第27章　建立用户身份验证机制和个性化设置

在这个项目中，我们将讨论如何让用户在我们的站点注册。当完成注册之后，我们能跟踪他们所感兴趣的事物并向他们显示适当的内容。这个过程叫做用户个性化设置。

这个项目允许用户建立一组网页书签，并根据他们以前的操作向他们建议其他可能感兴趣的链接。更常见的是，用户个性化设置可用于所有基于Web的应用程序，以用户希望的格式显示他们感兴趣的内容。

在这个项目和接下来的其他项目中，我们将开始了解一组需求，这个需求类似于从客户那里获得的需求。我们会将这些需求开发为一组解决方案组件，并建立一个将这些组件联系到一起的设计，然后再应用每个组件。

在这个项目中，我们将实现如下功能：

- 用户登录和验证用户
- 管理密码
- 记录用户的个人喜好
- 个性化内容
- 基于已有的用户信息为用户推荐他们可能感兴趣的内容

27.1　解决方案的组成

对于这个项目，我们希望为在线书签系统建立一个原型，称之为PHPbookmark，它与在Backflip网站上可以访问到的系统类似（但在功能上更有限）：http://backflip.com。

我们的系统应该能够让用户登录进来，保存他们的书签，并基于他们的个人喜好推荐给他们可能喜欢的其他站点。

这些解决方案需求分为3个主要部分。

首先，需要识别每个用户。我们应该有验证他们身份的方法。

其次，需要保存单个用户的书签。用户应该能够添加和删除书签。

再次，需要根据对他们的了解，向用户建议他们可能感兴趣的站点。

现在，我们已经对项目有了基本了解，可以开始设计解决方案及其组件了。下面，我们开始了解上述3个需求各自的解决方案。

27.1.1　用户识别和个性化设置

用户身份验证有许多可以选择的办法，这些在本书的其他部分已经讨论过。因为我们希望将用户和一些个性化信息联系起来，所以就要将用户的登录名和密码保存在一个MySQL数据库中，验证的时候用它做比较。

如果要让用户以用户名和密码登录，需要如下组件。

- 用户能够注册一个用户名和密码。我们需要限制用户名和密码的长度和格式。为安全起见，应该将密码加密之后再保存。
- 在注册过程中，用户应该能够看到他们提供的详细信息。
- 用户完成网站访问之后能够登出。从隐私角度考虑看，如果用户是使用自己家中的PC，这并不是很重要，但是在共用的PC上这就相当重要了。
- 网站能够检测用户是否登录，并为登录的用户访问数据。
- 为了安全起见，用户可以修改密码。
- 用户应该能够在不需要开发者帮助的前提下重置他们的密码。一个常用的方法是将密码发送到用户注册时提供的邮箱。这意味着要在注册的时候保存他们的邮箱地址。因为密码是以加密的形式进行保存的，而且其他人不能够解码。因此实际上只能为用户设置一个新密码，并将它发给用户。

我们要为所有这些功能编写函数。这些函数大多数是可重用的，或者可以从其他项目中拿过来，只要经过较小的修改就可以重用。

27.1.2 保存书签

要保存一个用户的书签，需要在MySQL数据库中开辟一定的空间。我们必须实现如下功能：

- 用户应该能够取回、浏览他们的书签
- 用户应该能够增添加新的书签，我们要检查这些URL是否有效
- 用户应该能够删除书签

我们要为以上每个功能编写函数。

27.1.3 推荐书签

我们可以采取不同的方式向用户推荐书签，可以推荐最流行的或者在某个主题上最流行的站点。对于这个项目，我们要实现一个“相似意向”建议系统，该系统可以查找与登录用户具有相同书签的用户，并将他们的其他书签推荐给用户。为了避免推荐任何个人的书签，我们只推荐几个用户同时拥有的书签。

我们同样需要为这个功能编写一个函数。

27.2 解决方案概述

在纸上绘制草图之后，我们提出了系统流程图，如图27-1所示。

我们将为该表中每一部分创建一个模块；其中一些模块需要一段脚本，而另外一些可能需要两脚本段。我们还要建立函数库，函数库作用如下：

- 用户身份验证
- 书签保存与检索
- 数据验证
- 数据库连接
- 输出到浏览器

我们将把所有HTML输出都限制到该函数库，确保在网站里所有可视的外观是和谐一致的（这就是函数API方法用来将逻辑和内容分离的功能）。

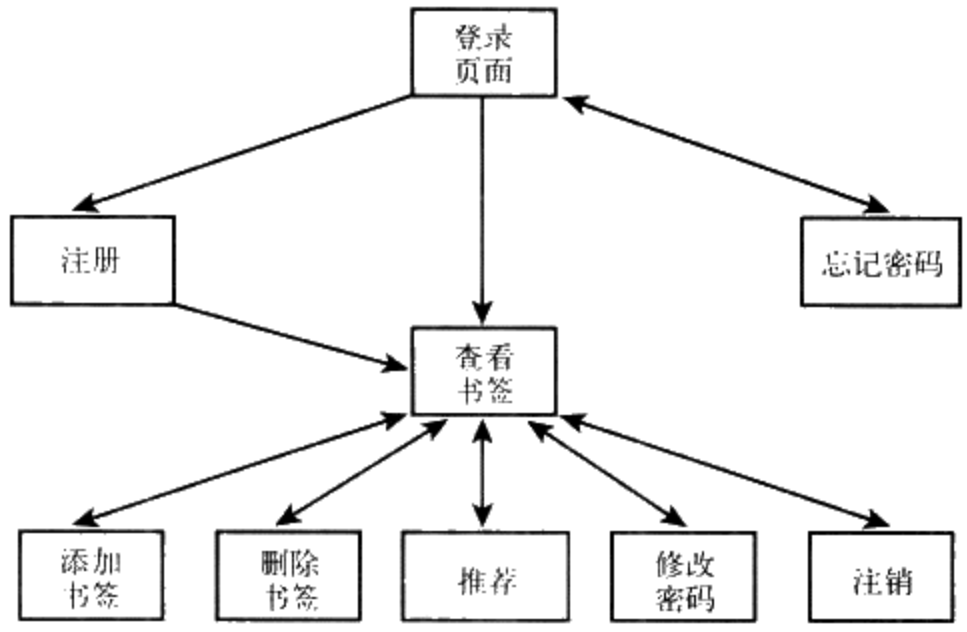

图27-1　该图显示了PHPbookmark系统的各种工作流程

我们还需要建立一个系统后台数据库。

我们只介绍这个解决方法的部分详细内容，该应用程序的代码都可以在随书附带文件的chapter27目录下找到。该目录下所包含文件的摘要如表27-1所示。

表27-1　PHPbookmark应用程序中的文件

文 件 名	描　述
bookmarks.sql	创建PHPbookmark数据库的SQL语句
login.php	包含系统登录表单的页面
register_form.php	系统中用户注册表单
register_new.php	处理新注册信息的脚本
forgot_form.php	用户忘记密码后需要填写的表单
forgot_passwd.php	重新设置遗忘密码的脚本
member.php	用户的主页面，包含该用户所有的当前书签
add_bm_form.php	添加书签的表单
add_bms.php	将书签真正添加到数据库中的脚本
delete_bms.php	从用户的书签列表中删除选定书签的脚本
recommend.php	基于用户以前的操作，推荐用户可能感兴趣的书签
change_passwd_form.php	用户修改密码时要填的表单
change_passwd.php	修改数据库中用户密码的表单
logout.php	将用户注销的脚本
bookmark_fns.php	应用程序的包含文件集合
data_valid_fns.php	确认用户输入数据有效的函数
db_fns.php	连接数据库的函数
user_auth_fns.php	用户身份验证的函数
url_fns.php	增加和删除书签的函数
output_fns.php	以HTML形式格式化输出的函数
bookmark.gif	PHPbookmark的logo图标

首先，我们将讨论应用程序中MySQL数据库的实现，因为它实际上是实现其他功能所必需的。

然后，我们将以编写代码的顺序详细研究代码。从首页开始，到用户验证，到书签保存和检索，最后书签建议。这个顺序是很有逻辑性的——正好是一个解决依赖性的问题，也就是依次创建下一个模块所需要的模块。

提示　要该应用能够正常工作，需要一个支持JavaScript的浏览器来查看应用程序。

27.3　实现数据库

对于PHPbookmark数据库来说，只需要一个非常简单的数据库模式。在程序中，我们要保存用户名和他们的邮箱地址以及用户密码，还要保存书签的URL。一个用户可能有许多书签，许多用户也可能注册了同一个书签。因此，我们需要两个表，user表和bookmark表，如图27-2所示。

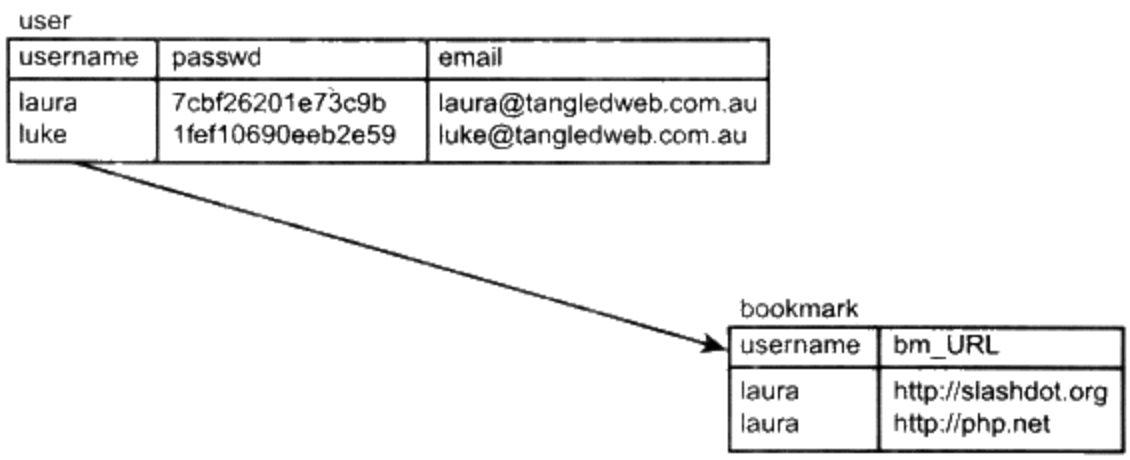

图27-2　PHPbookmark系统的数据库模式

user表用来保存用户的username（该表单的主键）、password和email。bookmark表用来保存username和bm_URL对。表中的username指向user表格中的username 。

创建本数据库的SQL脚本，以及创建一个用户用来通过网络连接到数据库的脚本，如程序清单27-1所示。注意如果要将该代码应用于自己的系统，需要进行一定的修改：改变用户密码，使其更安全！

程序清单27-1　bookmarks.sql——建立Bookmark数据库的SQL文件

```
create database bookmarks;
use bookmarks;

create table user (
  username varchar(16) not null primary key,
  passwd char(40) not null,
  email varchar(100) not null
);

create table bookmark (
```

```
  username varchar(16) not null,
  bm_URL varchar(255) not null,
  index (username),
  index (bm_URL),
  primary key(username, bm_URL)
);

grant select, insert, update, delete
on bookmarks.*
to bm_user@localhost identified by 'password';
```

可以以MySQL root用户的身份运行以上这些命令在系统中建立数据库。可以在系统的命令行下输入如下所示命令：

```
mysql -u root -p < bookmarks.sql
```

系统将要求输入root用户的密码。

数据库建好之后，我们可以继续网站建设的其他基本步骤。

27.4 实现基本的网站

我们需要创建的第一页称为login.php，因为它向用户提供了登录系统的机会。该页面的代码如程序清单27-2所示。

程序清单27-2　login.php——PHPbookmark系统的首页

```
<?php
 require_once('bookmark_fns.php');
 do_html_header('');

 display_site_info();
 display_login_form();

 do_html_footer();
?>
```

这段代码看起来非常简单，它大部分是调用我们将要为这个应用程序所创建的函数API。

稍后，我们就可看到这些函数的详细信息。从这些文件我们可以看出已经包含了一个文件（也就是包含了一些函数），调用一些函数绘制一个HTML标题，显示一些内容，然后再绘制一个HTML页脚。

以上脚本的输出如图27-3所示。

系统中的函数都包含在bookmark_fns.php文件中，如程序清单27-3所示。

程序清单27-3　bookmark_fns.php——包含Bookmark应用程序函数的文件

```
<?php
  // We can include this file in all our files
  // this way, every file will contain all our functions and exceptions
```

```
    require_once('data_valid_fns.php');
    require_once('db_fns.php');
    require_once('user_auth_fns.php');
    require_once('output_fns.php');
    require_once('url_fns.php');
?>
```

可以看到，该文件就是本应用程序中将要使用到的5个其他包含文件的“容器”。设计成这样的结构是因为函数在逻辑上可以分为几组。这些组中的一些函数可能对其他项目有用处，因此我们将每个函数组保存不同的文件，这样就可以在再需要它们的时候知道到哪里去查找。创建bookmark_fns.php文件是因为在大部分脚本里都要用到这5个函数文件。在每个脚本里包含这一个文件而不是使用5个require语句，这样会更容易一些。

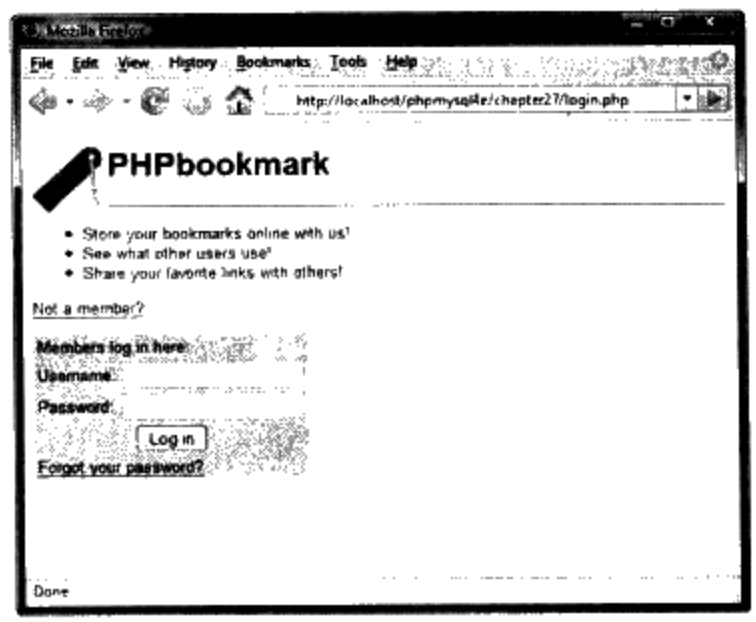

图27-3　PHPbookmark系统的首页由login.php中的绘制HTML的函数创建

在这个例子中，我们使用的函数来自output_fns.php文件。这些函数相当直观，它们的输出都是非常明了的HTML。该文件包含了我们在login.php中使用的4个函数，即do_html_header()、display_site_info()、display_login_form()和do_html_footer()等。

本书不再深入讨论所有这些函数，只举例说明其中一个函数。do_html_header()函数源代码如程序清单27-4所示。

程序清单27-4　output_fns.php文件中的函数do_html_header()——该函数输出在本应用程序的每个页面中都将出现的标准标题

```
function do_html_header($title) {
  // print an HTML header
?>
  <html>
  <head>
    <title><?php echo $title;?></title>
    <style>
      body { font-family: Arial, Helvetica, sans-serif; font-size: 13px }
      li, td { font-family: Arial, Helvetica, sans-serif; font-size: 13px }
      hr { color: #3333cc; width=300px; text-align:left}
      a { color: #000000 }
    </style>
  </head>
  <body>
  <img src="bookmark.gif" alt="PHPbookmark logo" border="0"
       align="left" valign="bottom" height="55" width="57" />
```

```
    <h1>PHPbookmark</h1>
    <hr />
<?php
    if($title) {
      do_html_heading($title);
    }
}
```

可以看到，函数中唯一的逻辑是在页面中添加适当的标题。login.php中使用的其他函数与该函数类似。display_site_info()函数添加一些关于网站的文本；display_login_form()显示如图27-3所示的灰色表单；do_html_footer()为页面添加一个标准的HTML页脚。

关于从主逻辑流中分离或删除HTML的意义，我们已经在第25章“在大型项目中使用PHP和MySQL”详细讨论过。在本章中，我们将介绍使用函数API方法。

在图27-3中，可以看到该页面有3个选择——用户可以注册、登录（如果已经注册）和修改密码（如果忘记了密码）。要实现这些模块，必须先了解下一节的用户身份验证。

27.5 实现用户身份验证

用户身份验证模块包括4个主要的元素：用户注册、登录和登出、修改密码以及重置密码。我们将按顺序详细讨论每个元素。

27.5.1 注册用户

要注册一个用户，需要通过一个表单获得用户的详细信息，并且将这些信息保存到数据库中。

当用户点击login.php页面上的“Not a member?”链接时，就会出现一个由register_form.php产生的注册表单。该脚本如程序清单27-5所示。

程序清单27-5　register_form.php——该表单让用户在PHPbookmark系统中注册

```
<?php
 require_once('bookmark_fns.php');
 do_html_header('User Registration');

 display_registration_form();

 do_html_footer();
?>
```

可以看到，该页非常简单，只调用了来自output_fns.php的输出库中的函数。该脚本输出如图27-4所示。

该页中的灰色表单是由display_registration_form()函数输出的，该函数也包含在output_fns.php中。当用户点击“Register”按钮时，register_new.php脚本将运行。该脚本如程序清单27-6所示。

程序清单27-6 register_new.php——该脚本验证新用户的数据，并将其存入数据库

```
<?php
  // include function files for this application
  require_once('bookmark_fns.php');

  //create short variable names
  $email=$_POST[ email'];
  $username=$_POST['username'];
  $passwd=$_POST['passwd ];
  $passwd2=$_POST['passwd2'];
  // start session which may be needed later
  // start it now because it must go before headers
  session_start();
  try {
    // check forms filled in
    if (!filled_out($_POST)) {
      throw new Exception('You have not filled the form out correctly -
          please go back and try again.');
    }
    // email address not valid
    if (!valid_email($email)) {
      throw new Exception('That is not a valid email address.
          Please go back and try again.');
    }

    // passwords not the same
    if ($passwd != $passwd2) {
      throw new Exception('The passwords you entered do not match -
          please go back and try again.');
    }

    // check password length is ok
    // ok if username truncates, but passwords will get
    // munged if they are too long.
    if ((strlen($passwd) < 6) || (strlen($passwd) > 16)) {
      throw new Exception('Your password must be between 6 and 16 characters.
          Please go back and try again.');
    }

    // attempt to register
    // this function can also throw an exception
    register($username, $email, $passwd);
    // register session variable
    $_SESSION['valid_user'] = $username;

    // provide link to members page
    do_html_header('Registration successful');
```

```
        echo 'Your registration was successful. Go to the members page to start
              setting up your bookmarks! ';
        do_html_url( member.php ,  Go to members page );

        /. end page
       (do_html_footer();
      }
      catch (Exception $e) {
         do_html_header('Problem:');
         echo $e->getMessage();
         do_html_footer();
         exit;
      }
?>
```

这是该项目中我们看到的第一个比较复杂的脚本。该脚本的起始部分包含了应用程序函数文件并启动了一个会话（用户注册的时候，我们将他的用户名创建为会话变量，正如我们在第23章中“在PHP中使用会话控制”所介绍的那样）。

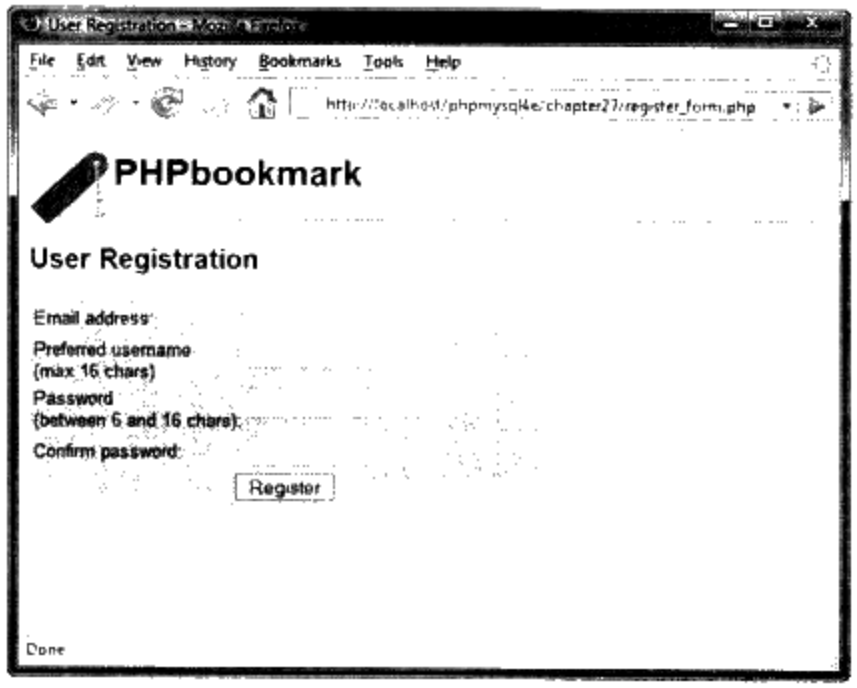

图27-4　注册表单获取了数据库需要的用户详细信息。密码要求输入两次，以防输入错误

脚本的主体有一个try语句块，因为需要检查许多条件。如果任何一个条件失败，执行将进入catch语句块，我们将在稍后详细介绍。

接下来验证用户输入的数据。在此过程中，我们要测试许多条件，如下所示。

■ 检查表单是否完全填写。调用filled_out()函数测试，如下所示：

```
if (!filled_out($_POST))
```

这个函数是我们自己编写的函数之一。它位于data_valid_fns.php函数库。稍后，我们将详细介绍这个函数。

■ 检查邮件地址是否有效。测试如下所示：

```
if (valid_email($email))
```

这个函数也是我们自己编写的函数之一。它也位于data_valid_fns.php函数库。

■ 验证用户两次输入的密码是否一致，如下所示：

```
if ($passwd != $passwd2)
```

■ 验证密码长度是否在规定范围之内，如下所示：

```
if ((strlen($passwd) < 6)
```

和

```
if ((strlen($passwd) > 16)
```

在我们的例子中，密码至少为6个字符，这样以防别人容易猜出，同时用户名要少于17个字符，以适合存放到数据库中。请注意，密码的最大长度并不局限于此，因为它是以SHA1哈希值保存的，因此通常是40个字符，而不受密码长度的限制。

本例用到的数据验证函数filled_out()和valid_email()，分别如程序清单27-7和程序清单27-8所示。

程序清单27-7 data_valid_fns.php文件中的filled_out()函数——该函数检查表单是否完全填写

```
function filled_out($form_vars) {
  // test that each variable has a value
  foreach ($form_vars as $key => $value) {
    if ((!isset($key)) || ($value == '')) {
      return false;
    }
  }
  return true;
}
```

程序清单27-8 data_valid_fns.php文件中的valid_email()函数——该函数检查邮件地址是否有效

```
function valid_email($address) {
  // check an email address is possibly valid
  if (preg_match("^[a-zA-Z0-9_\.\-]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$", $address)) {
    return true;
  } else {
    return false;
  }
}
```

函数filled_out()需要传递一个数组变量，通常是$_POST或$_GET变量数组。它检查表单是否完全填写，如果完全填写则返回true，否则返回false。

valid_email()函数使用了在第4章“字符串操作与正则表达式”中介绍的正则表达式来验证邮件地址。如果地址是有效的，就返回true，否则返回false。

在验证了输入数据之后，我们就可以尝试注册该用户了。如果回头看看程序清单27-6，会

发现我们是按如下方式实现的：

```
register($username, $email, $passwd);
// register session variable
$_SESSION[ valid_user'] = $username;

// provide link to members page
do_html_header('Registration successful');
echo 'Your registration was successful. Go to the members page to start
      setting up your bookmarks!';
do_html_url('member.php', 'Go to members page');

// end page
do_html_footer();
```

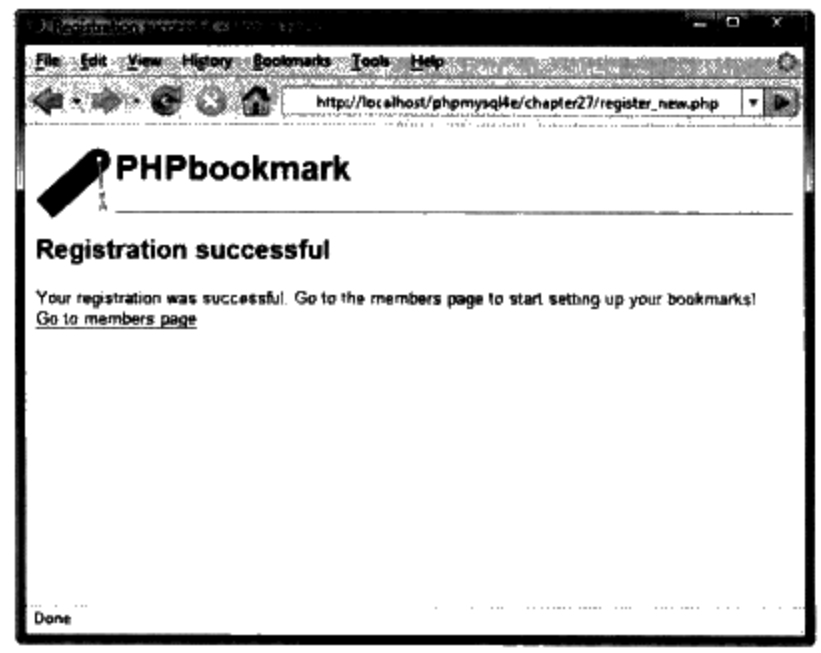

图27-5　注册成功，用户可以访问成员页

可以看到，我们使用用户输入的用户名、邮件地址和密码作为参数调用了register()函数。如果函数执行成功，我们就将用户名注册为会话变量，并为用户提供一个指向成员主页的链接（如果函数执行失败，它将抛出一个可以在catch语句块中捕获的异常）。其输出如图27-5所示。

register()函数包含在user_auth_fns.php函数库中。函数代码如程序清单27-9所示。

程序清单27-9　user_auth_fns.php文件中的register()函数——该函数试图将用户信息提交到数据库

```
function register($username, $email, $password) {
// register new person with db
// return true or error message

  // connect to db
  $conn = db_connect();

  // check if username is unique
  $result = $conn->query("select * from user where username='".$username."'");
  if (!$result) {
    throw new Exception('Could not execute query');
  }
  if ($result->num_rows>0) {
    throw new Exception('That username is taken - go back and choose another
one.');
  }

  // if ok, put in db
```

```
  $result = $conn->query("insert into user values
                          ('".$username."', sha1('".$password."'), '".$email."')");
  if (!$result) {
    throw new Exception('Could not register you in database - please try again
later.');
  }

  return true;
}
```

在这个函数中，也没有特别新的内容，只是将它连接到前面已经建立的数据库中。如果选定的用户名已经存在，或者数据库不能被更新，它将抛出一个异常。否则，它将更新数据库并返回true。

需要注意的是，我们使用了自己编写的函数db_connect()来执行数据库连接操作。该函数只提供了一个连接到数据库的地址，而该地址同时还保存着用户名和密码。这样，如果要修改数据库中的密码，只需改变应用程序中的一个文件即可。db_connect()函数如程序清单27-10所示。

程序清单27-10 db_fns.php文件中的db_connect()函数——该函数连接MySQL数据库

```
<?php

function db_connect() {
  $result = new mysqli('localhost', 'bm_user', 'password', 'bookmarks');
  if (!$result) {
    throw new Exception('Could not connect to database server');
  } else {
    return $result;
  }
}

?>
```

在用户注册之后，可以通过正规的登录或登出页面登录和退出网站。接下来，我们就将实现它。

27.5.2 登录

如果用户将他们的信息输入到login.php（如图27-3所示）的表单中，并提交给系统，系统将运行member.php脚本。如果用户信息正确，该脚本将允许用户登录，同时显示与登录用户所有相关的书签。这是本应用程序剩余部分的核心。该脚本如程序清单27-11所示。

程序清单27-11 member.php——该脚本是本应用程序的主体部分

```
<?php

// include function files for this application
```

```
require_once('bookmark_fns.php');
session_start();

//create short variable names
$username = $_POST['username'];
$passwd = $_POST['passwd'];

if ($username && $passwd) {
// they have just tried logging in
  try {
    login($username, $passwd);
    // if they are in the database register the user id
    $_SESSION['valid_user'] = $username;
  }
  catch(Exception $e) {
    // unsuccessful login
    do_html_header('Problem:');
    echo 'You could not be logged in.
          You must be logged in to view this page.';
    do_html_url('login.php', 'Login');
    do_html_footer();
    exit;
  }
}

do_html_header('Home');
check_valid_user();
// get the bookmarks this user has saved
if ($url_array = get_user_urls($_SESSION['valid_user'])) {
  display_user_urls($url_array);
}

// give menu of options
display_user_menu();

do_html_footer();
?>
```

可以看出脚本中的逻辑：我们在脚本中重用了第23章中的一些思想。

首先，检查用户是否是从前面的页面链接过来的——也就是说，他是否填了登录表单——然后尝试将其登录，如下所示：

```
if ($username && $passwd) {
// they have just tried logging in
  try {
    login($username, $passwd);
    // if they are in the database register the user id
```

```
    $_SESSION[ valid_user'] = $username;
}
```

可以看出，我们试图通过login()函数调用将用户登录进来。我们已经在user_auth_fns.php库中定义了这个函数。稍后，我们将详细介绍其源代码。

如果用户登录成功，我们就将注册他的会话，正如前面所做的一样。并将用户名保存到会话变量valid_user中。

如果一切顺利，为该用户显示成员页：

```
do_html_header('Home');
check_valid_user();
// get the bookmarks this user has saved
if ($url_array = get_user_urls($_SESSION['valid_user'])) {
  display_user_urls($url_array);
}

// give menu of options
display_user_menu();

do_html_footer();
```

该页面也是通过输出函数创建的。注意在这里我们使用了几个新函数。它们分别是user_auth_fns.php文件中的check_valid_user()函数、url_fns.php文件中的get_user_urls()函数，以及来自output_fns.php文件的display_user_urls()函数。check_valid_user()函数将检查当前用户是否拥有一个注册的会话。这是针对还没有登录却处于会话当中的用户。get_user_urls()函数将从数据库中获得用户的书签，而display_user_urls()函数将以表格的形式在浏览器中输出用户的书签。稍后，我们将详细介绍check_valid_user()函数，而其他两个函数将在书签存储和检索一节中介绍。

member.php脚本通过调用display_user_menu()函数显示一个菜单来结束本页面。图27-6显示了member.php的输出结果。

我们将进一步讨论login()和check_valid_user()函数。login()函数如程序清单27-12所示。

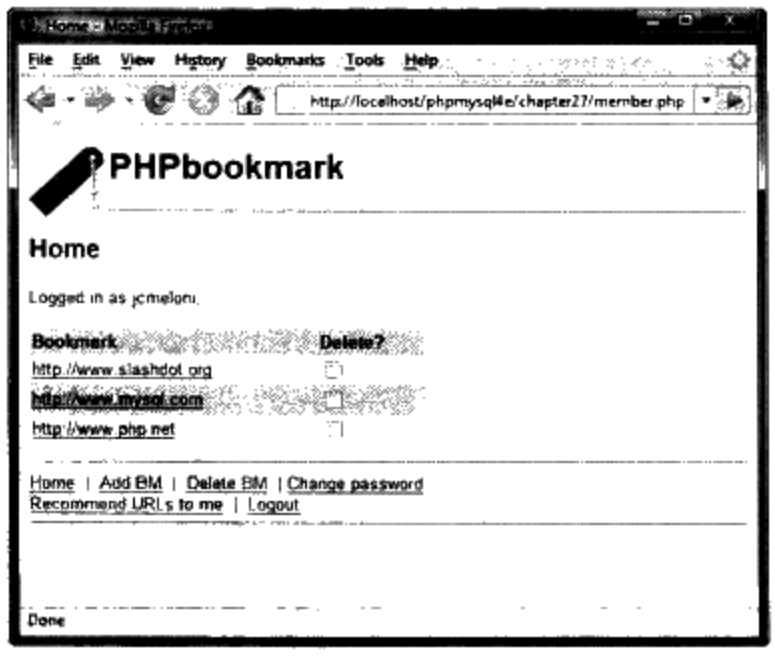

图27-6 member.php脚本检查用户是否登录；取回并显示他的书签；给出一个选项菜单

程序清单27-12 user_auth_fns.php文件中的login()函数——该函数将用户信息与数据库中保存的信息进行比较

```
function login($username, $password) {
// check username and password with db
```

```
// if yes, return true
// else throw exception

  // connect to db
  $conn = db_connect();

  // check if username is unique
  $result = $conn->query("select * from user
                          where username= ".$username."'
                          and passwd = sha1('".$password."')");
  if (!$result) {
     throw new Exception( Could not log you in. );
  }

  if ($result->num_rows>0) {
     return true;
  } else {
     throw new Exception('Could not log you in.');
  }
}
```

可以看到，该函数连接数据库并且检查数据库中是否有与本用户名和密码相匹配的用户。如果有，返回true，如果没有，或者本用户的信息不能被检查，将抛出一个异常。

函数check_valid_user()不再连接数据库，但是它将检查该用户是否有注册过的会话，也就是说，该用户是否已经登录。该函数如程序清单27-13所示。

程序清单27-13　user_auth_fns.php文件中的check_valid_user()函数——该函数检查用户是否有有效的会话

```
function check_valid_user() {
// see if somebody is logged in and notify them if not
  if (isset($_SESSION['valid_user'])) {
      echo "Logged in as ".$_SESSION['valid_user'].'.<br />";
  } else {
     // they are not logged in
     do_html_heading('Problem:');
     echo 'You are not logged in.<br />';
     do_html_url('login.php', 'Login');
     do_html_footer();
     exit;
  }
}
```

如果用户尚未登录，该函数将告诉他必须在登录之后才能浏览本页，并提供一个指向登录页的链接。

27.5.3 登出

我们可以已经注意到了，在图27-6所示的菜单选项上有一个标有“logout”的链接。该链接指向脚本logout.php。程序清单27-14给出了它的源代码。

程序清单27-14 logout.php——该脚本将结束一个用户会话

```
<?php

// include function files for this application
require_once('bookmark_fns.php');
session_start();
$old_user = $_SESSION['valid_user'];

// store to test if they *were* logged in
unset($_SESSION['valid_user']);
$result_dest = session_destroy();

// start output html
do_html_header('Logging Out');

if (!empty($old_user)) {
  if ($result_dest) {
    // if they were logged in and are now logged out
    echo 'Logged out.<br />';
    do_html_url('login.php', 'Login');
  } else {
   // they were logged in and could not be logged out
    echo 'Could not log you out.<br /> ;
  }
} else {
  // if they weren't logged in but came to this page somehow
  echo 'You were not logged in, and so have not been logged out.<br />';
  do_html_url( login.php', 'Login');
}

do_html_footer();

?>
```

我们会发现这段代码有些熟悉。因为它是根据第23章中编写的源代码而修改的。

27.5.4 修改密码

如果一个用户点击“Change Password”这个菜单选项，系统将打开如图27-7所示的表单。

表单是由脚本change_passwd_form.php生成的。该脚本比较简单，只是调用了输出库中的一些函数，因而在这里，我们没有介绍它的源代码。

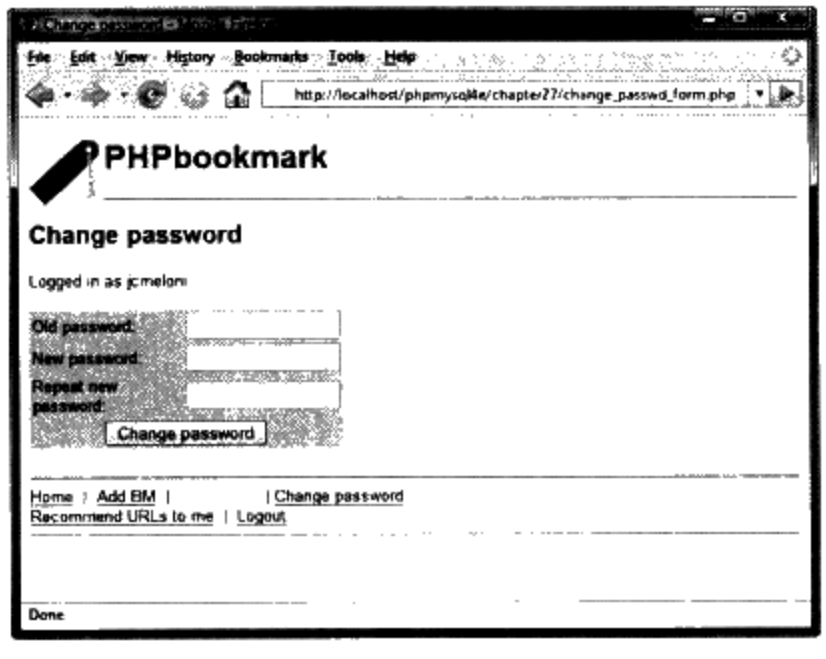

图27-7　change_passwd_form.php脚本为用户提供一个修改密码的表单

在用户提交该表单之后，将触发change_passwd.php脚本，如程序清单27-15所示。

程序清单27-15　change_passwd.php——该脚本将改变用户密码

```
<?php
  require_once('bookmark_fns.php');
  session_start();
  do_html_header('Changing password');

// create short variable names
$old_passwd = $_POST['old_passwd'];
$new_passwd = $_POST['new_passwd'];
$new_passwd2 = $_POST['new_passwd2'];

try {
  check_valid_user();
  if (!filled_out($_POST)) {
    throw new Exception('You have not filled out the form completely.
                         Please try again.');
  }

if ($new_passwd != $new_passwd2) {
  throw new Exception('Passwords entered were not the same.
                       Not changed.');
}

if ((strlen($new_passwd) > 16) || (strlen($new_passwd) < 6)) {
   throw new Exception('New password must be between 6 and 16 characters.
                        Try again.');
}

  // attempt update
```

```
    change_password($_SESSION['valid_user'], $old_passwd, $new_passwd);
    echo 'Password changed.';
  }
  catch (Exception $e) {
    echo $e->getMessage();
  }
  display_user_menu();
  do_html_footer();
?>
```

以上脚本将检查用户是否已经登录（使用check_valid_user()函数），是否已经填好密码表单（使用filled_out()函数），新密码是否一致以及其长度是否符合规定。这些都不是新内容，我们已经在以前介绍过了。如果所有这些都已经完成，可以调用change_password()函数，如下所示：

```
change_password($_SESSION['valid_user'], $old_passwd, $new_passwd);
echo 'Password changed.';
```

该函数来自user_auth_fns.php函数库。其源代码如程序清单27-16所示。

程序清单27-16 user_auth_fns.php文件中的change_password()函数——该函数更新数据库中的用户密码

```
function change_password($username, $old_password, $new_password) {
// change password for username/old_password to new_password
// return true or false

  // if the old password is right
  // change their password to new_password and return true
  // else throw an exception
  login($username, $old_password);
  $conn = db_connect();
  $result = $conn->query("update user
                          set passwd = sha1('".$new_password."')
                          where username = '".$username."'");
  if (!$result) {
    throw new Exception('Password could not be changed.');
  } else {
    return true;  // changed successfully
  }
}
```

该函数调用了前面所介绍的login()函数来判断用户输入的旧密码是否正确。如果正确，函数将连接到数据库并将它更新为新密码。

27.5.5 重设遗忘的密码

除了修改密码，我们还需要解决用户忘记其密码的情形。请注意，在首页login.php中，

我们专门为此情形提供了一个链接，“Forgotten your password?”，该链接指向脚本forgot_ form.php，该脚本调用输出函数来显示如图27-8所示的表单。

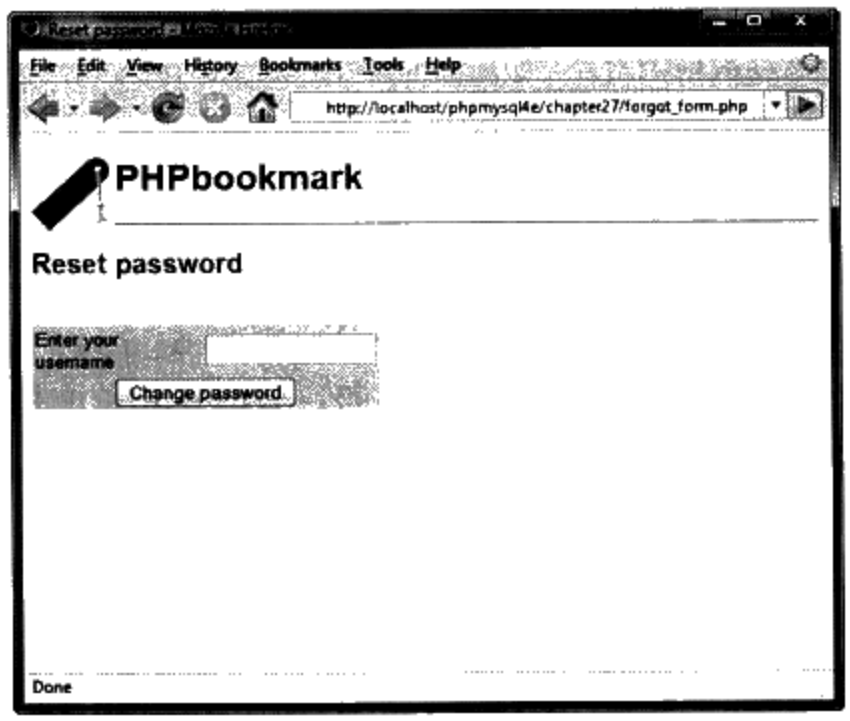

图27-8　forgot_form.php脚本提供了一个表单，在这里，用户可以请求重置他们的密码并发送给他们

该脚本非常简单，只是调用了一些输出函数，因此，我们将不再详细介绍它。当提交表单时，它将调用forgot_passwd.php脚本，这段代码更加有趣。其脚本如程序清单27-17所示。

程序清单27-17　forgot_passwd.php——该脚本将用户密码重置为一个随机值并将新密码发送到用户的邮箱

```
<?php
  require_once("bookmark_fns.php");
  do_html_header('Resetting password');

  // creating short variable name
  $username = $_POST['username'];

  try {
    $password = reset_password($username);
    notify_password($username, $password);
    echo 'Your new password has been emailed to you.<br />';
  }
  catch (Exception $e) {
    echo 'Your password could not be reset - please try again later.';
  }
  do_html_url('login.php', 'Login');
  do_html_footer();
?>
```

可以看到，该脚本使用两个主要函数来实现此功能：reset_password()和notify_

password()。我们将逐一介绍它们。

reset_password()函数将产生一个随机密码并将它保存到数据库。该函数的代码如程序清单27-18所示。

程序清单27-18 user_auth_fns.php文件中的reset_password()函数——该脚本将用户密码重置为随机值并将其发送到该用户邮箱

```
function reset_password($username) {
// set password for username to a random value
// return the new password or false on failure
  // get a random dictionary word b/w 6 and 13 chars in length
  $new_password = get_random_word(6, 13);

  if($new_password == false) {
    throw new Exception('Could not generate new password.');
  }

  // add a number between 0 and 999 to it
  // to make it a slightly better password
  $rand_number = rand(0, 999);
  $new_password .= $rand_number;

  // set user's password to this in database or return false
  $conn = db_connect();
  $result = $conn->query("update user
                          set passwd = sha1('".$new_password."')
                          where username = '".$username."'");
  if (!$result) {
    throw new Exception('Could not change password.'); // not changed
  } else {
    return $new_password; // changed successfully
  }
}
```

该函数通过从字典里获取随机单词来生成一个随机密码。调用get_random_word()函数并在得到的单词后面添加一个0～999之间的随机数作后缀。get_random_word()函数也包含在user_auth_fns.php库中。其脚本如程序清单27-19所示。

程序清单27-19 user_auth_fns.php文件中的get_random_word()函数——该函数从词典中获取一个随机单词，以生成新密码

```
function get_random_word($min_length, $max_length) {
// grab a random word from dictionary between the two lengths
// and return it

  // generate a random word
  $word = '';
  // remember to change this path to suit your system
```

```
  $dictionary = '/usr/dict/words'; // the ispell dictionary
  $fp = @fopen($dictionary, 'r');
  if(!$fp) {
    return false;
  }
  $size = filesize($dictionary);

  // go to a random location in dictionary
  $rand_location = rand(0, $size);
  fseek($fp, $rand_location);

  // get the next whole word of the right length in the file
  while ((strlen($word) < $min_length) || (strlen($word)>$max_length) ||
(strstr($word, "'"))) {
    if (feof($fp)) {
      fseek($fp, 0);          // if at end, go to start
    }
    $word = fgets($fp, 80);   // skip first word as it could be partial
    $word = fgets($fp, 80);   // the potential password
  }
  $word = trim($word); // trim the trailing \n from fgets
  return $word;
}
```

要使该函数能够正常工作，我们需要一个词典。如果使用的是UNIX系统，其内置的拼写检查器ispell就带有单词词典，通常它位于/usr/dict/words或/usr/share/dict/words目录下。如果在以上两个位置都没有找到，在大多数系统上，可以使用如下命令找到一个词典：

```
$ locate dict/words
```

如果使用的是其他的系统而且不愿安装ispell，不用担心，可以下载ispell使用的单词列表，其下载地址如下所示：http://wordlist.sourceforge.net/。

该网站也有许多其他语言的词典，因此如果喜欢其他任意一种语言，例如，Norwegian或Esperanto的单词，也可下载这些词典。所有这些文件的格式都是每个单词一行，每行通过换行符分开。

要从该文件中获取一个随机单词，首先应选取一个介于0到文件长度之间的位置，并从此位置开始读文件。如果从该随机位置开始一行一行地读，获取的很可能是单词的一部分，因此，我们通过两次调用fgets()函数，跳过开始的随机行，而将下面的一个单词作为需要的单词。

该函数有两处设计很巧妙。第一，如果在查找单词的时候到了文件结尾，可以从头开始，如下代码所示：

```
if (feof($fp)) {
   fseek($fp, 0);          // if at end, go to start
}
```

第二，可以搜索特定长度的单词：我们搜索从词典中抽出的每个单词，如果它的长度没有

介于$min_length和$max_length之间，就继续搜索。同时，我们还将过滤带有单引号的单词。当使用该词时，我们过滤这些字符，但是获得下一个单词会更容易一些。

回到reset_password()函数，在生成了一个新密码之后，需要更新数据库以体现密码已被修改，并将新密码返回到主脚本；然后又将它传到notify_password()这个函数，该函数将新密码发送到用户邮箱。下面，让我们来了解notify_password()函数，如程序清单27-20所示。

程序清单27-20 user_auth_fns.php文件中的notify_password()函数——该函数将新密码以电子邮件方式发送给用户

```
function notify_password($username, $password) {
// notify the user that their password has been changed

    $conn = db_connect();
    $result = $conn->query("select email from user
                            where username='".$username."'");
    if (!$result) {
      throw new Exception('Could not find email address.');
    } else if ($result->num_rows == 0) {
      throw new Exception('Could not find email address.');
      // username not in db
    } else {
      $row = $result->fetch_object();
      $email = $row->email;
      $from = "From: support@phpbookmark \r\n";
      $mesg = "Your PHPBookmark password has been changed to '.$password."\r\n"
              ."Please change it next time you log in.\r\n";

      if (mail($email, 'PHPBookmark login information', $mesg, $from)) {
        return true;
      } else {
        throw new Exception('Could not send email.');
      }
    }
}
```

在这个函数中，给定一个用户名和密码，我们只需要在数据库中查找该用户的邮箱地址，调用PHP的mail()函数将其发送给该用户。

给用户一个真正随机的密码是更保险的—该密码是任何小写字母、大写字母、数字和标点符号的组合——而不只是如上设计的随机单词和数字的组合。但是，像“zigzag487”这样的密码，用户更易阅读和输入，这比真正随机数好。因为用户通常容易混淆字符串中的0和O（数字0和大写O），以及1和l（数字1和小写l）。

在我们的系统中，词典文件包含了45 000个单词记录。如果一个黑客知道我们是如何创建密码的，而且知道用户名称，就可以在尝试22 500 000次获得一个用户的密码。看上去，这种

安全级别对于这种类型的应用程序是足够的，即使我们的用户没有按照电子邮件中的建议再次修改密码。

27.6 实现书签的存储和检索

在实现了与用户账户相关的功能后，现在，我们开始讨论如何保存、检索和删除书签。

27.6.1 添加书签

用户点击用户菜单上的“Add BM”链接可以添加书签。该链接将用户带到如图27-9所示的页面。

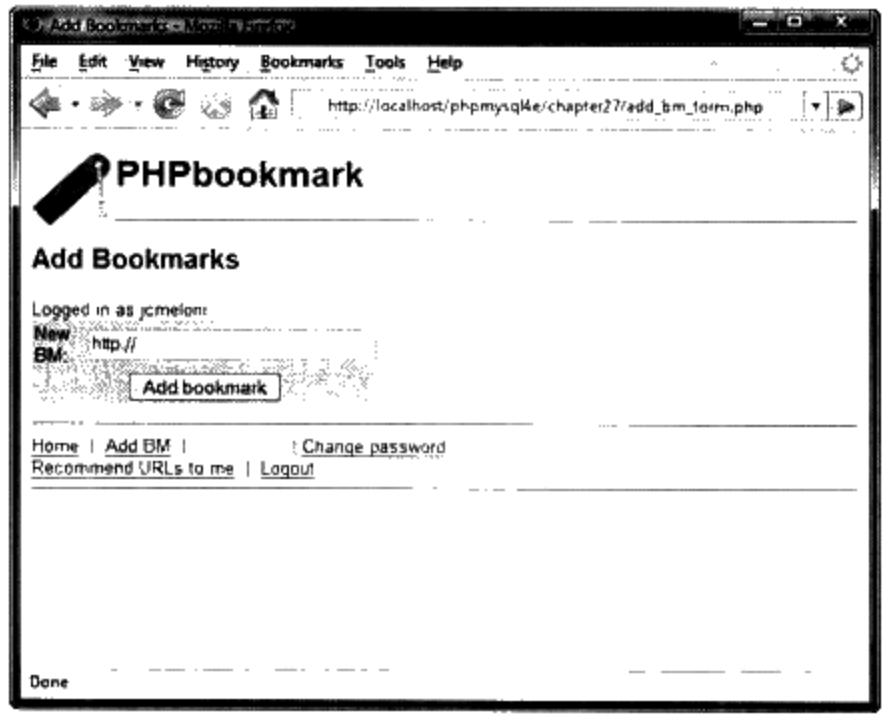

图27-9 add_bm_form.php脚本将提供一个表单，用户在此可以将书签添加到他们的书签页中

同样，由于这段脚本也是非常简单的，并且只调用了一些输出函数，因此，我们也不深入讨论。当表单被提交的时候，系统调用add_bms.php脚本，该脚本如程序清单27-21所示。

程序清单27-21 add_bms.php——该脚本添加新书签到用户的个人页面

```
<?php
 require_once('bookmark_fns.php');
 session_start();

  //create short variable name
  $new_url = $_POST['new_url'];
  do_html_header('Adding bookmarks');

  try {
    check_valid_user();
    if (!filled_out($_POST)) {
      throw new Exception('Form not completely filled out.');
    }
```

```
    // check URL format
    if (strstr($new_url, 'http://') === false) {
      $new_url = 'http://'.$new_url;
    }

    // check URL is valid
    if (!(@fopen($new_url, 'r'))) {
      throw new Exception('Not a valid URL.');
    }

    // try to add bm
    add_bm($new_url);
    echo 'Bookmark added.';

    // get the bookmarks this user has saved
    if ($url_array = get_user_urls($_SESSION['valid_user'])) {
      display_user_urls($url_array);
    }
  }
  catch (Exception $e) {
    echo $e->getMessage();
  }
  display_user_menu();
  do_html_footer();
?>
```

这段脚本也遵循了验证、数据库输入和输出的模式。

要验证用户身份，我们应该首先调用filled_out()函数检查该用户是否完全填写表单。然后，再执行两项检查URL的操作。首先，调用strstr()函数，检查URL是否以http://开头。如果不是，我们就将其添加到URL的开头。完成此操作后，就可确切地检查该URL是否存在。回顾一下第20章“使用网络函数和协议函数”，我们可以调用fopen()函数打开一个以http://开头的URL。如果可以打开这个文件，就假定该URL是有效的，并调用add_bm()函数将其添加到数据库中。

本函数和其他与书签相关的函数都保存在函数库url_fns.php中。程序清单27-22显示了add_bm()函数的代码。

程序清单27-22 url_fns.php文件中的add_bm()函数——
该函数将用户提交的新书签添加到数据库中

```
<?php
 require_once('bookmark_fns.php');
 session_start();

  //create short variable name
  $new_url = $_POST['new_url'];
```

```
do_html_header('Adding bookmarks');

try {
  check_valid_user();
  if (!filled_out($_POST)) {
    throw new Exception('Form not completely filled out.');
  }

  // check URL format
  if (strstr($new_url, 'http://') === false) {
     $new_url = 'http://'.$new_ur.;
  }

  // check URL is valid
  if (!(@fopen($new_url, 'r'))) {
    throw new Exception('Not a valid URL.');
  }

  // try to add bm
  add_bm($new_url);
  echo 'Bookmark added.';

  // get the bookmarks this user has saved
  if ($url_array = get_user_urls($_SESSION['valid_user'])) {
      display_user_urls($url_array);
    }
  }
  catch (Exception $e) {
    echo $e->getMessage();
  }
  display_user_menu();
  do_html_footer();
?>
```

该函数也很简单。它检查用户是否在数据库中已经有了该书签（尽管他们不可能两次输入同一个书签，但很可能要更新该页）。如果书签是新的，它就被添加到数据库中。

回头看看add_bm.php函数库，可以看出，它最后执行的操作是调用get_user_urls()函数和display_user_urls()函数，这与member.php是相同的。我们将在接下来的内容中继续讨论这些函数。

27.6.2　显示书签

在member.php脚本和add_bm()函数中，我们使用了函数get_user_urls()和display_user_urls()。它们分别从数据库中检索用户的书签和显示这些书签。get_user_urls()函数包含在url_fns.php库中。而display_user_urls()函数包含在output_fns.php库中。

get_user_urls()函数如程序清单27-23所示。

我们简要介绍一下该函数的执行步骤，它以用户名作为参数，从数据库中取回该用户的书签，返回一组URL；或者如果书签获取失败，返回false值。

程序清单27-23 url_fns.php文件中的get_user_urls()函数——该函数从数据库中取回用户书签

```
function get_user_urls($username) {
  //extract from the database all the URLs this user has stored

  $conn = db_connect();
  $result = $conn->query("select bm_URL
                          from bookmark
                          where username = '".$username."'");
  if (!$result) {
    return false;
  }

  //create an array of the URLs
  $url_array = array();
  for ($count = 0; $row = $result->fetch_row(); ++$count) {
    $url_array[$count] = $row[0];
  }
  return $url_array;
}
```

get_user_urls()函数将返回一组可以传给函数display_user_urls()的URL。该函数也是一个简单的HTML输出函数，它可将用户的URL以美观的表格形式显示在浏览器中，在这里，我们也不详细讨论。回到图27-6，看看输出是什么。实际上函数将URL输出到一个表单。而每个URL右边是一个复选框，用以选定要删除的书签。接下来，我们就将讨论它。

27.6.3 删除书签

当用户将一些书签标记为删除并点击菜单选项中的“Delete BM”时，就将提交一个包含URL的表单。每个复选框由display_user_urls()函数中的如下代码产生：

```
echo "<tr bgcolor=\"".$color."\"><td><a
➥href=\"".$url."\">".htmlspecialchars($url)."</a></td>
      <td><input type=\"checkbox\" name=\"del_me[]\"
           value=\"".$url."\"/></td>
      </tr>";
```

每个输入的名称是del_me[]。这就是说，在该表单激活的PHP脚本中，我们可以访问名为$del_me的数组，该数组包含所有要删除的书签。

点击“Delete BM”就触发了delete_bms.php脚本，该脚本源代码如程序清单27-24所示。

程序清单27-24　delete_bms.php——该脚本从数据库中删除书签

```
<?php
  require_once('bookmark_fns.php');
  session_start();

  //create short variable names
  $del_me = $_POST['del_me'];
  $valid_user = $_SESSION['valid_user'];

  do_html_header('Deleting bookmarks');
  check_valid_user();

  if (!filled_out($_POST)) {
    echo '<p>You have not chosen any bookmarks to delete.<br/>
          Please try again.</p>';
    display_user_menu();
    do_html_footer();
    exit;
  } else {
    if (count($del_me) > 0) {
      foreach($del_me as $url) {
        if (delete_bm($valid_user, $url)) {
          echo 'Deleted '.htmlspecialchars($url).'.<br />';
        } else {
          echo 'Could not delete '.htmlspecialchars($url).'.<br />';
        }
      }
    } else {
      echo 'No bookmarks selected for deletion';
    }
  }

  // get the bookmarks this user has saved
  if ($url_array = get_user_urls($valid_user)) {
    display_user_urls($url_array);
  }

  display_user_menu();
  do_html_footer();
?>
```

在脚本的开始，我们执行了常规的确认操作。当确定用户已经删除选中的书签时，将通过如下的循环将其删除：

```
foreach($del_me as $url) {
  if (delete_bm($valid_user, $url)) {
    echo 'Deleted '.htmlspecialchars($url).'.<br />';
```

```
    } else {
      echo 'Could not delete '.htmlspecialchars($url).'.<br />';
    }
  }
```

可以看到，delete_bm()函数并没有执行从数据库中删除书签的实际工作。该函数如程序清单27-25所示。

程序清单27-25 url_fns.php文件中的delete_bm()函数——该函数从用户的书签列表中删除一个书签

```
function delete_bm($user, $url) {
  // delete one URL from the database
  $conn = db_connect();

  // delete the bookmark
  if (!$conn->query("delete from bookmark where
    username='".$user."' and bm_url='".$url."' ")) {
    throw new Exception('Bookmark could not be deleted');
  }
  return true;
}
```

可以看到，这也是一个相当简单的函数。它试图从数据库中删除特定用户的书签。需要注意的是，我们要删除的是“用户名-书签”对。其他用户可能仍然拥有此书签URL。

在系统中运行删除脚本的输出示例如图27-10所示。

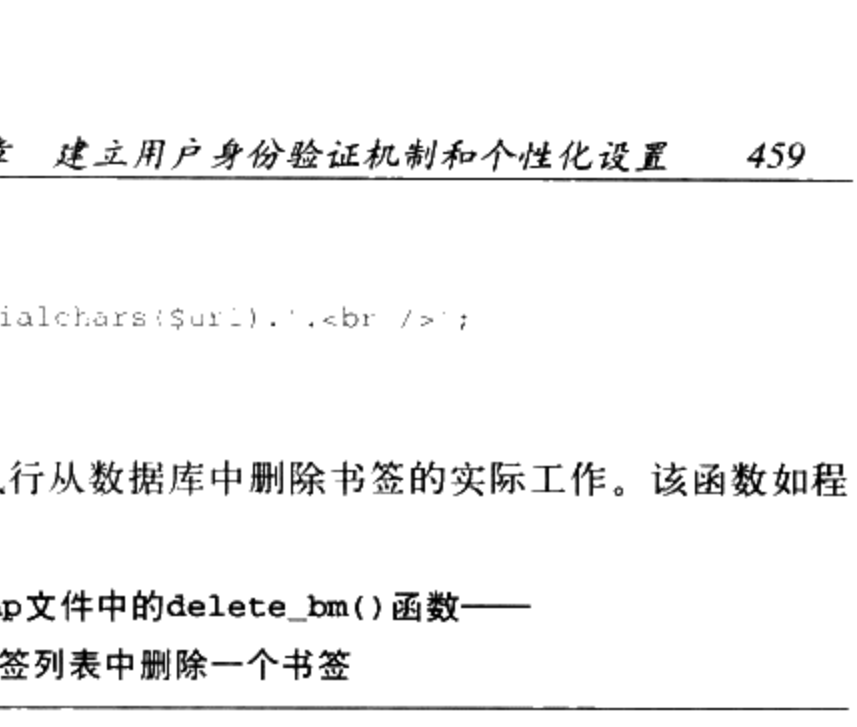

图27-10 删除脚本将通知用户已删除了书签，然后显示其余的书签

与在add_bms.php脚本中的操作类似，当数据库被修改之后，我们将调用get_user_urls()函数和display_user_urls()函数显示新的书签。

27.7 实现书签推荐

最后，我们将讨论书签推荐脚本recommend .php。我们可以通过许多方法实现书签推荐。在此，我们决定应用“相似意向”的推荐。该推荐的含义是，查找与给定用户至少有一个相同书签的其他用户。其他用户的其他书签也对给定的用户有吸引力。

将“相似意向”应用到SQL查询最简单的方法是使用子查询。第一个子查询如下所示：

```
select distinct(b2.username)
            from bookmark b1, bookmark b2
```

```
            where b1.username='".$valid_user."'
            and b1.username != b2.username
            and b1.bm_URL = b2.bm_URL)
```

这个查询使用别名将数据库表bookmark进行自身连接——这是一个很奇怪但是有时候又非常有用的概念。假设有两个书签表，b1和b2。在b1中，查询当前用户及其书签。在另一个表中，查询所有其他用户的书签。我们需要查找的是用户书签中有一个URL与当前用户相同(b1.bm_URL=b2.bm_URL)的其他用户(b2.username)。其他用户不包括当前用户(b1.username!=b2.username)。

该查询将给出一个与当前用户意向相似的人的列表。得到了这个用户列表后，可以用下面的查询搜索他们的其他书签了：

```
select bm_URL
from bookmark
where username in
            (select distinct(b2.username)
            from bookmark b1, bookmark b2
            where b1.username='".$valid_user."'
            and b1.username != b2.username
            and b1.bm_URL = b2.bm_URL)
```

可以添加第二个子查询来过滤当前用户的书签；如果用户已经有了这些书签，就不必再将该书签推荐给他。最后，对$popularity变量进行书签过滤。我们不希望推荐太个性化的URL，因此只将一定数量的其他用户做了书签的URL推荐给用户。最终的查询如下所示：

```
select bm_URL
from bookmark
where username in
            (select distinct(b2.username)
            from bookmark b1, bookmark b2
            where b1.username='".$valid_user."'
            and b1.username != b2.username
            and b1.bm_URL = b2.bm_URL)
and bm_URL not in
            (select bm_URL
            from bookmark
            where username='".$valid_user."')
group by bm_url
having count(bm_url)>".$popularity;
```

如果我们期望许多用户使用我们的系统，可以调整变量$popularity，只推荐许多其他用户做了书签的URL。许多人做了书签的URL可能质量更高，这样的书签当然比一般的页面更大众化、更具吸引力。

实现书签推荐的完整脚本如程序清单27-26和程序清单27-27所示。推荐的主脚本称为recommend.php（请参阅程序清单27-26），它调用来自url_fns.php函数库（请参阅程序清单27-27）的函数recommend_urls()。

程序清单27-26 recommend.php——推荐某一用户可能喜欢的书签

```
<?php
  require_once('bookmark_fns.php');
  session_start();
  do_html_header('Recommending URLs');
  try {
    check_valid_user();
    $urls = recommend_urls($_SESSION['valid_user']);
    display_recommended_urls($urls);
  }
  catch(Exception $e) {
    echo $e->getMessage();
  }
  display_user_menu();
  do_html_footer();
?>
```

程序清单27-27 url_fns.php文件中的recommend_urls()函数——该脚本做出实际的推荐

```
function recommend_urls($valid_user, $popularity = 1) {
  // We will provide semi intelligent recommendations to people
  // If they have an URL in common with other users, they may like
  // other URLs that these people like
  $conn = db_connect();

// find other matching users
// with an url the same as you
// as a simple way of excluding people's private pages, and
// increasing the chance of recommending appealing URLs, we
// specify a minimum popularity level
// if $popularity = 1, then more than one person must have
// an URL before we will recommend it

$query = "select bm_URL
          from bookmark
          where username in
            (select distinct(b2.username)
             from bookmark b1, bookmark b2
             where b1.username='".$valid_user."'
             and b1.username != b2.username
             and b1.bm_URL = b2.bm_URL)
          and bm_URL not in
             (select bm_URL
              from bookmark
              where username='".$valid_user."')
          group by bm_url
          having count(bm_url)>".$popularity;
```

```
    if (!($result = $conn->query($query))) {
        throw new Exception('Could not find any bookmarks to recommend.');
    }

    if ($result->num_rows==0) {
        throw new Exception('Could not find any bookmarks to recommend.');
    }

    $urls = array();
    // build an array of the relevant urls
    for ($count=0; $row = $result->fetch_object(); $count++) {
        $urls[$count] = $row->bm_URL;
    }

    return $urls;
}
```

recommend.php的输出示例如图27-11所示。

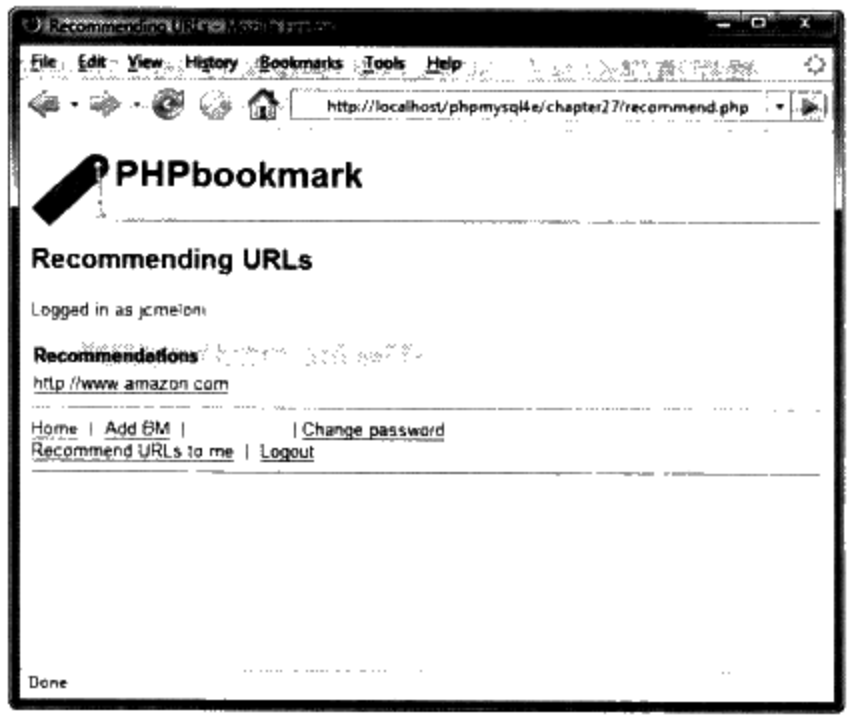

图27-11　脚本向该用户推荐了他可能喜欢的amazon.com，数据库中至少有两位用户将amazon.com作为他们的书签

27.8　考虑可能的扩展

在以上的内容中，我们介绍了PHPbookmark应用程序的基本功能。它还有许多可以扩展的地方。例如，可以考虑添加：

- 按主题分类的一组书签
- 书签推荐功能中的“将此URL添加到我的书签”的链接
- 基于数据库中最流行URL的推荐，或者基于某一特定主题的推荐
- 一个管理界面，用以创建、管理用户和书签
- 使推荐书签更智能化或更快的方法
- 附加的用户输入错误检查

实践！这是最好的学习方法。

27.9　下一章

在下一个项目中，我们将创建一个购物车，使用该购物车可以让用户浏览我们的网站，并在浏览的时候购买商品，直到最后结账并使用电子的方式付款。