考验技术专家设计思维的重要关卡，详细设计说明书应当把具体的模块以最“干净”的方式（黑箱结构）提供给编码者，使得系统整体模块化达到最大；一份好的详细设计说明书，可以使编码的复杂性降到最低，实际上，严格讲，详细设计说明书应当把每个函数的每个参数的定义都精精细细地提供出来，从需求分析到概要设计到完成详细设计说明书，一个软件项目就应当说完成了一半了。换言之，一个大型软件系统在完成了一半的时候，其实还没有开始一行代码工作。那些把做软件的程序员简单理解为写代码的，从根子上就错了。

21.3.1 系统设计流程

系统设计流程图如图 21-3 所示。

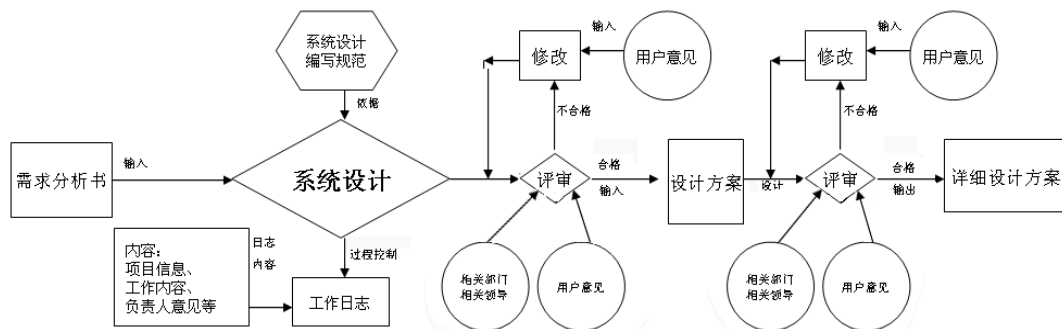


图 21-3 系统设计流程图

工作流程：需求分析经评审通过后，开发部组织人员进行系统设计；系统设计完成后，开发部组织相关专业部门进行评审并获得用户的确认。系统设计和系统详细设计均属于评审范围。

责任部门：开发部。

相关部门：市场部、主管副总、用户。

相关资料：需求分析书、系统总体设计规范、系统详细设计规范、数据字典、用户确认单、数据流定义、编码规范、日志。

相关规范：系统总体设计编制规范、系统详细设计编制规范。

21.4 编码测试

在规范化的研发流程中，编码工作在整个项目流程中最多不会超过 1/2，通常在 1/3 左右的时间。所谓“磨刀不误砍柴功”，设计过程完成得好，编码效率就会极大提高，编码时不同模块之间的进度协调和协作是最需要小心的，也许一个小模块的问题就可能影响整体进度，让很多程序员因此被迫停下工作等待，这种问题在很多研发过程中都出现过。

编码时的相互沟通和应急的解决手段都是相当重要的，对于程序员而言“bug”永远存在。按照测试执行方，可以分为内部测试和外部测试；按照测试范围，可以分为模块测试和整体联调；按照测试条件，可以分为正常操作情况测试和异常情况测试；按照测试的输入范围，可以分为全覆盖测试和抽样测试。总之，测试同样是项目研发中一个相当重要的步骤，对于一个大型软件，3 个月到 1 年的外部测试都是正常的，因为永远都会有不可预料的问题存在。完成测试后，完成验收并完成最后的一些帮助文档，整体项目才算告一段落，当然日后少不了升级、修补等工作，只要不是想通过一锤子买卖骗钱，就要不停地跟踪软件的运营状况并持续修补升级，直到这个软件被彻底淘汰为止。

21.4.1 编码与测试流程

编码与测试流程图如图 21-4 所示。

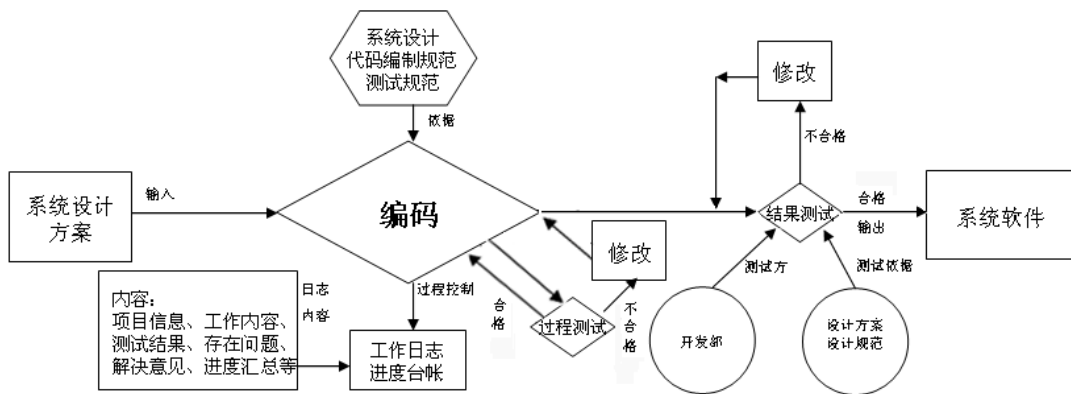


图 21-4 编码与测试流程图

工作流程：系统设计完成并经评审通过后，开发部组织人员进行代码编制（如采用外包方式编码，开发部要组织专门人员为外包单位提供代码编制规范和技术文档要求，并负责监控整个编码过程）。编码过程中，开发部相关人员应对完成后的每一模块组织进行过程测试；编码完成后，开发部组织相关人员对系统进行测试。测试分符合性测试和功能性测试两步进行，测试完成后，开发部组织相关专业部门对系统进行整体测评。

责任部门：开发部。

相关部门：主管副总、代码编制部门（外包）。

相关资料：系统详细设计、数据字典、编程记录；测试记录、测试报告、数据流定义、编码规范、代码描述、程序源代码及相关文档。

相关规范：软件设计代码编制规范、软件测试标准。

21.5 试运行

软件试运行是将产品交给最终用户，由其按实际业务流程对产品进行使用，一般试用



三个月或更长时间没问题后即可正常投产，试运行过程中，一般要进行双线运行，即同时按产品上线试运行前的操作和在试运行的软件中操作。

21.5.1 软件试运行流程

软件试运行流程图如图 21-5 所示。

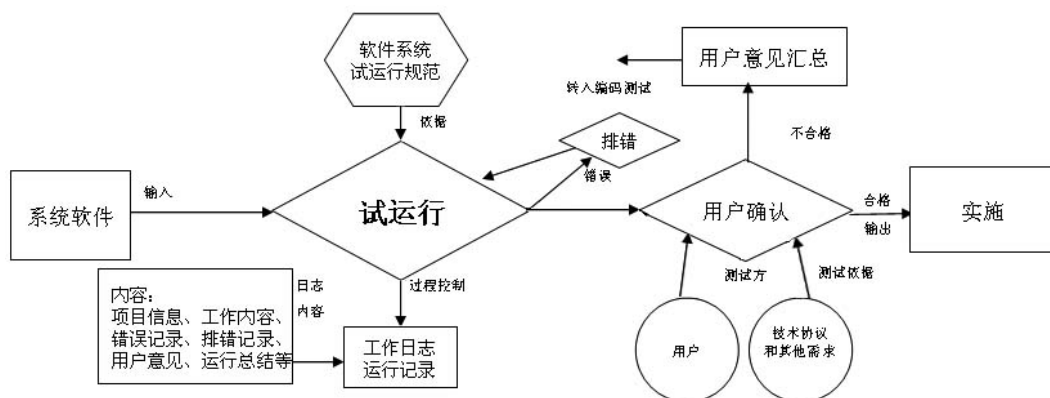


图 21-5 软件试运行流程图

工作流程：编码测试完成后，经相关部门同意，开发部组织系统试运行，试运行过程中要对系统所产生的问题进行详细记录并马上解决。

责任部门：开发部。

相关部门：用户、主管副总、代码编制部门（外包）。

相关资料：试运行记录、错误和排错记录、试运行总结报告。

相关规范：软件系统试运行规范、技术协议。

21.6 实施

软件项目实施方案概述软件产品，特别是行业解决方案软件产品不同于一般的商品，用户购买软件产品之后，不能立即进行使用，需要软件公司的技术人员在软件技术、软件功能、软件操作等方面进行系统调试、软件功能实现、人员培训、软件上线使用、后期维护等一系列的工作，我们将这一系列的工作称为软件项目实施。大量的软件公司项目实施案例证明，软件项目是否成功、用户的软件使用情况是否顺利、是否提高了用户的工作效率和管理水平，不仅取决于软件产品本身的质量，软件项目实施的质量效果也对后期用户应用的情况起到非常重要的影响。

21.6.1 软件实施流程

软件实施流程如图 21-6 所示。

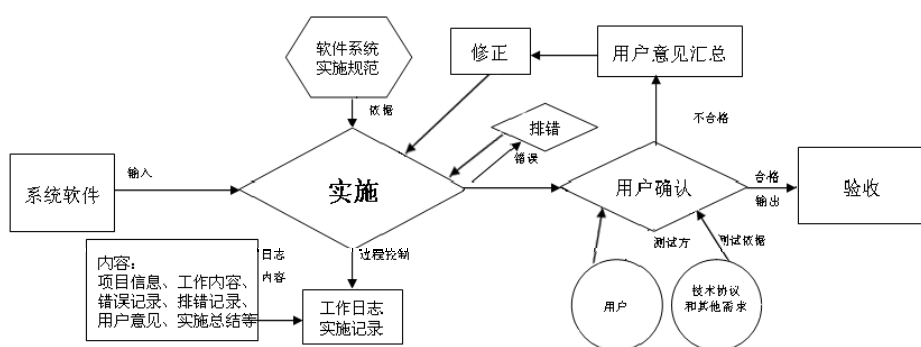


图 21-6 软件实施流程图

工作流程：试运行完成后，由开发部组织软件的实施（如由外包单位实施，开发部应该负责整个实施过程的监控、管理和协调）。

责任部门：开发部。

相关部门：用户、主管副总、代码编制部门（外包）。

相关资料：实施记录、用户意见表、用户意见反馈表、系统实施总结报告。

相关规范：软件系统实施规范、技术协议。

21.7 验收

在软件开发合同的签订阶段就提出软件验收项目和验收通过标准的意见；在软件的需求评审阶段，仔细审阅软件的需求规格说明书，指出不利于测试和可能存在歧义的描述；在开发方开发完软件并经过开发方内部仔细的测试后，对完成的软件进行评审或第三方的验收测试，提供完整的错误报告提交给用户方，由用户方根据之前签订的开发合同中相应的验收标准判断是否进行验收。

21.7.1 软件验收流程

软件验收流程图如图 21-7 所示。

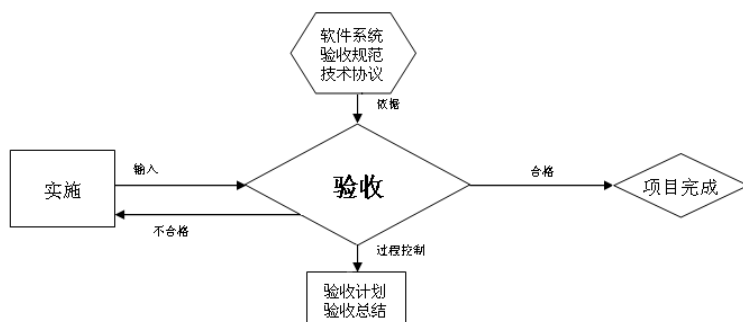


图 21-7 软件验收流程图



无兄弟，不编程

工作流程：实施完成后经用户确认，由开发部提交验收计划，并会同质量保证部、市场部和用户进行验收；验收完成后提交验收报告，软件开发及实施全部完成。

责任部门：开发部。

相关部门：用户、质量保证部、市场部。

相关资料：系统实施总结报告、用户意见表、验收计划、验收报告。

相关规范：软件系统验收标准、技术协议。

21.8 服务与维护

维护是指软件产品交付给客户之后的客户服务和产品维护。客户服务和产品维护的宗旨是提高客户对产品的满意度。项目组在完成产品开发的过程后，应根据实际情况组织编写培训大纲，同时对市场部技术人员做好培训工作。市场部技术人员接受培训之后，按培训大纲的内容整理并编写完整的培训手册，同时完成对客户的培训。用户在使用产品的过程中如对产品有何疑问或意见可咨询市场部，市场部技服人员有义务帮助客户解决问题。客户在使用产品的过程中如发现软件缺陷，原产品开发人员、维护人员有义务帮助客户解决缺陷。

21.8.1 责任人

开发部组织编写培训大纲，完成对市场部技术人员的培训工作。市场部技术人员经过培训组织编写培训手册，并完成客户的服务和培训。市场部应对客户在使用产品的过程中提出的问题进行收集，给予详细的解答，开发人员应对客户提出的软件缺陷给予解决。

21.8.2 收集信息

市场部在产品销售过程中定期通过上门或电话等方式回访客户，了解客户在使用产品的过程中遇到的问题，同时收集客户意见。客户通过各种渠道（如电话、Internet 等）向市场部客服人员提出服务请求。市场部客服人员应对客户提出的问题给予答复，对于自己可以解决的，立即给予解答，对于自己不能解决的，可以请相关的开发人员或测试组人员协助解决问题，同时做好服务与维护记录。如果客户提出软件维护，则转入维护分析；提出需求变更，则转入定制开发。

21.8.3 维护分析

一般来说，开发人员既要开发新的软件项目，又要维护老的软件项目（兼任维护人员），当维护人员接到客户或者市场部客服人员的维护请求时，需先进行维护分析。

➤ 如果用户因为不会使用产品的某些功能而请求帮助，那么维护人员应当尽快通过电

话或 Internet 指导用户操作。

- 如果用户发现了产品的缺陷，维护人员有义务尽快消除该缺陷，转入软件维护。
- 如果用户希望开发人员修改或者增加产品的功能，此时维护人员不可轻易答应客户，需按照《定制产品的开发规定》执行。

21.8.4 软件维护

- 维护人员和客户协商维护方式：如果可以通过 Internet 执行远程维护，那么采用远程维护以降低维护的成本。
- 维护人员在修改产品时，必须严格遵循《软件版本管理规定》来操作，避免工作成果的版本发生混乱。
- 维护人员必须对修改后的产品进行相关的测试，以免引入新的错误。
- 维护人员应填写《维护记录》。

21.8.5 改进

开发人员应定期对客户提出的需求改进和意见进行整理，同时收集市场的需求，通过对比分析，将比较通用的功能加入软件当中，从而提升软件的性能。

21.9 项目参考

在本章介绍的一些项目开发流程中，需要很多开发文档。本书以一个 CMS 项目（BroCMS）为例，在后面三个章节中分别提供了开发中最重要的三个文档。BroCMS 项目的源码存放在本书的配套光盘中，这个项目是在 BroPHP 框架基础上开发的，读者可以参考它去开发自己的项目，也可以提取部分功能直接应用到自己的项目中。

第22章

需求分析说明书

www.brophp.com

内容管理系统（BroCMS）

文件状态： [] 草稿 [√] 正式发布 [] 正在修改	文件标识：	LAMP 兄弟连-BroCMS-01
	当前版本：	V2.0
	作 者：	高洛峰
	完成日期：	2011-11-11

LAMP 兄弟连（北京易第优教育）
BroCMS

版本历史

版本/状态	作 者	参 与 者	起止日期	备 注
V1.0	高洛峰	教学组成员	2009-11-05 2009-11-20	《细说 PHP》第 1 版
V2.0	高洛峰	教学组成员	2011-11-15 2011-11-30	《细说 PHP》第 2 版

22.1 文档介绍

CMS 是 Content Management System 的缩写，即“内容管理系统”，即使用“系统”对网站的“内容”进行“管理”。BroCMS 的需求分析是认真查了用户对内容管理系统的需求后，根据 CMS 系统的业务分类、业务操作规程及数据结构等具体要求，调查了公司的业务范围、业务逻辑结构、业务操作规程、业务详本、业务数据规格，确定了系统性能要求、系统运行支持环境要求、数据项的名称、数据类型、数据规格。这一切都为下一步工作奠

定了良好的基础。

本系统的需求说明书全面、概括地描述了 CMS 系统所要完成的工作，使软件开发人员和用户对本系统中的业务流程及功能达成共识。通过本需求说明书，可以全面了解 CMS 系统所要完成的任务和所能达到的功能。

22.1.1 编写说明

此需求文档对公司正待开发的 CMS 系统做了比较全面的需求分析，对被开发软件系统的主要功能、性能进行完整描述。现以书面的形式将项目开发生命周期中的项目任务范围、项目团队组织结构、团队成员的工作责任、团队内外沟通协作方式、开发进度、检查项目工作等内容描述出来。目的如下：

(1) 作为项目相关人员之间的共识和约定，是软件系统开发技术协议的参考依据，为双方提供参考。

(2) 为软件开发者进行详细设计和编程提供基础。

(3) 保证项目开发人员按时保质地完成预定目标，更好地了解项目实际情况，按照合理的顺序开展工作。

(4) 作为项目生命周期内的所有项目活动的行动基础，使相关工作人员能了解到用户需求，并在开发、测试、验收、推广及维护过程中依据。

22.1.2 项目背景

内容管理系统 (CMS) 具有许多基于模板的优秀设计，可以加快网站开发的速度和减少开发的成本。CMS 的功能并不仅限于文本处理，也可以处理图片、Flash 动画、声像流、图像甚至电子邮件档案等。设计 CMS 的出发点是为了方便一些对于各种网络编程语言并不是很熟悉的用户用一种比较简单的方式来管理自己的网站。也就是可以让你不需要学习复杂的建站技术，不需要学习太多复杂的 HTML 语言，就能够利用 CMS 构建出一个风格统一、功能强大的专业网站。该项目开发的目的是作为《细说 PHP》的读者，以及 LAMP 兄弟连的学员在学习期间的一个标准化参考项目。

22.1.3 读者对象

系统应用人员（应用用户）；

美工设计人员（开发人员）；

程序开发者（开发人员）；

另外，PHP 新人项目设计和学习的参考文档（本书读者）。

22.1.4 参考资料

《细说 PHP》第 2 版；



无兄弟，不编程

《软件需求分析》；
内容管理系统（CMS）设计方案；
内容管理系统（CMS）项目审批表；
LAMP 兄弟连有关的规章制度。

22.1.5 术语与缩写解释

缩写、术语	解 释
CMS	内容管理系统
LAMP 兄弟连	北京易第优教育咨询有限公司
教程	《细说 PHP》
BroPHP	开源免费的超轻量级 PHP 框架
BroCMS	该 CMS 项目的名称
DFD 图	数据流图（Data Flow Diagram）：简称 DFD，它从数据传递和加工角度，以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化系统分析方法的主要表达工具及用于表示软件模型的一种图示法

22.2 任务概述

项目的需求说明是非常重要的，不仅可以让程序员了解产品的需求，知道开发的目标结果，也可以让系统应用人员了解业务流程设计是否完善。所以在项目需求说明书中，必须有详细的系统功能说明及用户的操作介绍。

22.2.1 产品的描述

BroCMS 系统作为 LAMP 兄弟连定制的内容管理系统，旨在为 LAMP 兄弟连的学员及《细说 PHP》的读者提供一个可以参考的标准化的项目开发全过程，当然也可以让一些非技术人员通过本系统轻松建设自己的站点，让你在任何时间、任何地点，通过网络方便地“生活”，不仅是信息传递与获取，还可以进行群体交流和资源共享，展示自我，为个人发展带来新机遇。通过本系统的应用可达到对网页的内容分类排版、文章内容的发布、用户之间的文章管理与交流。通过对 CMS 的调查分析，BroCMS 项目主要分为门户网站和后台管理系统两个部分应用。

1. 门户网站

用户分为访客和会员，访客可以在网站上浏览频道、浏览文章、搜索文章等。会员可以发布文章、对文章进行评论、加好友、发站内信、收藏文章等。

2. 后台管理系统

后台管理系统的用户分为超级管理员、网编与内容管理员三种角色：网编可进行系统

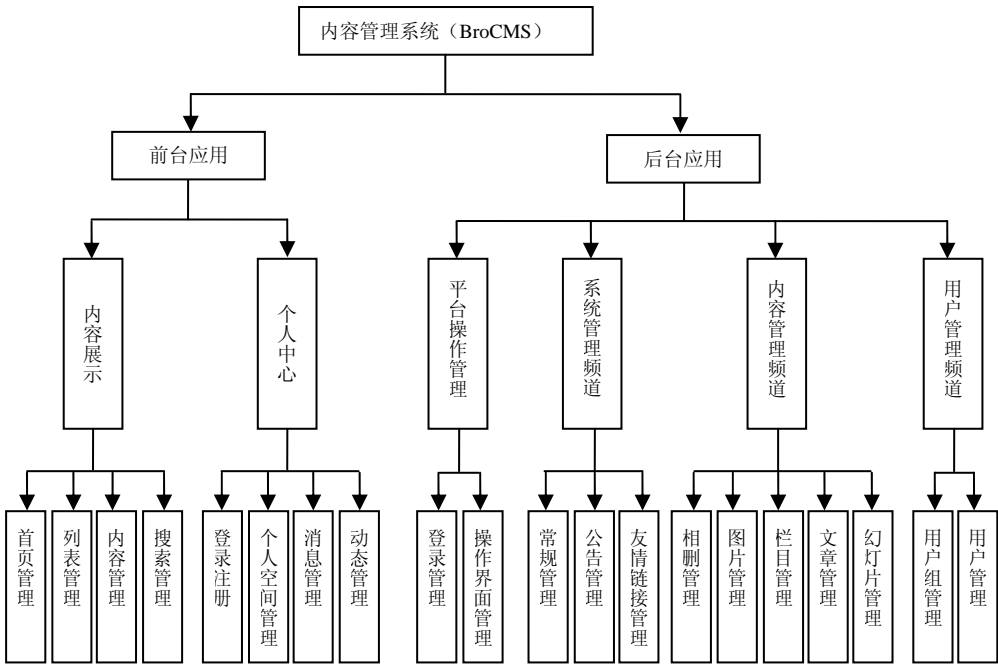
设计、管理栏目分类、友情链接管理及公告管理；内容管理员可以管理文章和幻灯片；超级管理员有所有权限，包括管理用户和用户组。

22.2.2 系统目标

- 根据需求分析和描述及与用户的沟通，现制定网站实现目标如下：
- 系统采用人机对话方式，界面美观友好，界面简洁，框架清晰，美观大方。
 - 灵活快速地发布文章信息；
 - 内容查询灵活、方便，数据存储安全可靠；
 - 实施强大的后台审核、栏目部署功能；
 - 实现强大的搜索功能，支持模糊查询。

22.2.3 系统功能结构

根据内容管理系统（CMS）的特点，可以将其分为前台和后台两个应用，前台应用包括内容展示和个人中心两部分，而后台应用则分为平台操作管理、系统管理频道、内容管理频道和用户管理频道 4 个部分。其中各个部分及其包括的具体功能模块如图 22-1 所示。



22-1 BroCMS 系统功能模块划分

22.2.4 系统流程图

内容管理系统模块为浏览者、会员、网站管理者等提供一个交流平台，根据角色的不

同，分别有不同的操作权限，BroCMS 的操作流程如图 22-2 所示。

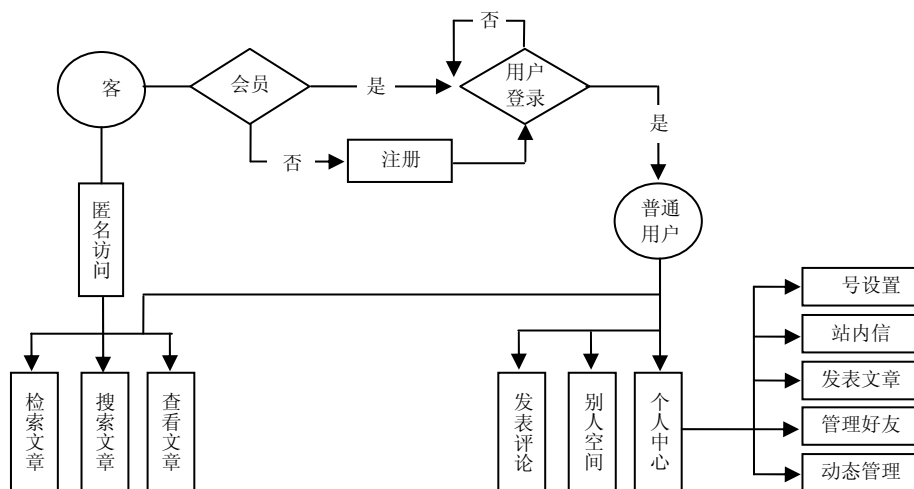


图 22-2 前台用户操作流程图

在内容管理系统中，浏览者也就是普通客只能够查看文章；注册的会员既可以查看文章，也可以发布和回复文章等；管理员则既是普通用户又可以登录后台对系统进行管理。后台管理的用户操作流程如图 22-3 所示。

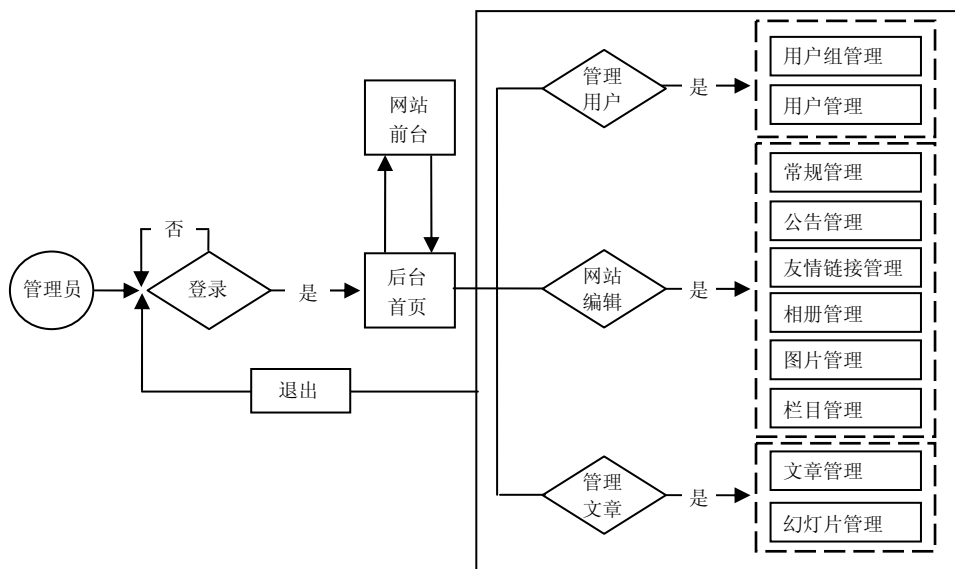


图 22-3 管理员登录系统后台的操作流程

管理员登录内容管理系统时，需要执行以下步骤。

- (1) 身份验证，只有具有管理用户、网站编辑和管理文章的权限的用户才能管理后台。
- (2) 根据不同的角色有不同的操作权限，超级管理员具有所有权限。

22.3 业务描述

虽然通过系统概述可以让我们了解到一些系统的功能结构和操作流程，但并不够详细，不能体现出项目的具体需求。所以就需要通过业务的详细描述让开发者和应用用户更深入地了解系统的需求。对功能构成及描述的规定包括按业务类型分类，一条列出实现的各项业务，以及对业务的详细描述，对系统需求的统一规定及要求，并对每一业务流程进行描述，说明各功能模块的简单实现。其中对各功能模块的描述应包括：

- 功能概述；
- 操作权限；
- 输入；
- 处理过程；
- 输出。

业务描述应详细准确，无二义性，以作为将来开发、测试和验收的标准。

22.3.1 后台登录管理

1. 功能构成

功能名称	后台登录管理	功能编号	F01	设计者	高洛峰
功能需求提出者（单位、名）	LAMP 兄弟连 高某某			完成时间	2011-11-5
功能修改提出者（单位、名）	LAMP 兄弟连 洛某某			修改时间	2011-11-5
功能修改批准者	峰某某	功能修改者	高洛峰	修改次数	2
功能框图： <div style="text-align: center; margin-top: 20px;"> <pre> graph TD A[登录管理] --> B[登录信息录入] </pre> </div>					
说明	实现用户登录操作，并进入系统的管理。 1 登录信息录入 进入后台的统一操作，通过操作登录界面录入正确的用户名称、密码及验证码信息后，再经过系统处理后才能进入管理平台进行操作				

2. 功能描述

功能需求表 f0101 登录信息录入

功能描述	实现用户登录操作
操作权限	具有用户管理、文章管理、网站设置三者之一的权限即可
输入	用户名称、用户密码、验证码



续表

加工 (处理过程)	除了验证码和其他条件验证以外，最主要的是根据用户名称和密码作为查询条件，在所有系统用户中进行查找，如果查找到并具有相应的操作权限，则可以顺利进入到后台操作平台，如果失败，则返回重新登录。
输出	用户全部信息及权限信息
DFD 图	<p>业务数据流程：</p> <pre>graph LR User[用户] -- 用户信息 --> Process((处理用户信息)) Record[用户记录] -- 用户及权限信息 --> Process Process -- 用户及权限信息 --> Page[后台首页]</pre>
注释	处理用户信息前一定要先进行验证（都不能为空，验证码确认）

3. 功能预览（如图 22-4 所示）



图 22-4 用户登录原型图

22.3.2 公告管理

1. 功能构成

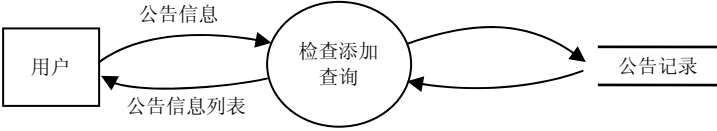
功能名称	公告管理		功能编号	F04	设计者	高洛峰
功能需求提出者（单位、 名）			LAMP 兄弟连 高某某		完成时间	2011-11-5
功能修改提出者（单位、 名）			LAMP 兄弟连 洛某某		修改时间	2011-11-5
功能修改批准者	峰某某	功能修改者	高洛峰	修改次数	1	
功能框图：						
<div><div>公告管理</div><div><div>写公告</div><div>查询公告</div><div>公告排序</div><div>编辑公告</div><div>删除公告</div></div></div>						
说明	<p>公告可以及时并动态地发布管理者的公开信息，公告管理系统也是网站中一个常用的组件，它可以浏览查询、 写、编辑和删除公告。</p> <p>1 写公告</p> <p>通过系统提供的表单界面 写并发布新的公告。</p>					

续表

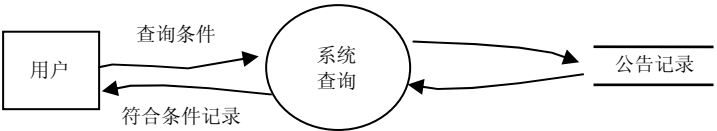
	<div>2 查询公告 可以根据多种条件筛选出需要处理的公告列表。</div> <div>3 公告排序 在查询列表中可以改变公告显示的顺序。</div> <div>4 编辑公告 和添加公告类似，对已发布的公告进行编辑修改。</div> <div>5 删除公告 对已经过期的或已经不需要的公告可以进行删除</div>
--	---

2. 功能描述

功能需求表 f0401 添加公告信息

功能描述	写公告
操作权限	需要网站设置权限
输入	公告标题、标题 色、起始日期、 止日期、公告内容、显示状态
加工 (处理过程)	对用户输入的公告信息进行检查并添加到数据库的公告记录中
输出	全部公告记录
DFD 图	<div>业务数据流程： </div>
注释	公告的标题、公告内容及起始日期必须录入

功能需求表 f0402 查询公告信息

功能描述	查询公告
操作权限	需要网站设置权限
输入	显示 不显示 无期限的 过期的 没到期的 全部显示 其中的一个条件
加工 (处理过程)	通过查询条件到数据库的公告记录中查出满足条件的记录
输出	符合条件的公告记录
DFD 图	<div>业务数据流程： </div>
注释	<div>色的记录被设置为不显示；</div> <div>如果结束时间为 色表示已经过期；</div> <div>如果开始时间为 色表示还没有到期</div>



功能需求表 f0403 公告排序

功能描述	公告排序
操作权限	需要网站设置权限
输入	公告编号、顺序号
加工 (处理过程)	通过每个公告的编号和提交的对应顺序号对公告重新进行排序
输出	排序后的公告记录
DFD 图	业务数据流程：
注释	可以通过输入整数改变公告显示顺序，按从小到大的排列顺序

功能需求表 f0404 修改公告信息

功能描述	修改公告
操作权限	需要网站设置权限
输入	公告编号、公告标题、标题 色、起始日期、止日期、公告内容、显示状态
加工 (处理过程)	对用户输入的公告信息进行检查并修改数据库中原来的公告记录
输出	全部公告记录
DFD 图	业务数据流程：
注释	公告的标题、公告内容及起始日期必须输入

功能需求表 f0405 删除公告信息

功能描述	删除公告
操作权限	需要网站设置权限
输入	公告编号
加工 (处理过程)	根据提供的公告编号对数据中的公告记录进行删除
输出	全部公告记录
DFD 图	业务数据流程：
注释	如果公告中有图片发布，删除公告时一定要将公告中发布的图片一起删除

3. 功能预览（如图 22-5 所示）



图 22-5 公告管理模块原型图

22.4 系统运行环境

22.4.1 硬件环境

- (1) 服务器的中 处理部件（CPU）建议使用 PIII 1GHz（以上）Xeon 处理器 片。
- (2) 服务器内存必须使用服务器专用 ECC 内存。
- (3) 为了保证数据存储的绝对可靠， 盘应使用 盘 列（RAID01）。

22.4.2 软件环境

- 开发内容管理系统（BroCMS）项目使用的软件开发环境如下。
- (1) 服务器端
 - 操作系统：Linux（推荐）/Windows NT
 - Web 服务器：Apache 2.2.9
 - 数据库：MySQL 5.0.51
 - 开发语言：PHP 5.2.6
 - (2) 客户端



无兄弟，不编程

- 浏览器：IE 6.0 以上版本/Mozilla Firefox
- 界面布局：DIV+CSS
- 页面特效：JavaScript
- 分辨率：最佳效果 1024 × 768 及以上像素

(3) 开发工具

- vim 或 Zend Studio

22.5 需求设计评审

需求设计评审的成员需要由有经验的专家和对业务熟悉的专员组成，目的是确认业务流程图和数据字典是否完全正确地反映了业务活动，确认该阶段的任务是否全部完成。保证设计质量，以免造成重大疏漏或者错误，并将需求分析产生的数据流图、数据字典、功能结构图等返回给用户，与用户一起检查、补充、修改，最终获得用户的认可。

第23章

数据库设计说明书

www.brophp.com

内容管理系统（BroCMS）

文件状态：	文件标识：	LAMP 兄弟连-BroCMS-02-DATABASE
<input type="checkbox"/> 草稿	当前版本：	2.0
<input checked="" type="checkbox"/> 正式发布	作 者：	高洛峰
<input type="checkbox"/> 正在修改	完成日期：	2011-11-15

版本历史

版本/状态	作者	参与者	起止日期	备注
1.0	高洛峰	教学组成员	2009-11-05 2009-11-20	《细说 PHP》第 1 版
2.0	高洛峰	教学组成员	2012-03-15 2012-03-30	《细说 PHP》第 2 版

23.1 引言

在使用任何数据库之前，都必须设计好数据库，包括将要存储的数据的类型，数据之间的相互关系及数据的组织形式。数据库设计是指对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据。在 CMS 项目中总是需要处理大量的数据资源，这正是内容管理系统的基础和核心，为了合理地组织和高效率地存取数据，目前最好的方式，就是建立数据库系统，因此在系统的总体设计阶段，数据库的建立与设计是一项十分重要的内容。由于数据库应用系统的复杂性，为了支持相关程序运行，数据库设计就变得异常复杂，因此最佳设计不可能一而就，而只能是一种“反复，步求精”的过程，也就是规划和结构化数据库中的数据对象及这些数据对象之间关系的过程。



23.1.1 编写目的

一个成功的管理系统，是由 50%的业务+50%的软件 组成的，而 50%的成功软件又由 25%的数据库+25%的程序 组成，数据库设计的好 是一个关键。如果把企业的数据比做生命所必需的 ，那么数据库的设计就是应用中最重要的一部分，是一个系统的根基。在 BroCMS 内容管理系统的需求分析和系统概要设计的基础上，对数据进行分析，在结构上进行设计，用于开发人员进行项目设计，以此作为编码的依据，同时也为后续的数据库维护工作提供了良好的使用说明，也可以作为 来版本升级时的重要参考资料。数据库设计的目标是建立一个合适的数据模型。这个数据模型应当是满足用户要求的，既能合理地组织用户需要的所有数据，又能支持用户对数据的所有处理功能。也要满足 CMS 数据库管理系统的要求，又能够在数据库管理系统中实现。并且要具有较高的范式，数据完整性好，效率高，便于理解和维护，没有数据冲突。

23.1.2 背景

名 称	说 明
数据库名称	BroCMS（兄弟连内容管理系统）
数据库系统	MySQL5.0
客户端连接工具	MySQL Command Line Client
项目任务提出者	LAMP 兄弟连
项目开发者	高洛峰
使用用户	使用用户：《细说 PHP》读者及 LAMP 兄弟连学员

注：些数据库设计说明书文档范围只适用于内容管理系统 BroCMS V2.0，作为 Web 程序员项目设计和学习的参考文档。

23.1.3 定义

CMS：Content Management System，内容管理系统

E-R 图：实体关系图

23.1.4 参考资料

- A 《细说 PHP》教程
- B 《BroCMS 项目需求分析说明书》
- C www.brophp.com 和 bbs.lampbrother.net
- D 本项目相关的其他参考资料

23.2 外部设计

外部设计是研究和考虑所要建立的数据库的信息环境，对数据库应用领域中各种信息要求和操作要求进行详细的分析，了解应用领域中数据项、数据项之间的关系和所有的数据操作的详细要求，了解哪些因素对响应时间、可用性和可靠性有较大的影响等各方面的因素。

23.2.1 标识符和状态

数据库表前缀：bro_

用户名：root

密码：123456

权限：全部

有效时间：开发阶段

说明：系统正式发布后，可能更改数据库用户/密码，请在统一位置编写数据库连接字符串，在发行前请予以改正。

23.2.2 使用它的程序

本系统主要利用 PHP 作为前端的应用开发工具，使用 MySQL 作为后台的数据库，Linux 或 Windows 均可作为系统平台。

23.2.3 约定

- 所有命名一定要具有描述性，绝一切 或 文混杂的命名方式。
- 字符集采用 UTF-8，请注意字符的转换。
- 所有数据表第一个字段都是系统内部使用 r 主键列，自增字段，不可空，名称为 id，确保不把此字段 给最终用户。
- 除特别说明外，所有日期格式都采用 int 格式，无时间值。
- 除特别说明外，所有字段默认都设置不允许为空，需要设置默认值。
- 所有普通索引的命名都是表名加设置索引的字段名组合，例如用户表 User 中 name 字段设置普通索引，则索引名称命名方式为 user_name。

23.2.4 支持软件

操作系统：Linux / Windows

数据库系统：MySQL



无兄弟，不编程

查询浏览工具：PHPMYAdmin

命令行工具：mysql

注意：mysql 命令行环境下对中文支持不好，可能无法书写带有中文的 SQL 语句，也不要使用 PHPMYAdmin 录入中文。

23.3 结构设计

数据库的结构设计中有许多需要考虑的因素，对数据库的背景、应用环境等方面都需要有深入的了解，只有有一个对所有这些因素都很了解的数据库设计专家，设计的数据库才能易于使用和维护，并且具有高效和一致的特点。虽然这样只对数据库设计过程有一个概要的了解，但是仍然有助于读者了解和掌握 SQL，使读者可以很好地分析数据间的相互关系，在使用 SQL 进行报表的生成、子查询及视图等操作时，可以更好地进行操作。

23.3.1 概念结构设计

概念数据库的设计是进行具体数据库设计的第一步，概念数据库设计的好坏直接影响到逻辑数据库的设计，影响到整个数据库的好坏。在 BroCMS 系统的分析阶段，我们已经得到了系统的数据流程图和数据字典，现在就是要结合数据规范化的理论，用一种模型将用户的数据要求明确地表示出来。概念数据库的设计应该极易于转换为逻辑数据库模式，又容易被用户所理解。概念数据库设计中最主要的就是采用实体-关系数据模型来确定数据库的结构。数据是表达信息的一种重要的量化符号，是信息存在的一种重要形式。数据模型则是数据特点的一种抽象。它描述的是数据的共性，而不是个别的数据。一般来说，数据模型包含两方面内容。

(1) 数据的静态特性：主要包括数据的基本结构、数据间的关系和数据之间的相互约束等特性。

(2) 数据的动态特性：主要包括对数据进行操作的方法。

在数据库系统设计中，建立反映客观信息的数据模型，是设计中最为重要的，也是最基本的步骤之一。数据模型是连接客观信息世界和数据库系统数据逻辑组织的桥梁，也是数据库设计人员与用户之间进行交流的共同基础。概念数据库中采用的实体-关系模型，与传统的数据模型有所不同。实体-关系模型是面向现实世界，而不是面向实现方法的，它主要是用于描述现实信息世界中数据的静态特性。而不涉及数据的处理过程。但由于它简单易学，且使用方便，因而在数据库系统应用的设计中，得到了广泛应用。实体-关系模型可以用来说明数据库中实体的等级和属性。以下是实体-关系模型中的重要标识：

- 在数据库中存在的实体
- 实体的属性
- 实体之间的关系

23.3.1.1 实体和属性的定义

按照定义的数据类型和属性创建实体和实体属性列表。实体形成表，如“用户”就是一个实体，属性则为表中的列，如对应于实体“用户”属性包含“用户名”、“用户 ID”等。

1 实体

实体是实体 关系模型的基本对象，是现实 界中各种事物的抽象。 是可以相互区别开并可以被识别的事、物、概念等对象均可认为是实体。在本书示例的简单的 BroCMS 数据库中，基本的实体列表如下：

- 相册
- 栏目
- 图片
- 文章
- 幻灯片
- 评论
- 用户组
- 用户
- 站内信
- 公告
- 友情链接
- 动态

在 制实体 关系图（E-R 图）时，实体出现在 形中，如图 23-1 所示。



图 23-1 表示实体的 E-R 图

一般来说，每个实体都相当于数据库中的一个表。以上介绍的实体都是强实体，每个实体都有自己的键。但是在实际领域中，经常存在一些实体，它们没有自己的键，这样的实体称为 实体。 实体中不同的记录有可能完全相同，难以区别，这些值依 于另一个实体（强实体）的意义，必须与强实体联合使用。在创建了实体之后，就可以标识各个实体的属性了。

2 属性

每个实体都有一组特 或性质，称为实体的属性。实体的属性值是数据库中存储的主要数据，一个属性实际上相当于表中的一个列。下面来看看“文章”（article）实体。这个



实体具有哪些属性呢？对于一篇文章来说，都具有文章标题、文章简介、添加时间、文章来源、文章内容、关键字、访问次数、推荐状态、审核状态。所以关于“文章”实体的属性如下：

- 文章标题 (title)
- 文章编号 (id)
- 文章简介 (summary)
- 添加时间 (posttime)
- 文章来源 (comefrom)
- 文章内容 (content)
- 关键字 (keyword)
- 访问次数 (views)
- 推荐状态 (recommend)
- 审核状态 (audit)

实体“栏目 (column)”包含的属性如下：

- 栏目标题 (title)
- 栏目路径 (path)
- 栏目描述 (description)
- 排序编号 (ord)

由于篇幅有限，这里就不列出所有实体的属性了，在绘制 E-R 图时，属性由椭圆包围，在属性和它所属的实体间使用直线进行连接，以实体“文章”为例进行示例，如图 23-2 所示。

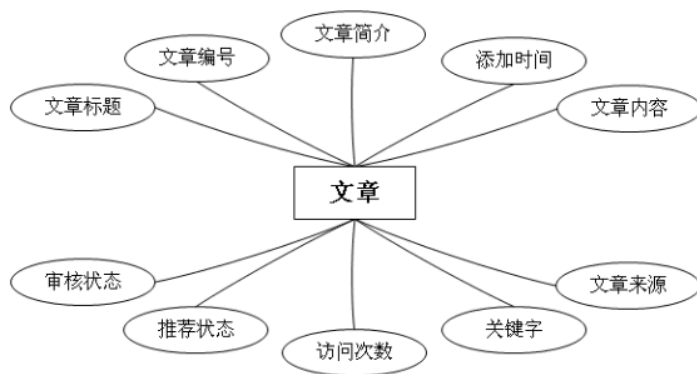


图 23-2 包含属性的 Department 的 E-R 图

对于每个实体，都有其确定的主属性（实体中的主属性实际上相当于表中的主键），就可以唯一地确定实体的每个记录。最好是创建一个单独的属性作为主属性，在实体文章中可以选择“文章编号”作为主属性，在绘制 E-R 图时，主属性在属性下画下划线来说明。以实体“文章”为例，如图 23-3 所示。

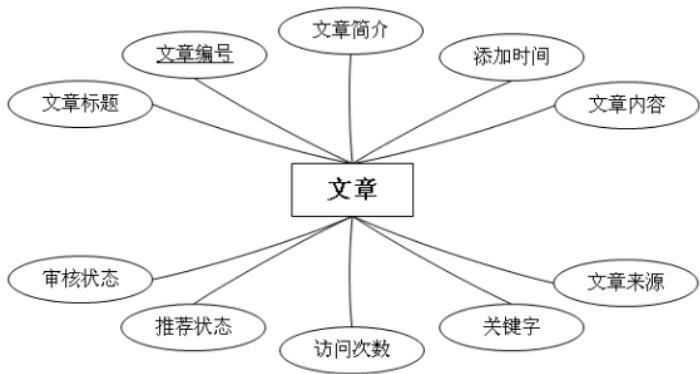


图 23-3 定义了主属性的“文章”的 E-R 图

注意：在数据库设计中，选择和设置列作为主键是一个关键步骤。

23.3.1.2 E-R 图的绘制

实体 关系图是表现实体 关系模型的图形工具，简称 E-R 图。节会以 BroCMS 数据库为例，给出一个完整的数据库的 E-R 图设计示例。图 23-4 给出了在 E-R 图中使用的各种元素的图形符号。

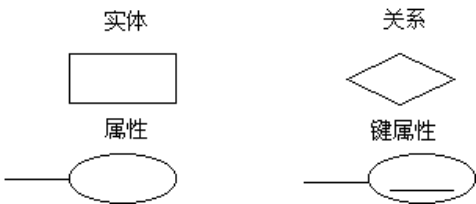


图 23-4 E-R 图中使用的各种元素的图形符号

在 E-R 图中，实体之间的关系以 形表示，关系中各方面的表通过直线与 形中的关系名称相连接。还要为每个关系命名一个“关系名称”，实体与关系相连的直线 都根据关系的属性标注有“1”或“N”。

E-R 图为读者的数据库提供了一个不错的 图，可以分成三步进行：首先设计局部 E-R 图；然后合并各局部 E-R 图，并解决可能存在的冲突，得到初步 E-R 图；最后修改和重构初步 E-R 图，消除其中的 部分，得到最终的全局 E-R 图，即概念模式。设计全局 E-R 模式的目的在于把 干局部 E-R 模式形式上合并为一个 E-R 模式，而在于消除冲突，使之成为能够被全系统中所有用户共同理解和接受的统一的概念模型。使设计人员仅从用户角度看待数据及处理要求和约束，产生一个反 用户观点的概念模式。

23.3.1.3 设计局部 E-R 模式

先设计局部 E-R 图，也称用户视图。在设计初步 E-R 图时，要尽量能充分地把组织中各部门对信息的要求集中起来，而不需要考虑数据的 问题。局部概念模型设计是从用户的观点出发，设计符合用户需求的概念结构。局部概念模型设计的就是组织、分类收集到的数据项，确定哪些数据项作为实体，哪些数据项作为属性，哪些数据项是同一实体的



属性等。确定实体与属性的原则如下。

- 能作为属性的尽量作为属性而不要划为实体；
- 作为属性的数据元素与所描述的实体之间的联系只能是 1:n 的联系；
- 作为属性的数据项不能再用其他属性加以描述，也不能与其他实体或属性发生联系。

图 23-5 图 23-9 所示是 BroCMS 系统的部分局部 E-R 图的设计。

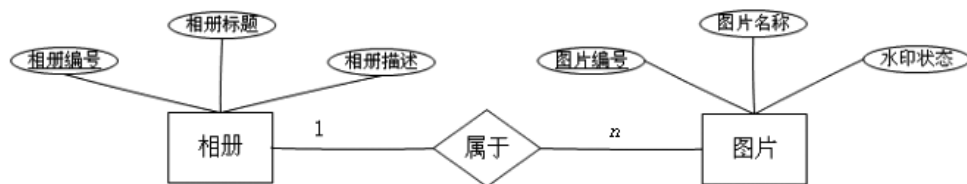


图 23-5 相册、图片的 E-R 图

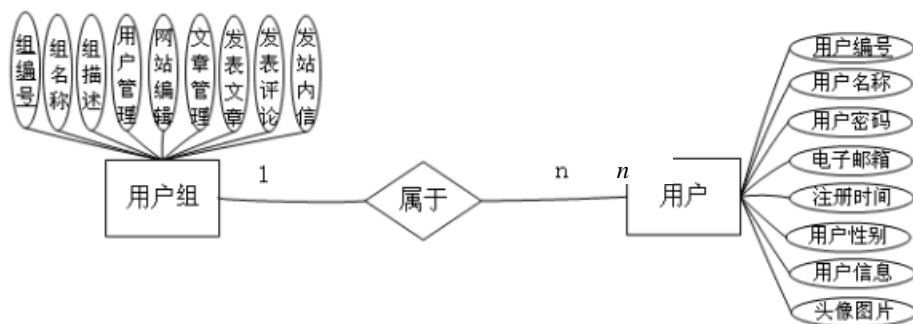


图 23-6 用户组、用户的 E-R 图

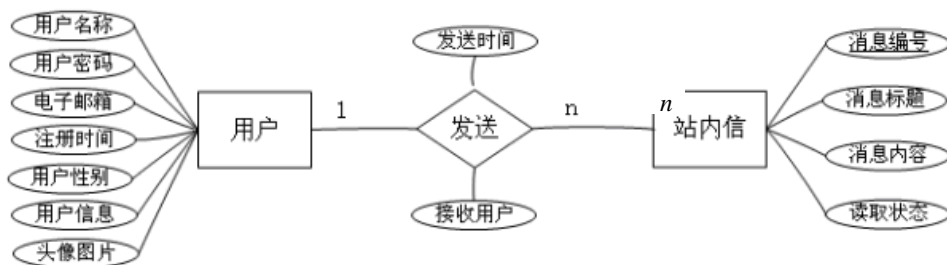


图 23-7 用户、站内信的 E-R 图

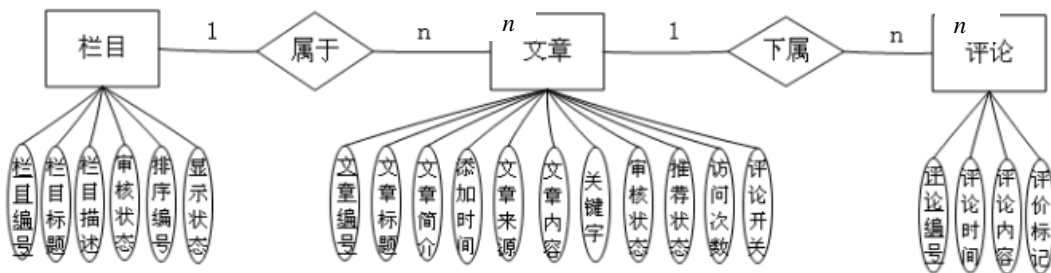


图 23-8 栏目、文章、评论的 E-R 图

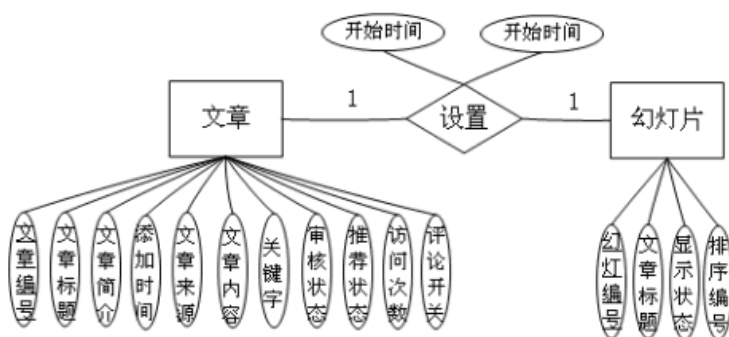


图 23-9 文章、幻灯片的 E-R 图

23.3.1.4 设计全局 E-R 模式

合各局部 E-R 图，形成总的 E-R 图，即用户视图的集成。所有局部 ER 模式都设计好了后，接下来就是把它们 合成单一的全局概念结构。全局概念结构不仅要支持所有局部 ER 模式，而且必须合理地表示一个完整、一致的数据库概念结构，如图 23-10 所示。

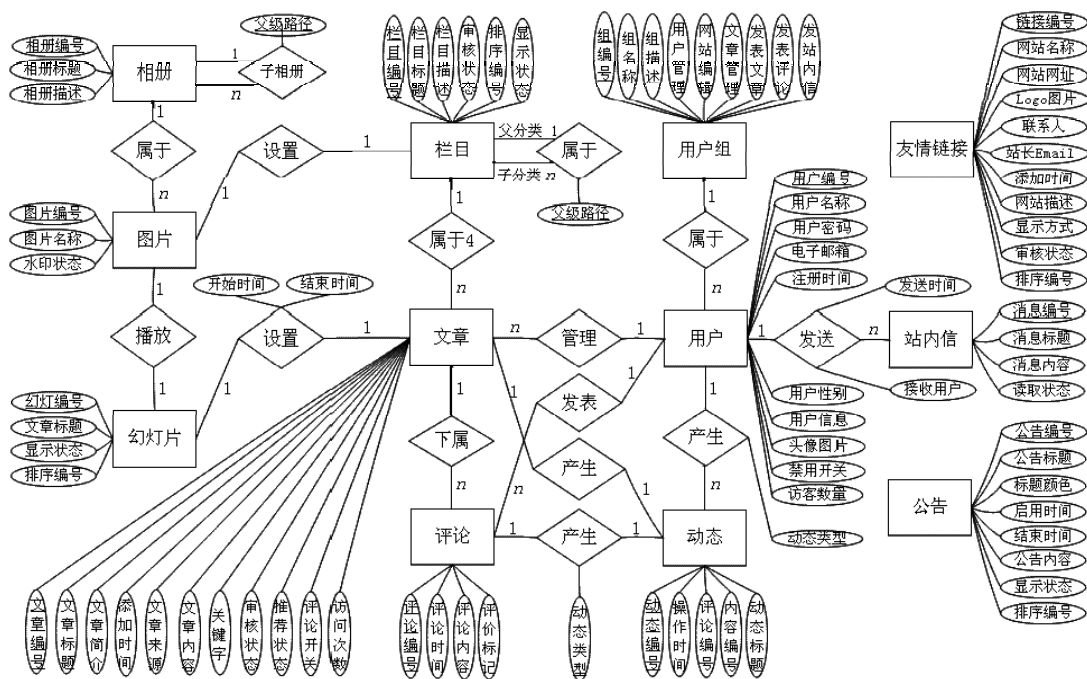


图 23-10 BroCMS 的全局 E-R 图

另外，在进入下一节之前，先回 一下概念数据库的设计，其中主要是实体 关系模型的建立。简要总结一下实体 关系模型建立的步骤。

(1) 对需求进行分析，从而确定系统中所包含的实体。

(2) 分析得出每个实体所具有的属性。

(3) 保证每个实体有一个主属性，该主属性可以是实体的一个属性或多个属性的组合。主属性必须能唯一地描述每个记录。



(4) 确定实体之间的关系。

经过这些步骤后，读者就可以制出 E-R 图。之后可以再看看数据库的需要，判断是否获取了所需的信息，是否有遗漏信息等，读者可以再对 E-R 图进行修改，添加或删除实体与属性。

23.3.1.5 全局 ER 模式的优化

在得到全局 ER 模式后，为了提高数据库系统的效率，还应进一步依据处理需求对 ER 模式进行优化。一个好的全局 ER 模式，除能准确、全面地反映用户功能需求外，还应满足下列条件：

- (1) 实体类型的个数要尽可能少
- (2) 实体类型所含属性个数尽可能少
- (3) 实体类型间联系无

23.4 逻辑结构设计

逻辑结构设计的任务是把概念设计阶段建立的基本 E-R 图，按照选定的内容管理系统软件支持的数据模型，转化成相应的逻辑设计模型。也就是可以将实体、实体间的关系等模型结构转变为关系模式，即生成数据库中的表，并确定表的列。下述论由实体关系模型生成表的方法。

1. 任务

将基本 E-R 图转换为与选用 DBMS 产品所支持的数据模型相符合的逻辑结构。

2. 过程

- (1) 将概念结构转换为现有 DBMS 支持的关系模型。
- (2) 从功能和性能要求上对转换的模型进行评价，看它是否满足用户要求。
- (3) 对数据模型进行优化。

23.4.1 ER 图向关系模型的转化

在上面实体之间的关系的基础上，将实体、实体的属性和实体之间的联系转换为关系模式。这种转换的原则如下。

- 一个实体转换为一个关系，实体的属性就是关系的属性，实体的码就是关系的码。
- 一个联系也转化为一个关系，联系的属性及联系所连接的实体的码都转化为关系的属性，但是关系的码会根据关系的类型变化，如果是：
 - (1) 1:1 联系，两端实体的码都成为关系的候选码。
 - (2) 1:n 联系，n 端实体的码成为关系的码。
 - (3) m:n 联系，两端的实体码的组成为关系的码。

23.4.2 确定关系模式

根据转换算法，E-R 图中有 12 个实体类型，可以转换成 12 个关系模式。

- (1) 相册（相册编号，上级编号，父级路径，相册标题，相册描述）
- (2) 图片（图片编号，类别编号，图片名称，水印状态）
- (3) 分类栏目（栏目编号，上级编号，父级路径，栏目标题，栏目描述，图片编号，审核状态，排序编号，显示状态）
- (4) 文章（文章编号，文章标题，文章简介，添加时间，用户编号，文章来源，文章内容，关键字，类别编号，审核状态，推荐状态，评论开关，访问次数）
- (5) 幻灯片（幻灯编号，文章编号，文章标题，图片编号，开始时间，结束时间，显示状态，排序编号）
- (6) 评论（评论编号，用户编号，文章编号，评论时间，评论内容，评价标记）
- (7) 用户组（用户组编号，用户组名称，用户组描述，用户管理权限，文章管理权限，发表文章权限，发表评论权限，发站内信权限）
- (8) 用户（用户编号，组编号，用户名称，用户密码，电子邮箱，注册时间，用户性别，用户信息，头像图片，用开关，访客数量）
- (9) 站内信（消息编号，消息标题，用户编号，接收用户，发 时间，消息内容，读取状态）
- (10) 公告（公告编号，公告标题，标题 色，启用时间，结束时间，公告内容，显示状态，排序编号）
- (11) 友情链接（链接编号，网站名称，网站网址，LOGO 图片，联系人名字，站长 E-mail，添加时间，网站描述，显示方式，审核状态，排序编号）
- (12) 动态信息（动态编号，用户编号，动态类型，操作时间，评论编号，内容编号，动态标题）

23.4.3 消除冗余

所谓 的数据，是指可由基本数据导出的数据， 的联系是指可由其他联系导出的联系。 数据和 联系容易 数据库的完整性，给数据库的维护增加 难，应当予以消除。本系统的 数据和 关系已经在概念结构设计中处理过了，这里不再进行过多的 述。

23.5 物理结构设计

数据库设计的最后阶段是确定数据库在物理设备上的存储结构和存取方法，也就是设计数据库的物理数据模型，主要是设计表结构。一般地，实体对应于表，实体的属性对应于表的列，实体之间的关系称为表的约束。逻辑设计中的实体大部分可以转换成物理设计中的表，但是它们并不一定是一一对应的。本次项目开发采用 MySQL 建立数据库。



23.5.1 设计数据表结构

在利用 MySQL 创建一个新的数据表以前，应当根据逻辑模型和数据字典先分析和设计数据表，描述出数据库中基本表的设计。需要确定数据表名称、所包含字段名称、数据类型、宽度，以及建立的主键、外键等描述表的属性的内容。BroCMS 项目全部 12 个数据表结构设计如表 23-1 表 23-12 所示。

表 23-1 相册表

表 名	bro_album 用于保存相册记录，表引擎为 MyISAM 类型，字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	SMALLINT (5)	无符号/非空/自动增	主键	相册编号
pid	SMALLINT (5)	无符号/非空/缺省 0	外键/普通索引 (album_pid)	上级编号
path	VARCHAR (100)	非空/缺省''	外键/普通索引 (album_path)	父级路径
title	VARCHAR (100)	非空/缺省''		相册标题
description	VARCHAR (200)	非空/缺省''		相册描述
补充说明	父级路径：保存了所有顶层父类的关系，使用“-”分隔的字符串			

表 23-2 图片表

表 名	bro_image 用于保存图片记录，表引擎为 MyISAM 类型，字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	INT (11)	无符号/非空/自动增	主键	图片编号
pid	SMALLINT (5)	无符号/非空/缺省 0	外键/普通索引 (image_pid)	类别编号
name	CHAR (24)	非空/缺省''		图片名称
water	TINYINT(1)	非空/缺省 0		水印状态
补充说明	水印状态：表示图片是否使用水印，0 表示没有水印，1 表示使用了水印			

表 23-3 分类栏目表

表 名	bro_column 用于保存栏目分类记录，表引擎为 MyISAM 类型，字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	SMALLINT (5)	无符号/非空/自动增	主键	栏目编号
pid	SMALLINT (5)	无符号/非空/缺省 0	外键/普通索引 (column_pid)	上级编号
path	VARCHAR (100)	非空/缺省''	外键/普通索引 (column_path)	父级路径
title	VARCHAR (100)	非空/缺省''		栏目标题
description	VARCHAR (200)	非空/缺省''		栏目描述
picid	SMALLINT (5)	无符号/非空/缺省 0	外键/普通索引 (column_picid)	图片编号
audit	SMALLINT (1)	无符号/非空/缺省 1	普通索引 (column_audit)	审核状态
ord	SMALLINT (3)	无符号/非空/缺省 0	普通索引 (column_ord)	排序编号
display	SMALLINT (3)	无符号/非空/缺省 1	普通索引 (column_display)	显示状态
补充说明	父级路径：保存了所有顶层父类的关系，使用“-”分隔的字符串 审核状态：表示该栏目下的文章是否需要人工审核，1 为需要，0 为不需要 排序编号：设置顶层栏目在页面中的显示顺序，以数值的从小到大顺序排列 显示状态：用于设置栏目的显示或隐藏，值 1 为显示，值 0 为隐藏			

表 23-4 文章表

表 名	bro_article 用于保存文章记录, 表引擎为 MyISAM 类型, 字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	INT (11)	无符号/非空/自动增	主键	文章编号
title	VARCHAR (50)	非空/缺省''	普通索引 (article_title)	文章标题
summary	VARCHAR(200)	非空/缺省''		文章简介
posttime	INT(10)	无符号/非空/缺省 0		添加时间
uid	INT(11)	无符号/非空/缺省 0	外键/普通索引 (article_uid)	用户编号
comefrom	VARCHAR(50)	非空/缺省''		文章来源
content	TEXT	非空		文章内容
keyword	VARCHAR(20)	非空/缺省''	普通索引 (article_keyword)	关键字
pid	SMALLINT(5)	无符号/非空/缺省 0	外键/普通索引 (aricle_pid)	类别编号
audit	SMALLINT(1)	无符号/非空/缺省 0	普通索引 (article_audit)	审核状态
recommend	SMALLINT(1)	无符号/非空/缺省 0	普通索引 (article_recommend)	推荐状态
allow	SMALLINT(1)	无符号/非空/缺省 1	普通索引 (article_allow)	评论开关
views	SMALLINT(5)	无符号/非空/缺省 0		访问次数
补充说明	用户编号: 使用这个字段关联用户 类别编号: 使用这个字段关联文章所属的类别 审核状态: 用于标记文章是否审核, 值 1 为审核通过, 值 0 表示还没审核 推荐状态: 数值越大推荐的人就越多, 用户每推荐一次数值增 1 评论开关: 设置文章是否允许评论, 值 1 为允许评论, 值 0 则不允许评论			

表 23-5 幻灯片表

表 名	bro_play 用于保存幻灯片记录, 表引擎为 MyISAM 类型, 字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	SMALLINT (5)	无符号/非空/自动增	主键	幻灯编号
aid	INT(11)	无符号/非空/缺省 0	外键/普通索引 (play_aid)	文章编号
title	VARCHAR(80)	非空/缺省''		文章标题
picid	SMALLINT(5)	无符号/非空/缺省 0	外键/普通索引 (article_picid)	图片编号
starttime	INT(10)	无符号/非空/缺省 0	普通索引 (article_starttime)	开始时间
endtime	INT(10)	无符号/非空/缺省 0	普通索引 (article_endtime)	结束时间
display	SMALLINT(1)	无符号/非空/缺省 1	普通索引 (article_display)	显示状态
ord	SMALLINT(3)	无符号/非空/缺省 0	普通索引 (article_ord)	排序编号
补充说明	图片编号: 用于设置幻灯片 放的图片			

表 23-6 评论表

表 名	bro_comment 用于保存用户评论记录, 表引擎为 MyISAM 类型, 字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	INT (11)	无符号/非空/自动增	主键	评论编号
uid	INT(11)	无符号/非空/缺省 0	外键/普通索引 (comment_picid)	用户编号
aid	INT(11)	无符号/非空/缺省 0	外键/普通索引 (comment_aid)	文章编号



续表

列 名	数据类型	属 性	约束条件	说 明
ptime	INT(10)	无符号/非空/缺省 0		评论时间
content	TEXT	非空		评论内容
cmt	SMALLINT(5)	非空/缺省 0		评价标记
补充说明	用户编号：用于标记评论所属用户 文章编号：用于标记评论所属文章 评价标记：表示用户评价级别，包括：0 为中立、1 为好评、-1 为 评			

表 23-7 用户组表

表 名	bro_group 用于保存用户组记录，表引擎为 MyISAM 类型，字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	SMALLINT (4)	无符号/非空/自动增	主键	用户组编号
groupname	VARCHAR(20)	非空/缺省''		用户组名称
description	VARCHAR(200)	非空/缺省''		用户组描述
useradmin	TINYINT(1)	非空/缺省 0		用户管理权限
webadmin	TINYINT(1)	非空/缺省 0		网站编辑权限
articleadmin	TINYINT(1)	非空/缺省 0		文章管理权限
sendarticle	TINYINT(1)	非空/缺省 0		发表文章权限
sendcomment	TINYINT(1)	非空/缺省 0		发表评论权限
sendmessage	TINYINT(1)	非空/缺省 0		发站内信权限
补充说明	在用户组中可以设置 6 个权限，设置方式相同，值 1 为 有权限，值 0 为没有权限。所属该用户组中的所有用户 有该组设置的权限			

表 23-8 用户表

表 名	bro_user 用于保存用户记录，表引擎为 MyISAM 类型，字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	INT (11)	无符号/非空/自动增	主键	用户编号
gid	SMALLINT(4)	无符号/非空/缺省 0	外键/普通索引 (user_gid)	组编号
username	VARCHAR(20)	非空/缺省''	普通索引 (user_username)	用户名称
userpwd	VARCHAR(40)	非空/缺省''	普通索引 (user_userpwd)	用户密码
email	VARCHAR(60)	非空/缺省''		电子邮箱
regtime	INT(10)	无符号/非空/缺省 0		注册时间
sex	SMALLINT(3)	非空/缺省 0		用户性别
info	VARCHAR(120)	非空/缺省''		用户信息
upic	CHAR(24)	非空/缺省''		头像图片
disable	SMALLINT(3)	无符号/非空/缺省 0	普通索引 (user_disable)	用开关
views	SMALLINT(5)	无符号/非空/缺省 0		访客数量
补充说明	用户密码：使用 MD5 加密 用户性别：有三个值，1 为男，2 为女，0 为保密。可以用来设置默认的用户头像 用开关：0 为开启，1 为 用用户			

表 23-9 站内信表

表 名	bro_message 用于保存发表的站内信记录，表引擎为 MyISAM 类型，字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	INT (11)	无符号/非空/自动增	主键	消息编号
title	VARCHAR(80)	非空/缺省''		消息标题
uid	INT(11)	无符号/非空/缺省 0	外键/普通索引 (message_uid)	用户编号
revicename	VARCHAR(30)	非空/缺省''	普通索引 (message_revicename)	接收用户
ptime	INT(10)	无符号/非空/缺省 0		发 时 间
content	TEXT	缺省''		消息内容
stutas	SMALLINT(1)	非空/缺省 0	普通索引 (message_stauts)	读取状态
补充说明	用户编号：指发 站内信的用户编号（关联发 者） 接收用户：指接收站内信的用户名称（关联接收者） 读取状态：有两个值 0 表示 读消息，1 表示已读			

表 23-10 公告表

表 名	bro_notice 用于保存公告记录，表引擎为 MyISAM 类型，字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	SMALLINT (5)	无符号/非空/自动增	主键	公告编号
title	VARCHAR(80)	非空/缺省''		公告标题
color	CHAR(6)	非空/缺省' 000000'		标题 色
starttime	INT(10)	无符号/非空/缺省 0	普通索引 (notice_starttime)	启用时间
endtime	INT(10)	无符号/非空/缺省 0	普通索引 (notice_endtime)	结束时间
content	TEXT	缺省''		公告内容
display	SMALLINT(1)	无符号/非空/缺省 1	普通索引 (notice_display)	显示状态
ord	SMALLINT(3)	无符号/非空/缺省 0	普通索引 (notice_ord)	排序编号
补充说明	公告 色：用于设置公告标题的高 显示 色，使用 16 进制 RGB 值 显示状态：有两个可用值，1 为显示，0 为不显示			

表 23-11 友情链接表

表 名	bro_flink 用于保存友情链接记录，表引擎为 MyISAM 类型，字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	SMALLINT (5)	无符号/非空/自动增	主键	链接编号
webname	VARCHAR(30)	非空/缺省''		网站名称
url	VARCHAR(60)	非空/缺省''		网站网址
logo	VARCHAR(60)	非空/缺省''		Logo 图片
rname	VARCHAR(30)	非空/缺省''		联系人名字
email	VARCHAR(50)	非空/缺省''		站长 EMAIL
dtime	INT(10)	无符号/非空/缺省 0		添加时间
msg	VARCHAR(200)	非空/缺省''		网站描述
list	SMALLINT(1)	无符号/非空/缺省 0	普通索引 (flink_list)	显示方式
audit	SMALLINT(1)	无符号/非空/缺省 0	普通索引 (flink_audit)	审核状态
ord	SMALLINT(3)	无符号/非空/缺省 0	普通索引 (flink_list)	排序编号



续表

补充说明	Logo 图片：为需要链接的网站 Logo 图片地址 显示方式：有两种显示方式，0 为显示网站名称，1 为显示网站的 Logo 图片 审核状态：有两个状态，0 值为没开通，值 1 为可以显示
------	---

表 23-12 动态信息表

表 名	bro_dynamic 用于保存用户动态信息记录，表引擎为 MyISAM 类型，字符集为 UTF-8			
列 名	数据类型	属 性	约束条件	说 明
id	INT (11)	无符号/非空/自动增	主键	动态编号
uid	INT(11)	无符号/非空/缺省 0	外键/普通索引 (dynamic_uid)	用户编号
otype	SMALLINT(1)	无符号/非空/缺省 0	普通索引 (dynamic_otype)	动态类型
ptime	INT(11)	无符号/非空/缺省 0		操作时间
pid	INT(11)	无符号/非空/缺省 0	外键/普通索引 (dynamic_pid)	评论编号
cid	INT(11)	无符号/非空/缺省 0	外键/普通索引 (dynamic_cid)	内容编号
title	VARCHAR(100)	非空/缺省''		动态标题
补充说明	<p>用户编号：标识具体用户的动态信息</p> <p>动态类型：共 5 种类型，以 1~5 的整数表示。包括：1 表示用户发表文章；2 表示用户发表评论；3 表示求用户的收藏；4 表示求用户的推荐；5 表示添加关注好友。</p> <p>评论编号：如果动态类型值为 2 时，用来保存评论的编号</p> <p>内容编号：内容编号会根据动态类型的值进行设置，例如：类型为 1、2、3、4 时内容编号为发表文章的编号，类型值为 5 时内容编号则为关注的用户编号。</p> <p>动态标题：动态标题也是根据动态类型的值进行设置，例如：类型为 1、2、3、4 时动态标题为发表文章的标题，类型值为 5 时动态标题则为关注的用户名称</p>			

注意：上述数据字典为在 MySQL 中呈现的方式，数据类型在其他数据库产品中部分需要改动。

第24章

程序设计说明书

www.brophp.com

内容管理系统（BroCMS）

文件状态：	文件标识：	LAMP 兄弟连-BroCMS-02-Program
<input type="checkbox"/> 草稿	当前版本：	2.0
<input checked="" type="checkbox"/> 正式发布	作 者：	高洛峰
<input type="checkbox"/> 正在修改	完成日期：	2011-12-15

版本历史

版本/状态	作 者	参与者	起止日期	备注
1.0	高洛峰	教学组成员	2009-11-05 2009-11-20	《细说 PHP》第 1 版
2.0	高洛峰	教学组成员	2012-04-15 2012-04-30	《细说 PHP》第 2 版

24.1 引言

根据分析，我们开发的软件是一个 CMS 系统，也就是我们常说的新 发布系统。CMS 系统就是能够自动地发布各种信息的平台，所以我们的设计思想就是围 着这个目标展开的。本系统是基于 PHP 开发的，为了便于代码重用，我们使用 BroPHP 框架进行开发，并且统一管理。

24.1.1 编写目的

本说明是 CMS 软件产品的程序设计说明，记录了系统整体实现上技术层面的设计。程序设计说明书是进行系统编码的依据，编写本文档的目的在于为程序员的编码提供详细



的说明，使程序员能根据详细设计的框图进行正确的编码。并且以需求说明作为依据，同时，该文档将作为产品实现、特性要求和控制的依据。本文档的读者对象为程序员，系统设计人员。软件开发小组的每一位参与开发成员应该阅读本说明，以清楚产品在技术方面的要求和实现策略。

24.1.2 背景

本系统是《细说 PHP》最后一部分的项目实例，可以作为读者和 LAMP 兄弟连学员的项目参考资料。除了学习项目开发参考使用之外，也可以通过本系统建设自己的网站。

- 软件系统的名称：BroCMS 网站内容管理系统 for NT(v2.0)
- 该软件系统开发项目的任务提出者：《细说 PHP》作者
- 该软件系统的用户：公司客户

24.1.3 定义

为了便于表达及避免歧义，现将本说明书中使用的专门术语的定义和外文首字母组词的原词组列出如下。

CMS：内容管理系统，Content Management System

API：插件管理（第三方开放的插件管理）

BroPHP：本系统应用的超轻量级 PHP 开发框架

24.1.4 使用技术

开发技术：PHP，BroPHP 框架，JavaScript

数据库技术：MySQL

程序控制软件：VIM

24.1.5 参考资料

- 《细说 PHP》
- 《BroPHP 框架》手册
- 软件需求说明书（本书第 28 章）
- 编码规范（本书附录 A）
- 项目进度文档
- API 接口文档
- 数据库设计说明书

24.2 系统的结构

系统总体结构设计采用了 MVC 的设计模式，以及完全使用面向对象的思想开发，并根据 BroPHP 框架的规则去规划视图、控制器和实体类及调用关系。提高代码的易维护性、易读性，增加类内部的 度、类之间调用的灵活性。

24.2.1 项目的目录结构

在需求阶段，我们将 CMS 分为了前台和后台两个应用。因为是通过 BroPHP 框架进行开发的，用以只需要为每个应用单独声明一个入口文件，分别命名为 index.php（前台入口）和 admin.php（后台应用入口），并存放到项目目录 brocms（该目录存放在 Web 服务器的根目录下）下面。在前台应用的入口文件 index.php 中指定前台的应用目录为“home”目录，在后台应用的入口文件 admin.php 中指定后台的应用目录为“admin”目录。分别访问两个入口文件 BroPHP 自动生成项目结构目录，如下所示：

-- brocms 目录	#项目根目录
-- 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）
-- home 目录	#自定义的前台项目应用目录
-- 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）

按 BroPHP 框架的要求，程序的配置文件使用项目目录下的 config.inc.php 文件，自己开发的实体类存放在 classes 目录中，公用的图片、JS 和 CSS 等资源存放在 public 目录中，用户上传的图片存放在 public/uploads/目录中。

后台应用的目录为项目目录下的 admin 目录，控制器、视图和模型分别写在 controls 目录、views 目录和 models 目录中。前台应用也是独立的，存放在项目目录中的 home 目录下，也有对应的控制器、视图和模型目录。本系统为前台应用开发了两套模板，所以在前台的 views 目录下有两个目录，用户可以进行模板风格切换。



24.2.2 模块结构

按需求分析的结果，将后台应用分为 12 个模块，前台应用分为 8 个模块。根据主要功能确定前后和后台应用中每个模块的操作和操作权限，如表 24-1 和 24-2 所示。

表 24-1 后台应用的模块操作说明

模 块	操 作	权 限
登录管理	获取登录界面、处理登录、退出、获取验证码操作	无
操作界面管理	主页、顶部、单、主区域、底部	后台登录用户
常规管理	获取系统信息、信息设置界面、设置网站信息、更新缓存	网站编辑权限
公告管理	查询公告列表、获取添加界面、添加、获取修改界面、修改、排序、删除，上传图片	网站编辑权限
友情链接管理	查询友情链接列表、获取添加界面、添加、获取修改界面、修改、排序、删除	网站编辑权限
相册管理	查询相册列表、获取添加界面、添加、获取修改界面、修改、删除	网站编辑权限
图片管理	查询图片列表、获取上传界面、添加、删除、出列表	网站编辑权限
栏目管理	查询栏目列表、获取添加界面、添加、获取修改界面、修改、排序、删除、设置栏目显示	网站编辑权限
文章管理	查询文章列表、获取添加界面、添加、获取修改界面、修改、删除、设置显示状态、设置是否允许评论	管理文章权限
幻灯片管理	查询幻灯片列表、获取添加界面、添加、获取修改界面、修改、删除	管理文章权限
用户组管理	查询用户组列表、获取添加界面、添加、获取修改界面、修改、删除	管理用户权限
用户管理	查询用户列表、获取添加界面、添加、获取修改界面、修改、删除	管理用户权限

表 24-2 前台应用的模块操作说明

模 块	操 作	权 限
首页管理	获取首页全部内容、获取公告内容	无
列表管理	获取某一个栏目的全部信息	无
内容管理	获取某一文章内容，获取文章评论信息	发表评论权限
搜索管理	搜索和显示	无
登录注册	获取注册界面、添加、登录处理、退出、唯一性检查	无
个人空间管理	显示个人空间首页平台、顶部处理、单、用户信息修改、密码设置、头像设置、设置关注、取消关注、获取关注列表、获取 列表	登录用户
消息管理	获取消息列表、写消息、显看单个消息、删除消息	登录用户并有发消息权限
动态管理	获取用户动态列表、删除动态、获取收藏或推荐、设置收藏数、设置推荐、添加评论、修改评论、删除评论、上传图片、上传 Flash 等	登录用户

24.2.3 程序结构

根据反复 论的需求说明和上面的模块规划，并结合 MVC 设计模式的思想，为每个模块声明一个控制器类来操作。先确定每个控制器和其中每个操作的名称，这样就可以大概确定程序的结构。前台和后台每个应用模块的控制器类及其说明如表 24-3 和 24-4 所示。

表 24-3 后台应用每个控制器类的结构说明

模 块	控制器类	操作方法	简要说明
登录管理	Login	index()	获取登录界面的操作
		prologin()	处理登录的操作
		logout()	用户退出的操作
		code()	获取验证码信息
操作界面管理	Index	index()	获取后台主页分 结构
		top()	获取后台主页顶部分 页面
		menu()	获取后台主页左部分 单页面
		main()	获取后台主页右部主体分 页面
		bottom()	获取后台主页底部分 页面
常规管理	Base	sysinfo()	获取网站服务器的系统信息
		baseset()	获取网站设置的表单界面
		set()	接收数据并设置网站信息
		upcache()	更新网站缓存
公告管理	Notice	index()	查询公告列表
		add()	获取添加公告表单界面
		insert()	接收公告数据并插入到数据库中
		mod()	获取修改公告表单界面
		update()	接收公告信息修改数据库原有的记录
		order()	对公告显示顺序进行排列
		del()	删除指定的公告信息
友情链接管理	Flink	index()	查询友情链接列表
		add()	获取添加友情链接表单界面
		insert()	接收友情链接数据并插入到数据库中
		mod()	获取修改友情链接表单界面
		update()	接收友情链接信息修改数据库原有的记录
		order()	对友情链接显示顺序进行排列
		del()	删除指定的友情链接信息
相册管理	Al bum	index()	查询相册列表
		add()	获取添加相册表单界面
		insert()	接收相册数据并插入到数据库中
		mod()	获取修改相册表单界面
		update()	接收相册信息修改数据库原有的记录
		del()	删除指定的相册信息
图片管理	Image	index()	查询图片列表
		add()	获取上传图片表单界面
		insert()	接收图片到服务器中并插入到数据库
		open()	显示 出式图片列表
		del()	从服务器文件和数据库中删除指定的信息



续表

模 块	控制器类	操作方法	简要说明
栏目管理	Column	index()	查询栏目列表
		add()	获取添加栏目表单界面
		insert()	接收栏目数据并插入到数据库中
		mod()	获取修改栏目表单界面
		update()	接收栏目信息修改数据库原有的记录
		del()	删除指定的栏目信息
		dis()	设置栏目的显示状态
		order()	对栏目显示顺序进行排列
文章管理	Article	index()	查询文章列表
		add()	获取添加文章表单界面
		insert()	接收文章数据并插入到数据库中
		mod()	获取修改文章表单界面
		update()	接收文章信息修改数据库原有的记录
		del()	删除指定的文章信息
		fpro()	批量处理文章的各种状态
		status()	单个文章状态的设置
幻灯片管理	Play	index()	查询幻灯片列表
		add()	获取添加幻灯片表单界面
		insert()	接收幻灯片数据并插入到数据库中
		mod()	获取修改幻灯片表单界面
		update()	接收幻灯片信息修改数据库原有的记录
		order()	对幻灯片显示顺序进行排列
		del()	删除指定的幻灯片信息
用户组管理	Group	index()	查询用户组列表
		add()	获取添加用户组表单界面
		insert()	接收用户组数据并插入到数据库中
		mod()	获取修改用户组表单界面
		update()	接收用户组信息修改数据库原有的记录
		del()	删除指定的用户组信息
用户管理	User	index()	查询用户列表
		add()	获取添加用户表单界面
		insert()	接收用户数据并插入到数据库中
		mod()	获取修改用户表单界面
		update()	接收用户信息修改数据库原有的记录
		del()	删除指定的用户信息
全局管理	Common	init()	公用操作，用于处理登录和控制权限操作
		mess()	处理用户提示消息
		upimage()	处理文章和公告的图片上传
		upflash()	处理文章和公告的 Flash 上传

表 24-4 前台应用每个控制器类的结构说明

模 块	控制器类	操作方法	简要说明
首页管理	Index	index()	输出网站首页信息
		notice()	输出网站公告页面信息
列表管理	List	index()	以分页格式输出一个栏目记录信息
文章内容管理	Article	index()	输出用户请求的文章详细信息
		comment()	以分页显示当前文章的评论信息
搜索管理	Search	index()	处理用户搜索并显示搜索结果
登录注册	Login	index()	提供用户登录表单
		logout()	处理用户退出
		register()	提供用户注册表单
		insert()	将用户注册信息插入到数据库中
		unique()	检查用户名是否唯一
		code()	提供用户注册的验证码
		vcode()	验证用户输入的验证码
个人空间管理	User	index()	提供个人空间的操作界面
		top()	个人空间顶部信息的操作
		menu()	个人空间左部 单的操作
		set()	设置用户信息
		pset()	用户密码设置
		tset()	用户头像设置
		follow()	设置用户关注
		delfollow()	取消用户关注
		allfollowed()	获取 列表
		allfollowing()	获取关注列表
消息管理	Message	index()	显示用户的全部消息列表
		write()	发用户消息
		view()	查看某个消息内容
		del()	删除指定的消息
动态管理	Dynamic	index()	显示一个用户的所有动态信息
		Del()	删除指定的用户动态
		Gts()	获取收藏或推荐数
		Collection()	设置收藏数
		Recommnd()	设置推荐
		Addc()	添加用户评论
		Modc()	修改用户评论
		Delc()	删除用户评论
		Add()	获取文章添加表单
		Insert()	向数据表中添加新的文章
		Mod()	获取文章修改表单
		Update()	更新已有一条文章记录



续表

模 块	控制器类	操作方法	简要说明
全局管理	Common	Upimage()	处理添加文章时的图片上传
		Upflash()	处理添加文章时的 Flash 动画上传
		init()	公用操作，用于处理登录和控制权限操作

按 BroPHP 框架的规则，将前台的控制器类声明在项目目录下的 home/controls 中，后台控制器类声明在 admin/controls 目录中。并且控制器类所在的文件名要以“控制器类名.class.php”格式命名，文件名称一小写。前台和后台两个应用控制器类之间的继承关系如图 24-1 和图 24-2 所示。

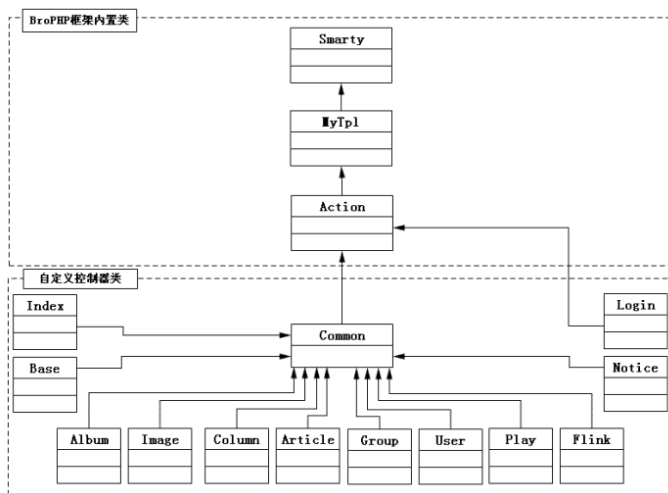


图 24-1 BroCMS 后台控制器类图继承关系

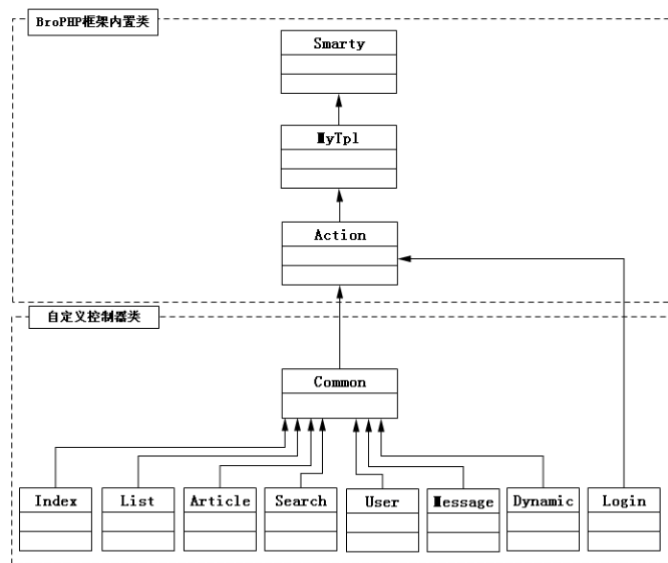


图 24-2 BroCMS 前台控制器类图继承关系

24.3 用户管理模块设计说明

从本节开始，将一个给出各个模块的设计考虑，但限于篇幅，本书只给出一个用户模块的设计参考。

24.3.1 功能

网站在发展过程中需要用户互动来促进网站发展，另一方面也需要防止用户对网站信息的越权访问或随意发布各类信息。因此需要针对不同的服务对象（非注册用户、注册用户、不同的注册用户类型），根据其需求开设不同的信息栏目、提供不同范围的信息服务。根据网站的管理需要设定不同权限的管理员，以方便协同管理网站。当用户在网站中注册为注册会员，则相当于在网站中有了一个通行证，会员可用于别属于自己的信息、访问或发布权限允许内的信息。用户模块的具体操作如下所示：

- 1 添加用户
除了新用户自己注册以外，还可以通过用户管理系统提供的表单界面添加新的用户。
- 2 查询用户
可以根据多种条件筛选出需要处理的用户列表，并通过分页管理多条数据。
- 3 编辑用户
和添加用户类似，对已经注册的用户进行编辑修改。
- 4 删除用户
可以删除一些非法的用户记录，可以单条删除，也可以通过复选框选择多项一起删除。

24.3.2 流程逻辑

用户模块操作流程如图 24-3 所示。

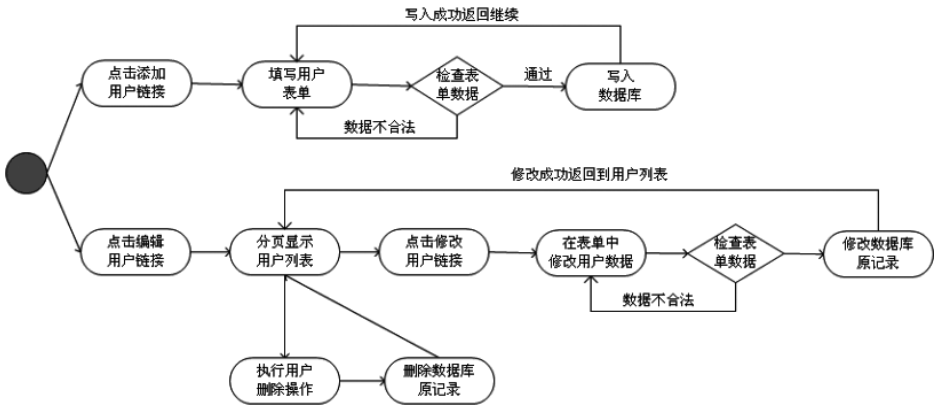
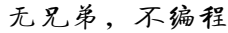
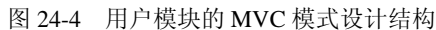


图 24-3 用户模块操作流程



根据 BroPHP 框架的规则，在设计用户管理模块时，需要在指定的位置一个控制器类 (User)。并在控制器类中声明 6 个操作方法，包括用户列表操作 `index()`、获取添加界面的操作 `add()`、数据入库的操作 `insert()`、获取修改页面的操作 `mod()`、修改数据库记录的操作 `update()` 和删除操作 `del()`。其中有 3 个操作方法需要通过模板显示数据，并在一些操作中需要通过模型类及实体类完成一些业务。用户管理模块的 MVC 模式结构如图 24-4 所示。



本模块只需要对一个用户表操作即可，所有数据操作都通过 **Model** 类来完成。数据表结构详见《数据库设计数明书》（第 29 章）。

- 在控制器类、模型类和实体类的首部添加注释说明；
- 在类中的每个方法上方添加注释说明；
- 对各变量的功能、范围、默认条件等加上注释；
- 对使用的逻辑算法加上注释。

在添加和修改数据时用户名称、用户密码和电子邮箱必须填写。管理员用户不能删除，否则将不能登录系统，用户被删除时，他的文章及评论等所有信息也会一同删除。

24.3.7 测试计划

通过测试达到以下目标：

- 测试已实现的模块是否达到设计的要求，各个操作点是否以实现。
- 产品规定的操作和运行是否一定。
- Bug 数和缺陷率控制在可接受的范围之内。
- 业务流程及数据流从软件中的一个模块流到另一个模块的过程中的正确性。

24.3.8 尚未解决的问题

无（在本程序的设计中 解决而设计者认为在软件完成之前应解决的问题）。

24.3.9 获取添加用户的界面操作 add()

1. 程序描述

如果需要向数据库中添加一条用户记录，就需要先为管理员提供一个添加表单。操作用户先通过管理平台的 单项单击“添加用户”链接，就会通过链接中的 URL 访问到用户模块的 add()操作中。在该操作中，需要从模型中获取用户组的列表发 给对应的模板中显示，并输出模板文件（add.tpl），提供给操作用户一个添加用户界面。

2. 输入项

本操作的访问 URL 为：/brocms/admin.php/user/add。

3. 输出项

用户请求时并不需要提供其他的参数输入，但需要从用户组模型中获取一个用户组列表。需要请求 Group 模型中的“formselect("gid", 2)”，第一个参数为用户组表单列表的名称，第二个参数是默认从第二个开始显示。将从 Group 模型中获取的用户组列表通过变量名“select”发 到模板中，输出添加用户的模板。

4. 算法

第一步：提示用户添加的规则

第二步：从用户组模型中获取用户组下拉列表分配到模板中

第三步：加载并输出添加模板 add.tpl

5. 模板设计

需要为该操作声明一个添加用户的模板文件，在模板中只需要声明一个 HTML 表单。在模板设计时需要用到的一些信息如下所示。

模板文件：声明在视图目录中，并在用户模块目录（user）下，文件名称为 add.tpl。

子模板：包含公用的头部（header.tpl）和尾部（footer.tpl）模板文件。

表单提交位置：当前模块的 insert()操作中。



提交方法：POST。

表单项的设计：如表 24-5 所示。

表 24-5 添加用户表单内容项

表 单 项	类 型	名 称	说 明
用户组	checkbox	gid	提供用户组下拉列表
用户名称	text	username	可以使用中文，但 止除[@ .]以外的特 符号
用户密码	password	userpwd	初使密码为 brophp
确认密码	password	repwd	
电子邮箱	text	email	请正确添写你的电子邮件地址
用户性别	radio	sex	男、女、保密
是否 用	checkbox	disable	用该用户

说明：如果需要连续录入多个用户，可以添加一下用户组的“记住选项”。

24.3.10 用户数据入库的操作 insert()

1. 程序描述

操作者在添加用户表单中录入一个用户信息后，将用户所有录入的数据以 POST 方法传递到 insert() 操作中，处理数据并插入到数据表 user 中。

2. 输入项

本操作的访问 URL 为：/brocms/admin.php/user/insert。

本操作的输入项是用户在表单中录入的全部信息，包括用户组编号、用户名称、用户密码确认密码、电子邮箱、用户性别和 用状态。以 HTTP 的 POST 方法传递到本操作中，所以全部保存在超全局数据 \$_POST 中，其中需要对用户密码进行加密处理（使用 md5() 函数）。数组 \$_POST 的格式如下所示：

```
$_POST=array(
    'username'=>'user',           //用户名
    'userpwd'=>'123456',          //用户密码
    'repwd'=>'123456',            //确认密码
    'sex'=>'1',                    //用户性别
    'disable'=>'0',                // 用状态
    'email'=>'user@brophp.com',    //用户电子邮箱
    'sub'=>'添加用户'             //提交按钮
);
```

3. 输出项

本操作中的输出项是要插入到数据库中的数据，是通过验证并处理过的数据。

4. 算法

第一步：处理用户提交的数据；

第二步：使用 BroPHP 框架中的自动验证方式进行数据验证；

第三步：如果验证通过，则将数据插入到数据库 user 表中，插入成功后再返回到添加表单界面，让操作者可以继续添加下一个用户；

第四步：如果验证数据失败，也是返回到添加表单去重新添加数据，但要保留上一次录入过的数据。

第五步：数据添加成功或是失败都要提示操作者。

24.3.11 查询用户列表操作 index()

1. 程序描述

操作者通过“编辑用户”链接进入到用户的列表中，用户的记录列表全部从数据库中获取并以分页形式显示。操作者可以通过用户组的下拉列表选择对应某个用户组下面的全部用户记录，也可以通过用户名称的模糊搜索快速定位到需要的查找的用户，并且通过分页切换页面时也要保持查询条件。

2. 输入项

本操作的访问 URL 为：/brocms/admin.php/user/index。

(1) 用户组编号通过 GET 方法传递，格式为\$_GET['gid']。

(2) 搜索的用户名通过 GET 方法传递，格式为\$_GET['search']。

3. 输出项

(1) 用户列表，以一个变量形式分配到模板中，类型为一个二维数组，变量名为\$users。

(2) 分页内容，也是以一个变量形式分配到模块中，类型为一个字符串，变量名为\$page。

4. 算法

第一步：提示用户操作方式；

第二步：当操作者从下拉列表中选择一个用户组后，用户的记录列表则为当前用户组下面的所有用户，所以需要按组编号组合一个查询的 WHERE 条件，并要根据查询条件组合出分页需要的传递参数；

第三步：同样在搜索用户时也需要按用户称组合一个查询的 WHERE 条件和分页参数，并且要保持在特定用户组下的进行搜索，只查询当前用户组下的符合条件的用户记录；

第四步：如果没有指定用户组或搜索条件为空，则查询全部记录；

第五步：加载输出 index.tpl 模板显示用户记录。

5. 模板设计

在用户列表模块中，需要 用户数组列表及其他一些操作链接按钮，使用 DIV+CSS 布局页面模板。在模板设计时需要用到的一些信息如下所示。

模板文件：声明在视图目录中，并在用户模块目录（user）下，模板文件名称和操作名称相同，为 index.tpl。

子模板：包含公用的头部（header.tpl）和尾部（footer.tpl）模板文件。



表单提交位置：其中删除会用到表单，提交到当前模块的 `del()` 操作中，提交方法使用 `post`。

模板中用到的变量：主要有两个，一个是保存列表数据的二维数组 `$users`，另一个是保存分页字符串变量 `$fpage`。其中 `$users` 的格式如下：

```
$user = array (
    [0]=>array(
        'id'=>'12',                //用户编号
        'username'=>'user',        //用户名
        'regtime'=>'1312323323',   //添加时间
        'email'=>'user@brophp.com', //用户电子邮箱
        'disable'=>'0',            // 用状态
    ),
    [1]=>array(
        )
    id,username,regtime,email,disable
    .....
);
```

24.3.12 获取修改用户的界面操作 `mod()`

1. 程序描述

如果需要修改数据库中一条用户记录，和添加用户相似，需要先为操作者提供一个修改表单。操作者通过单击用户列表中的“修改”链接，就会通过链接中的 URL 访问到用户模块的 `mod()` 操作中。在该操作中加载一个修改模板并输出，为操作者提供一个修改用户界面。

2. 输入项

本操作的访问 URL 为： `/brocms/admin.php/user/mod`。

(1) 用户编号：通过 GET 方法传递，格式为 `$_GET['id']`。

3. 输出项

(1) 一条用户数据，以一个变量形式分配到模板中，类型为一个一维数组，变量名为 `$users`。

(2) 用户组列表，以一个变量形式分配到模板中输出，类型为一个字符串，变量名为 `$select`。

4. 算法

第一步：提示用户修改的规则；

第二步：通过 URL 的 GET 方式传递进来要修改的用户记录 ID，除了和获取添加界面操作一样，需要获取用户组的下列表分配到模板中使用外，并按用户 ID 从数据库的用户表中获取这个用户的全部信息数据，分配到 `mod.tpl` 模板中；

第三步：在表单中对应的位置回填这些数据；

第四步：加载并输出修改模板 mod.tpl。

5. 模板设计

需要为该操作声明一个修改用户的模板文件，在模板中只需要声明一个 HTML 表单，并将控制器中分配的用户信息对应地回填到每个表单项中。在模板设计时需要用到的一些信息如下所示。

模板文件：声明在视图目录中，并在用户模块目录（user）下，文件名称为 mod.tpl。

子模板：包含公用的头部（header.tpl）和尾部（footer.tpl）模板文件。

表单提交位置：当前模块的 update()操作中。

提交方法：POST。

表单项的设计：如表 24-6 所示。

表 24-6 修改用户表单设计项

表 单 项	类 型	名 称	默 认 值	说 明
用户编号	hidden	id	\$user.id	当前用户的编号
用户组	checkbox	gid	控制器中指定	提供用户组下拉列表
用户名称	text	username	\$user.username	可以使用中文，但 止除[@][_]以外的特 符号
用户密码	password	userpwd	无	
确认密码	password	repwd	无	
电子邮箱	text	email	\$user.email	请正确添写你的电子邮件地址
用户性别	radio	sex	\$user.sex	男、女、保密
是否 用	checkbox	disable	\$user.disable	用该用户

24.3.13 用户数据修改的操作 update()

1. 程序描述

操作者在用户修改表单中修改一些信息以后，将新修改的数据以 POST 方法传递到 update()操作中，处理数据后并修改数据表 user 中当前的记录。

2. 输入项

本操作的访问 URL 为：/brocms/admin.php/user/update。

本操作的输入项是用户在表单中录入的全部信息，以 HTTP 的 POST 方法传递到本操作中，所以全部保存在超全局数据\$_POST 中，其中需要对用户密码进行加密处理（使用 md5()函数）。数组\$_POST 的格式如下所示：

```
$_POST=array(
    'id'=>'user',           //用户编号，作为修改的条件
    'username'=>'user',     //用户名
    'userpwd'=>'123456',    //用户密码
    'repwd'=>'123456',     //确认密码
    'sex'=>'1',             //用户性别
    'disable'=>'0',         // 用状态
    'email'=>'user@brophp.com', //用户电子邮箱
```



```
);  
    'sub'=>'添加用户' //提交按钮
```

3. 输出项

本操作中的输出项是要修改到数据库中的数据，是通过验证并处理过的数据。

4. 算法

第一步：处理用户提交的数据；

第二步：使用 BroPHP 框架中的自动验证方式进行数据验证；

第三步：如果验证通过，则修改数据表（user）中这条记录，并返回到用户记录列表页面；

第四步：如果修改失败，则返回到修改表单界面重新进行修改，还要将表单中的数据值还原；

第五步：数据修改成功或是失败都要提示操作者。

24.3.14 删除用户操作 del()

1. 程序描述

操作者可以通过每条用户记录后面提供的“删除”链接，单击删除一条用户记录，也可以通过每条记录前面的复选框选择多条记录删除。删除操作成功或是失败都要再返回到用户列表中。

2. 输入项

本操作的访问 URL 为：/brocms/admin.php/user/del。

输入为两种类型的用户编号：

(1) 一种是单个用户编号，通过 GET 方法传递格式为\$_GET['id']。

(2) 另一种是通过 POST 方法传递多个用户编号，格式为\$_POST['id']。

3. 输出项

无

4. 算法

第一步：提示用户删除的规则。

第二步：组合用户的删除条件，如果是 GET 方式传递的用户 ID，则删除一条记录。如果是通过 POST 方式传递的数据，则要删除多条记录。并且还要组合删除后的跳转位置。

第三步：删除指定的用户和用户的全部资源（包括用户发布的文章、评论、短消息和用户动态等）。

第四步：返回到删除操作前的用户列表。

第五步：处理删除成功或失败的提示状态。