第28章　创建一个购物车

在本章中，我们将学习如何创建一个基本的购物车。我们将此购物车添加到Book-O-Rama数据库之上，该数据库是我们在第二篇“使用MySQL”实现的。此外，我们还将探讨另外一种选择：创建和使用一个已有的、开放源代码的PHP购物车。

如果以前还没有听说过这些内容，那让我们先了解一下术语“*购物车*”。“*购物车*”（有时也叫*购物篮*）用来描述一种特定的在线购物机制。当浏览一个在线目录的时候，可以将一些商品添加到购物车中。在完成浏览之后，要与在线商店结账——也就是买下购物车内的商品。

为了实现购物车，我们需要完成下面的功能：

- 在线出售商品的数据库
- 一个在线产品目录，按商品种类分类
- 一个能记录用户打算购买商品的购物车
- 结账脚本，处理付款和运送细节
- 一个管理界面

28.1　解决方案的组成

回忆一下在本书第二篇创建的Book-O-Rama数据库。在本项目中，我们将建立Book-O-Rama在线商店并运行它。该解决方案的组成主要包括如下所示。

- 我们需要找到一种将数据库连接到用户的浏览器的方法。用户能够按目录浏览商品。
- 用户应该能够从商品目录中选取商品以便此后购买。我们也要能够记录他们选中的物品。
- 当用户完成购买，要合计他们的订单，获取运送商品细节，并处理付款。
- 我们还应该为Book-O-Rama网站创建一个管理界面，以便管理员在上面添加、编辑图书和目录。

现在，我们已经了解了项目的需求，开始开始设计解决方案及其组成。

28.1.1　创建一个在线目录

我们已经有了一个用来创建Book-O-Rama目录的数据库。但是，要实现该目录，还需要修改和添加一些功能。其中之一就是要添加图书的目录，如需求中所述。

我们还要为现存的数据库添加一些关于商品运送地址、付款细节等的信息。我们已经知道了如何使用PHP创建一个连接到MySQL数据库的界面，因此创建管理界面这一步就很简单了。

在完成顾客订单的时候，还应该使用事务。要使用事务，必须将Book-O-Rama表格转换成使用InnoDB存储引擎。这个处理也是非常简单和直观的。

28.1.2　在用户购买商品的时候记录购买行为

记录用户所购买的商品有两种基本方法。一种是将用户的选择存入数据库中，另一种是使用会话变量。

使用会话变量逐页记录用户的方法是很容易实现的，因为它不要求我们不停地查询数据库。这样做也避免了在结束的时候留下许多垃圾数据，这些数据来自于那些只是浏览或不停地改变主意的用户。

因此，我们需要设计一个或一组会话变量来保存用户的选择。当用户完成购买并付款之后，将此信息送到数据库作为一个事务处理的记录。

我们还可以使用该数据给出一个当前购物车的摘要描述，将其显示在页面的某个位置，以便用户在任何时候都知道花费是多少。

28.1.3　实现一个付款系统

在本项目中，我们要合计用户的订单总价，并获取送货详细信息。实际上，我们并不处理任何付款。如今，有许多的付款系统可供使用，但是，对于每一个系统应用，付款系统又各不相同。我们将编写一个dummy函数，该函数可以用所选定系统的界面代替。

尽管有一些不同的付款网关可以使用，但是，对于这些付款网关的接口来说，实时信用卡处理接口的功能都是类似的。你必须在银行开通一个商业账户，确定能够接受的信用卡类型——通常，你的银行会针对你所选择的支付系统给出一个推荐的信用卡提供商列表，你的付款系统提供商会给出该付款系统所需的参数以及如何传递这些参数。大多数付款系统都有PHP版本的示例代码，这样就便于替代本章所创建的示例函数。

当使用付款系统时，系统将你的数据发送给银行并且返回成功代码或任何一个错误代码类型。在数据交换的时候，付款网关将收取设置费用或年费，以及基于交易金额的手续费。某些提供商甚至会收取取消交易的费用。

不过，你的付款系统至少需要来自客户的信息（例如，信用卡），标识你的信息（指定交易款项的商业账户），以及交易的总金额。

我们可以通过一个用户的购物车会话变量计算出订单的总量。将订单的最终信息记录到数据库，并在此时销毁会话变量。

28.1.4　创建一个管理界面

除了这些，我们将创建一个管理员界面。在此界面上，可以添加、删除和编辑数据库中的图书及目录。

通常，我们将要用到的一个编辑功能是修改某一物品的价格（例如，要举行一次特别的出售活动或甩卖）。这就意味着在保存用户订单的时候，也要保存他所买商品的价格。如果我们已经保存的记录只有每个用户所订购的物品，而按照每个商品的当前价格来计费，可以想像，那会是一场“噩梦”。这也意味着如果客户要退回或更换商品，我们也可以正确地计算商品价格。

在本项目中，我们并不打算创建一个可履行和记录订单的界面。我们可以根据需要在这个

基本系统中添加此功能。

28.2 解决方案概述

现在，让我们将各个部分整合到一起。该系统有两个基本视图：用户视图和管理员视图。在仔细考虑需要的功能之后，我们提出了两个流程设计，一个视图对应一个流程设计。它们分别如图28-1和图28-2所示。

在图28-1中，我们显示了网站中用户部分的脚本之间的主要链接。用户将首先进入主页面，该页面给出了网站中所有图书的目录。从该页面开始，用户可以进入特定的图书目录，从该目录又可以进入某一本书的详细介绍。

我们将为用户提供一个链接以添加特定的图书到购物车。根据购物车中的商品，用户可以与该在线商店结账付款。

图28-2所示的是管理员界面——它需要更多的脚本，但是新脚本却并不是很多。这些脚本允许管理员登录并插入图书和目录记录。

实现编辑、删除图书和目录最简单的方法是为管理员提供一个界面，该界面与用户界面略有不同。管理员仍然能够浏览目录和图书，但与用户访问购物车不同的是，管理员能够进入特定的图书和目录，并且编辑和删除该书和目录。通过设计适用一般用户和管理员的同样脚本，我们可以节省时间和精力。

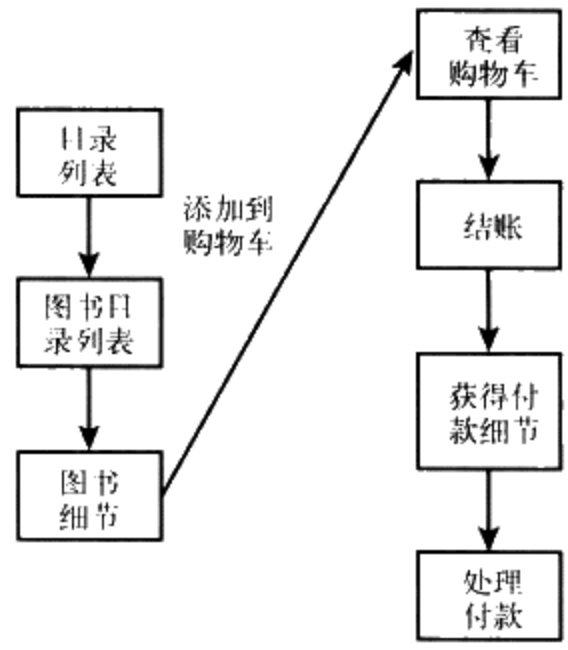

图28-1 Book-O-Rama系统的用户视图可使得用户按照目录查看图书，浏览图书的详细信息，将书添加到购物车，以及购买它们

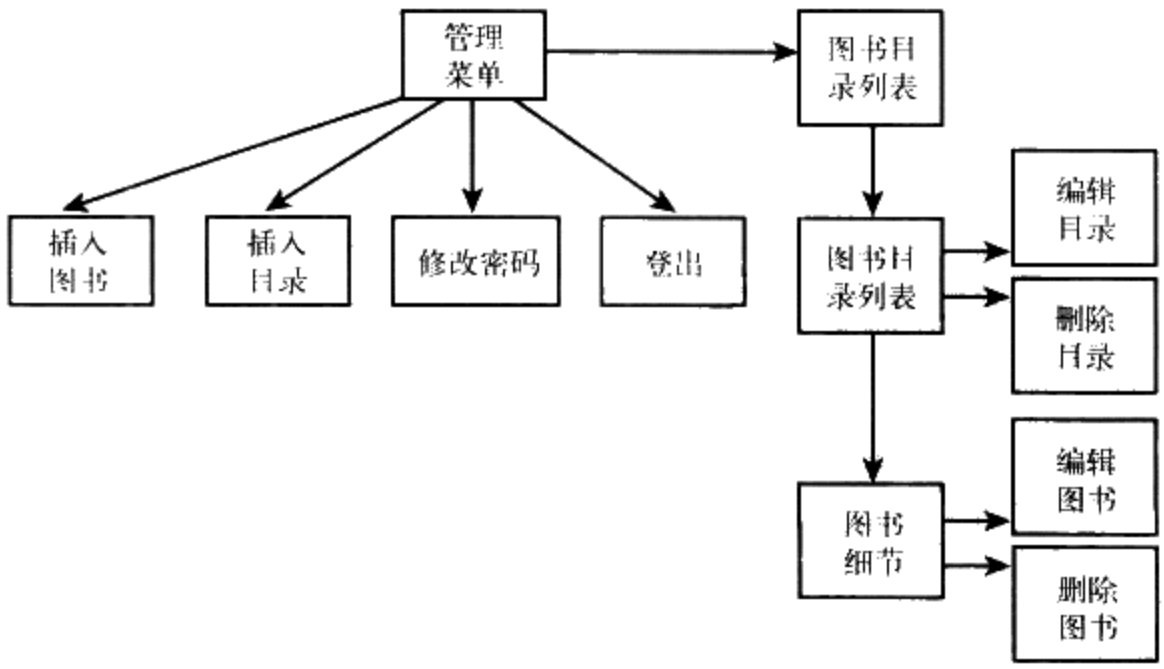

图28-2 Book-O-Rama系统管理员视图允许管理员插入、编辑和删除图书及目录

该应用程序由如下所示的三个主要代码模块组成：

- 目录
- 购物车和订单处理（我们将此二者捆绑在一起是因为它们之间的联系非常紧密）

■ 管理

与第27章“建立用户身份验证机制和个性化设置”介绍的项目一样，我们也要创建和使用一组函数库。在这个项目中，我们将使用一个函数API，它与上一个项目的函数API类似。我们试图把输出HTML的代码放到一个函数库中，以支持分离逻辑和内容的原则，更重要的是，这样做可以使代码更易阅读和维护。

针对这个项目，我们也需要对Book-O-Rama数据库作一些小修改。我们已经重新命名了数据库book_sc（购物车），以区分在本书第二篇所创建的数据库。

本项目所需的所有代码都可以在本书附带的文件中找到。表28-1给出了本应用程序中用到的这些文件的摘要。

表28-1　购物车应用程序中用到的文件

名　称	模　块	描　述
index.php	目录	网站首页，显示系统中的图书目录
show_cat.php	目录	显示特定目录包含的所有图书
show_book.php	目录	显示特定图书的详细信息
show_cart.php	购物车	显示用户购物车的内容。也用来向购物车中添加图书
checkout.php	购物车	向用户显示所有的订单细节。获取商品运送细节
purchase.php	购物车	从用户获取付款细节
process.php	购物车	处理付款细节，将订单添加到数据库
login.php	管理	允许管理员登录进行修改
logout.php	管理	管理员退出
admin.php	管理	主管理菜单
change_password_form.php	管理	允许管理员修改密码的表格
change_password.php	管理	修改管理员密码
insert_category_form.php	管理	允许管理员向数据库中添加一个目录的表格
insert_category.php	管理	向数据库中插入新目录
insert_book_form.php	管理	管理员添加新书到系统的表单
insert_book.php	管理	将新书插入到数据库
edit_category_form.php	管理	管理员编辑目录的表单
edit_category.php	管理	更新数据库中的目录
edit_book_form.php	管理	管理员编辑图书信息的表单
edit_book.php	管理	更新数据库中的图书信息
delete_category.php	管理	从数据库中删除一个目录
delete_book.php	函数	从数据库中删除一本书
book_sc_fns.php	函数	该应用程序的包含文件集合
admin_fns.php	函数	管理脚本使用的函数集合
book_fns.php	函数	用以保存和获取图书数据的函数集合
order_fns.php	函数	用以保存和获取订单数据的函数集合
output_fns.php	函数	输出HTML的函数集合
data_valid_fns.php	函数	验证用户输入数据的函数集合
db_fns.php	函数	连接book_sc数据库的函数集合
user_auth_fns.php	函数	授权管理员用户的函数集合
book_sc.sql	SQL	创建book_sc数据库的SQL
populate.sql	SQL	插入样本数据到book_sc数据库中的SQL

现在，让我们开始了解每个模块的具体实现。

提示　该应用程序的代码量非常大。其中要实现的许多功能我们都已经见过（特别是在上一章中），例如，将数据保存到数据库，从数据库中取回数据，授予用户管理权限。在这方面的代码，我们只简要地看一下，而将大部分时间花在购物车函数上。

28.3　实现数据库

正如前面所提到的，我们已经对在本书第二篇创建的数据库进行了少量修改。创建book_sc数据库的SQL代码如程序清单28-1所示。

程序清单28-1　book_sc.sql——创建book_sc数据库的SQL代码

```
create database book_sc;

use book_sc;

create table customers
(
  customerid int unsigned not null auto_increment primary key,
  name char(60) not null,
  address char(80) not null,
  city char(30) not null,
  state char(20),
  zip char(10),
  country char(20) not null
) type=InnoDB;

create table orders
(
  orderid int unsigned not null auto_increment primary key,
  customerid int unsigned not null references customers(customerid),
  amount float(6,2),
  date date not null,
  order_status char(10),
  ship_name char(60) not null,
  ship_address char(80) not null,
  ship_city char(30) not null,
  ship_state char(20),
  ship_zip char(10),
  ship_country char(20) not null
) type=InnoDB;

create table books
(
   isbn char(13) not null primary key,
   author char(100),
```

```
    title char(100),
    catid int unsigned,
    price float(4,2) not null,
    description varchar(255)
) type=InnoDB;

create table categories
(
  catid int unsigned not null auto_increment primary key,
  catname char(60) not null
) type=InnoDB;

create table order_items
(
  orderid int unsigned not null references orders(orderid),
  isbn char(13) not null references books(isbn),
  item_price float(4,2) not null,
  quantity tinyint unsigned not null,
  primary key (orderid, isbn)
) type=InnoDB;

create table admin
  (
  username char(16) not null primary key,
  password char(40) not null
);

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

尽管最初的Book-O-Rama界面并没有错误，但要在线使用，必须满足一些其他的新需求。对最初数据库进行的修改如下所示：

- 增加更多的用户地址域。既然要建立实际的应用程序，这就很重要了。
- 为每个订单增加一个运送地址。顾客的联系地址可能并不是商品运送地址，特别当他在该网站上给别人买礼物的时候。
- 增加一个categories表并在books表中增加目录标识符（catid）。将图书分类会使网站更易浏览。
- 增加item_price列到order_items表，因为某项商品的价格可能改变。我们要知道顾客购买时候的商品价格。
- 增加一个admin表以保存管理员登录名和密码的详细信息。
- 删除评论表。可以以该项目扩展特性的形式增加评论功能。作为替代，每本书有一个描述域，该域对本书作简要的介绍。
- 将存储引擎修改为InnoDB。这样处理后，就可以使用外键，同时还可以在输入顾客订

单信息的时候使用事务。

要在系统中建立数据库，必须以root的身份在MySQL中运行book_sc.sql脚本，如下所示：

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

(需要提供root用户密码)

在进行以上操作之前，最好修改book_sc用户的密码，修改后的密码应该比原始密码"password"好。请注意，如果在book_sc.sql脚本中修改密码，还需要在db_fns.php脚本中修改它（稍后，我们将详细介绍）。

我们还要包含一个示例数据文件。该文件名为populate.sql。可以使用上述方法运行该sql语句将示例数据添加到数据库。

28.4 实现在线目录

在这个应用程序中，将使用3个目录脚本：主页、目录页和图书详细信息页。

本站点的首页由脚本index.php产生。该脚本输出如图28-3所示。

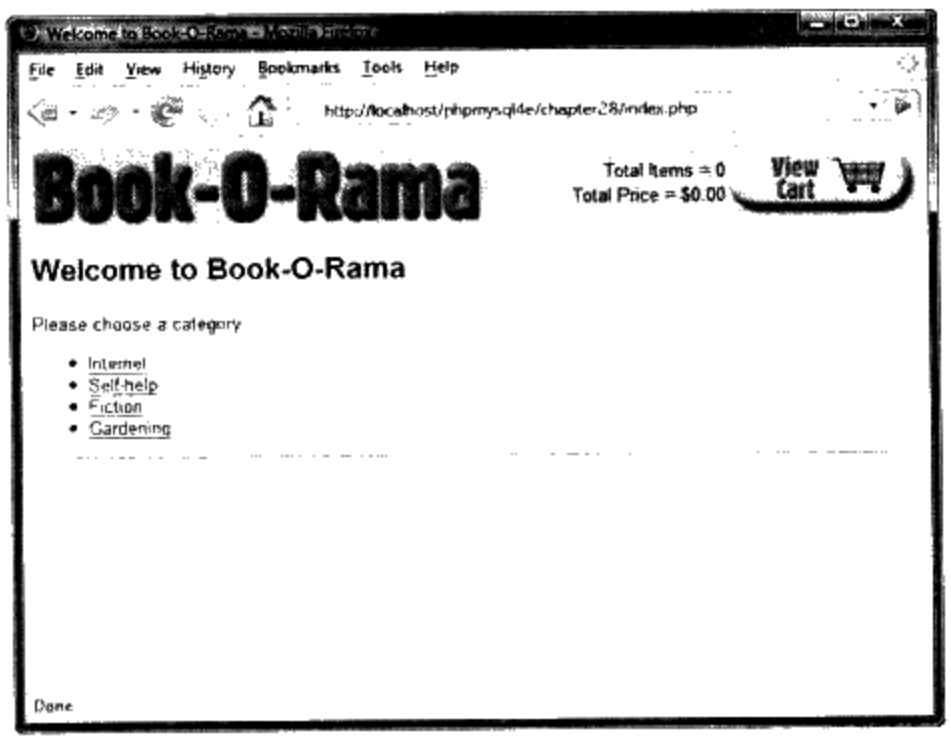

图28-3 该页列出了可在线购买图书的目录

注意在站点中，除了给出了一个目录列表外，屏幕的右上角还有一个购物车链接，以及对该购物车的简要描述。在用户浏览站点和购买商品的时候，它会一直显示在每一个页面上。

如果用户点击其中的一个目录链接，它将进入一个目录页面，该页由脚本show_cat.php产生。Internet图书的目录页面如图28-4所示。

所有包含在Internet目录的图书以链接形式列出。如果用户点击其中一个链接，将进入该书的详细信息页。一本书的详细信息页如图28-5所示。

在详细信息页中，除了“View Cart”链接外，我们还提供了“Add to Cart”链接，用户点击该链接可以选中一个物品。在讨论了如何创建购物车之后，我们再详细介绍它。

下面，让我们来了解这3个脚本。

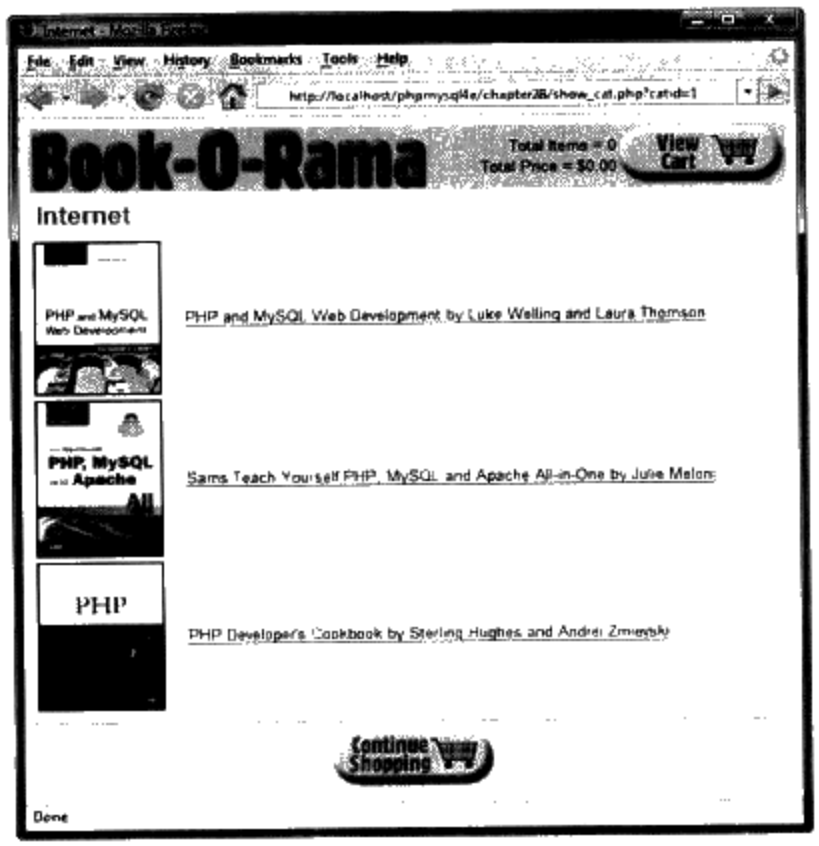

图28-4　列出目录中的每本书，旁边附有一张图片

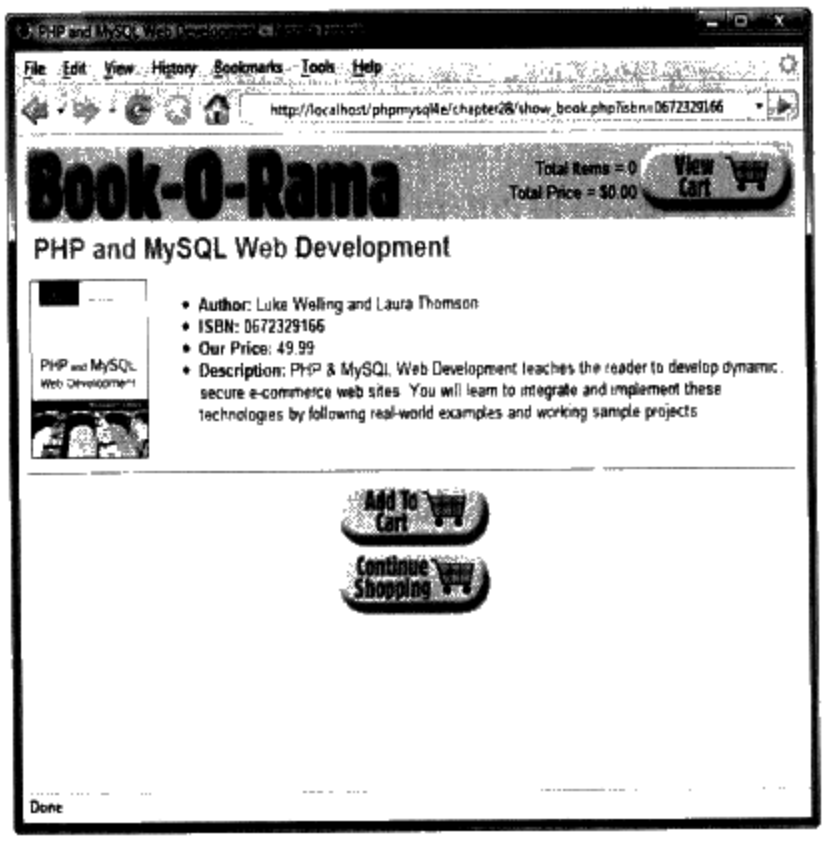

图28-5　每本书有一个详细信息页，该页详细地描述了该书

28.4.1　列出目录

在本项目中使用的第一个脚本——index.php列出了数据库中的所有目录。该脚本如程序清单28-2所示。

程序清单28-2　index.php——该脚本产生本站首页

```
<?php
  include ( book_sc_fns.php');
  // The shopping cart needs sessions, so start one
  session_start();
  do_html_header("Welcome to Book-O-Rama");

  echo "<p>Please choose a category:</p>";

  // get categories out of database
  $cat_array = get_categories();

  // display as links to cat pages
  display_categories($cat_array);

  // if logged in as admin, show add, delete, edit cat links
  if(isset($_SESSION[ admin_user ])) {
    display_button("admin.php", "admin-menu", "Admin Menu");
  }
  do_html_footer();
?>
```

在以上脚本的开始处，包含了book_sc_fns.php脚本，该文件包含该应用程序所有的函数库。

然后，我们必须开始一个会话。在购物车有了一个会话后，它才能正常工作。本网站中的每一页面都使用了会话。

index.php脚本还必须包含一些对HTML输出函数的调用，例如，do_html_header()和do_html_footer()（二者都包含在文件output_fns.php中）。该脚本还需要包含一些代码来检查用户是否以管理员的身份登录。如果是，就为它提供一些不同的导航选项。我们将在管理函数部分讨论它。

以上脚本最重要的部分是：

```
// get categories out of database
$cat_array = get_categories();

// display as links to cat pages
display_categories($cat_array);
```

函数get_categories()和display_categories()分别来自函数库book_fns.php和output_fns.php。函数get_categories()将返回系统中的一组目录，我们将该组目录传递到display_categories()函数。下面，我们来看一下get_categories()函数的代码，如程序清单28-3所示。

程序清单28-3 book_fns.php文件中的函数get_categories()——该函数从数据库中取回一个目录列表

```
function get_categories() {
   // query database for a list of categories
   $conn = db_connect();
   $query = "select catid, catname from categories";
   $result = @$conn->query($query);
   if (!$result) {
     return false;
   }
   $num_cats = @$result->num_rows;
   if ($num_cats == 0) {
      return false;
   }
   $result = db_result_to_array($result);
   return $result;
}
```

可以看到，该函数连接到数据库并获得所有目录的标识符（ID）和名称。我们已经编写和使用了名为db_result_to_array()的函数，该函数包含在db_fns.php中。其代码如程序清单28-4所示。它以一个MySQL搜索结果标识符作为参数，返回一组数字索引的行，其中每一行是一个相关数组。

程序清单28-4　db_fns.php文件中的db_result_to_array()函数——该函数将一个MySQL结果标识符转换为结果数组

```
function db_result_to_array($result) {
    $res_array = array();

    for ($count=0; $row = $result->fetch_assoc(); $count++) {
      $res_array[$count] = $row;
    }

    return $res_array;
}
```

在这个例子中，我们自始至终将这个数组返回给index.php，在该文件中，我们将该数组传递给output_fns.php的函数display_categories()。display_categories()函数以链接形式显示每个目录，此链接指包含该目录所有图书的页面。本函数源代码如程序清单28-5所示。

程序清单28-5　output_fns.php文件中的display_categories()函数——该函数以一列指向目录链接的形式显示一组目录

```
function display_categories($cat_array) {
  if (!is_array($cat_array)) {
    echo "<p>No categories currently available</p>";
    return;
  }
  echo "<ul>";
  foreach ($cat_array as $row) {
    $url = "show_cat.php?catid=".$row['catid'];
    $title = $row['catname'];
    echo "<li>";
    do_html_url($url, $title);
    echo "</li>";
  }
  echo "</ul>";
  echo "<br />";
}
```

该函数将数据库中的每一个目录转换成一个链接。每个链接导致下一个脚本的执行——show_cat.php——但是都有不同的参数，参数为目录的标识符或catid（这是一个唯一的标识数字，由MySQL生成，用以识别目录）。

这个将要传递给下一个脚本的参数将确定我们需要查看哪一个目录。

28.4.2　列出一个目录中的所有图书

列出一个目录中的所有图书的处理过程与前面类似。实现此功能的函数称为show_cat.php，

其源代码如程序清单28-6所示。

程序清单28-6 show_cat.php——该脚本显示特定目录中包含的所有图书

```
<?php
  include ( book_sc_fns.php');
  // The shopping cart needs sessions, so start one
  session_start();

  $catid = $_GET['catid'];
  $name = get_category_name($catid);

  do_html_header($name);

  // get the book info out from db
  $book_array = get_books($catid);

  display_books($book_array);


  // if logged in as admin, show add, delete book links
  if(isset($_SESSION['admin_user'])) {
    display_button("index.php", "continue", "Continue Shopping");
    display_button("admin.php", "admin-menu", "Admin Menu");
    display_button("edit_category_form.php?catid=".$catid,
                   "edit-category", "Edit Category");
  } else {
    display_button("index.php", "continue-shopping", "Continue Shopping");
  }

  do_html_footer();
?>
```

该脚本在结构上与index页面非常类似，不同的是，在这里，我们获取的是图书信息而不是目录信息。

与列出目录信息示例类似，我们调用session_start()函数开始一个会话，然后调用函数get_category_name()将目录标识符转换为目录名，如下所示：

```
$name = get_category_name($catid);
```

该函数从数据库中查询目录名。其代码如程序清单28-7所示。

程序清单28-7 book_fns.php文件中的get_category_name()函数——该函数将一个目录标识符转换为一个目录名

```
function get_category_name($catid) {
  // query database for the name for a category id
  $conn = db_connect();
  $query = "select catname from categories
```

```
            where catid = '".$catid." ';
    $result = @$conn->query($query);
    if (!$result) {
      return false;
    }
    $num_cats = @$result->num_rows;
    if ($num_cats == 0) {
       return false;
    }
    $row = $result->fetch_object();
    return $row->catname;
}
```

获得目录名称后，我们可以完成HTML标题并进而从数据库中获取和列出该选中目录的图书。如下所示：

```
$book_array = get_books($catid);
display_books($book_array);
```

函数get_books()和display_books()分别与函数get_categories()和display_categories()类似。因此，在这里我们将不再详细讨论。它们唯一的不同是前者从books表中获取信息，而后者是从categories表中获取。

display_books()函数通过show_book.php脚本为目录中的每一本书提供了链接。而且，每个链接后面都有一个参数。在本项目中，该参数是当前图书的ISBN。

在show_cat.php脚本的末尾，可以看到，如果是管理员登录的话，有一些函数将会显示一些附加的功能。我们将在管理函数部分讨论这些。

28.4.3　显示图书详细信息

show_book.php脚本将ISBN作为参数，取回并显示该书的详细信息。该脚本的代码如程序清单28-8所示。

程序清单28-8　show_book.php——该脚本显示特定图书的详细信息

```
<?php
  include ('book_sc_fns.php');
  // The shopping cart needs sessions, so start one
  session_start();

  $isbn = $_GET['isbn'];

  // get this book out of database
  $book = get_book_details($isbn);
  do_html_header($book['title']);
  display_book_details($book);

  // set url for 'continue button"
```

```
  $target = 'index.php';
  if($book['catid']) {
    $target = "show_cat.php?catid=".$book[ catid'];
  }

  // if logged in as admin, show edit book links
  if(check_admin_user()) {
    display_button("edit_book_form.php?isbn=".$isbn, "edit-item", "Edit Item");
    display_button("admin.php", "admin-menu", "Admin Menu");
    display_button($target, "continue", "Continue");
  } else {
    display_button("show_cart.php?new=".$isbn, "add-to-cart",
                   "Add".$book['title']." To My Shopping Cart");
    display_button($target, "continue-shopping", "Continue Shopping");
  }

  do_html_footer();
?>
```

对于这段脚本，我们还是做了与前两个脚本非常相似的操作。首先，我们将开始一个会话，然后使用如下语句：

```
$book = get_book_details($isbn);
```

从数据库中获取图书的信息，再使用如下语句：

```
display_book_details($book);
```

以HTML形式输出数据。

在这里，需要注意的是，display_book_details()函数将为每本书寻找一个图像，例如，images/".$book['isbn'].".jpg,。这里，该文件名称是该图书的ISBN号加上".jpg"的文件扩展名。如果该图像文件不存在，就不显示任何图像。脚本其余的部分将建立一个导航。普通用户将会有"Continue shopping"的选项，该选项将他们带回到目录页；此外还有"Add to Cart"选项，这将允许用户将图书添加到购物车。如果用户以管理员的身份登录，选项将有所不同，我们将在介绍管理部分时详细介绍它。

以上代码只是完成了目录系统的基本功能。下面，让我们继续讨论实现购物车功能。

28.5 实现购物车

购物车功能都是围绕名为cart的会话变量展开的。该变量是一个相关数组，它以ISBN作为主键，以图书的数量作为值。例如，如果在购物车中添加了一本图书，那么该数组将包含：

```
0672329166-> 1
```

也就是说，这是ISBN为0672329166的一本书。当我们将该书添加到购物车的时候，此信息也加到了该数组。当浏览购物车的时候，就是使用cart数组查看数据库中存储该物品的详细信息。

我们还使用了另外两个会话变量来控制标题栏的显示，该标题栏显示了全部物品数和总价格。这两个变量分别是items和total_price。

28.5.1 使用show_cart.php脚本

通过了解show_cart.php脚本，可以明白购物车代码是如何实现其功能的。如果我们点击“View Cart”或“Add to Cart”链接的话，show_cart.php脚本将显示我们要访问的页面。如果不使用任何参数来调用show_cart.php，将看到购物车的内容；如果用一个ISBN作为参数，该ISBN对应的物品被将添加到购物车中。

要完全理解这些，请参阅图28-6。

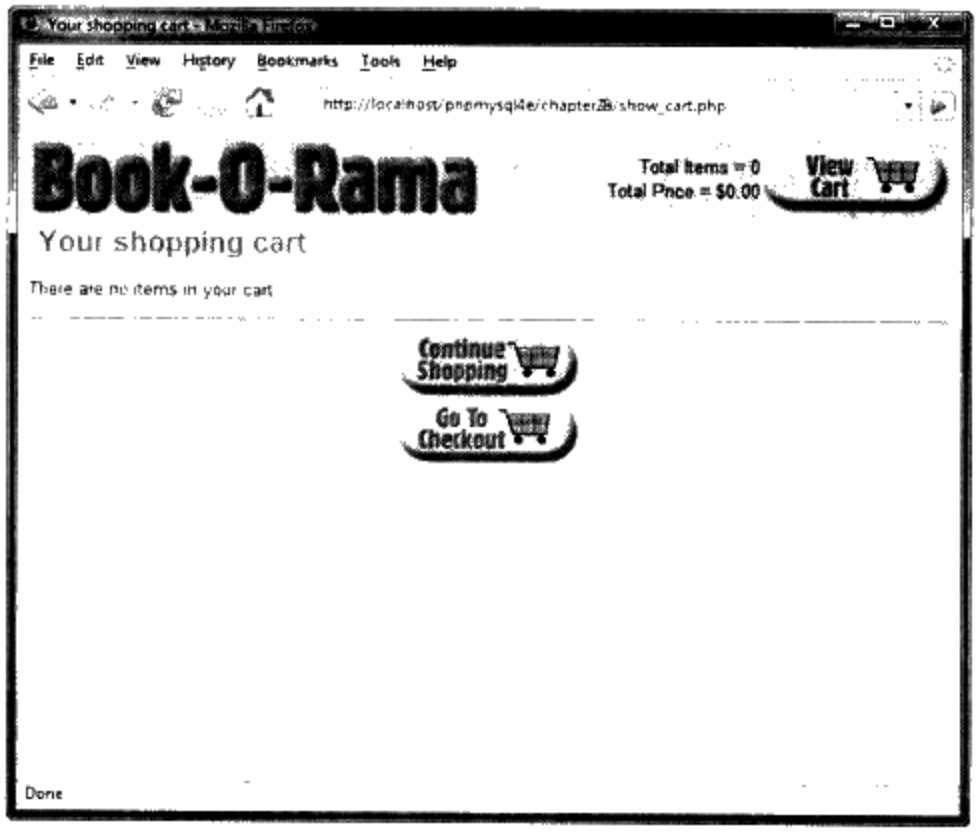

图28-6　不使用参数，show_cart.php脚本只显示购物车的内容

在这个例子中，我们点击了“View Cart”链接，此时购物车为空；也就是说，我们还没有选中任何要买的物品。

图28-7进一步显示了用户的购物车的记录，此时我们已经选中了两本要买的书。在这种情况下，我们是通过点击该书在show_book.php页面上的“Add to Cart”链接而进入本页面的，该书为《PHP和MySQL Web开发》。如果仔细查看URL地址栏，可以看到这次我们使用了一个参数来调用该脚本。该参数为new，其值为0672329760——即刚刚添加到购物车的图书的ISBN。

从这个页面，可以看到我们已经多了两个其他选项。一个是“Save Changes”按钮，它可以用来修改购物车中物品的数量。要修改数量，我们可直接改变物品的数量并点击“Save Changes”按钮。它实际上是一个提交按钮，可以将我们带到show_cart.php脚本以更新购物车。

除此之外，该页面中还有一个“Go To Checkout”按钮，当用户准备离开的时候，可以点击此按钮。稍后，我们将详细介绍它。

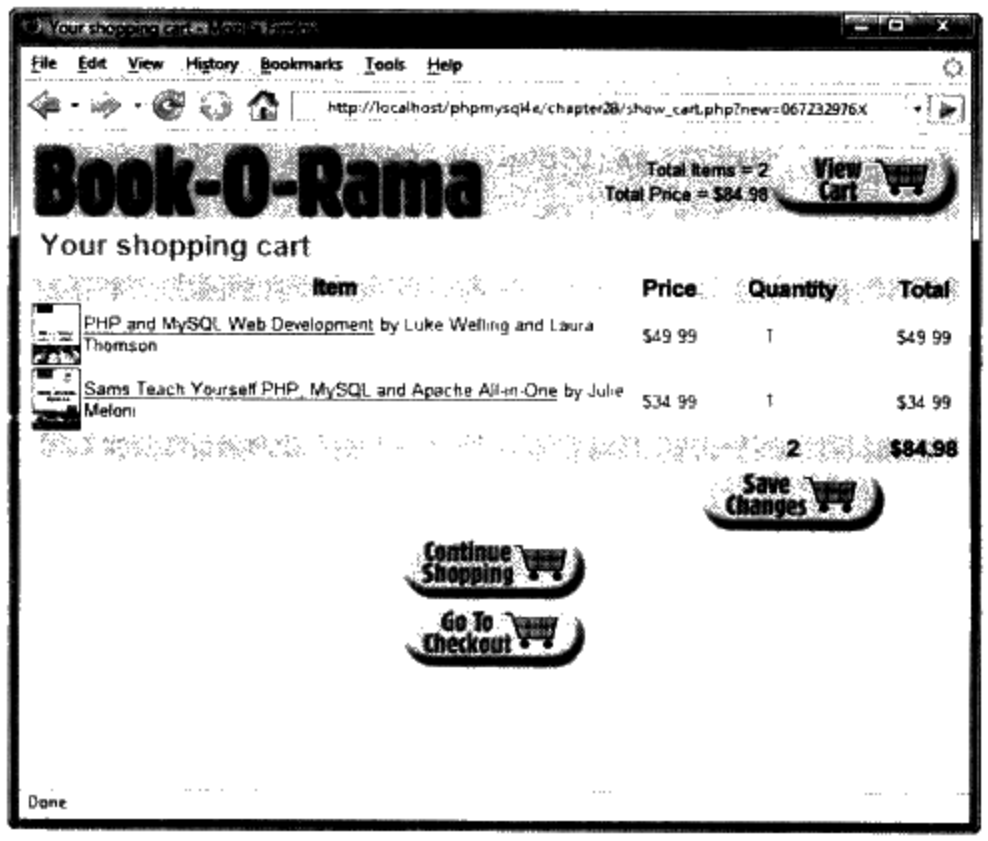

图28-7 带有参数new的show_cart.php脚本将添加一个物品到购物车

从现在开始，我们来了解show_cart.php脚本的代码。该脚本如程序清单28-9所示。

程序清单28-9 show_cart.php——该脚本控制购物车

```
<?php
  include ('book_sc_fns.php');
  // The shopping cart needs sessions, so start one
  session_start();

  @$new = $_GET['new'];

  if($new) {
    //new item selected
    if(!isset($_SESSION['cart'])) {
      $_SESSION['cart'] = array();
      $_SESSION['items'] = 0;
      $_SESSION['total_price'] ='0.00';
    }

    if(isset($_SESSION['cart'][$new])) {
      $_SESSION['cart'][$new]++;
    } else {
      $_SESSION['cart'][$new] = 1;
    }

    $_SESSION['total_price'] = calculate_price($_SESSION['cart']);
    $_SESSION['items'] = calculate_items($_SESSION['cart']);
```

```
}

if(isset($_POST['save'])) {
  foreach ($_SESSION['cart'] as $isbn => $qty) {
    if($_POST[$isbn] == '0') {
      unset($_SESSION['cart'][$isbn]);
    } else {
      $_SESSION[ cart'][$isbn] = $_POST[$isbn];
    }
  }

  $_SESSION[ total_price'] = calculate_price($_SESSION[ cart']);
  $_SESSION[ items ] = calculate_items($_SESSION['cart']);
}

do_html_header('Your shopping cart");

if(($_SESSION[ cart']) && (array_count_values($_SESSION['cart']))) {
  display_cart($_SESSION[ cart']);
} else {
  echo "<p>There are no items in your cart</p><hr/>";
}

$target = 'index.php';
  // if we have just added an item to the cart, continue shopping in that category
  if($new) {
    $details = get_book_details($new);
    if($details['catid']) {
      $target = "show_cat.php?catid-'.$details[ catid ];
    }
  }
  display_button($target, 'continue-shopping', "Continue Shopping");

  // use this if SSL is set up
  // $path = $_SERVER['PHP_SELF'];
  // $server = $_SERVER['SERVER_NAME ];
  // $path = str_replace('show_cart.php , ' , $path);
  // display_button('https://'.$server.$path."checkout.php",
  //                "go-to-checkout", 'Go To Checkout");

  // if no SSL use below code
  display_button("checkout.php", "go-to-checkout", 'Go To Checkout');

  do_html_footer();
?>
```

该脚本主要由3个部分组成：显示购物车、添加物品到购物车以及保存购物车的修改结果。接下来，我们将在下面3个小节里分别讨论它们。

28.5.2 浏览购物车

不论我们从哪个页面进入，都需要显示购物车的内容。在这个最基本的例子中，当用户点击“View Cart”按钮后，系统将只执行如下所示的代码：

```
if(($_SESSION['cart']) && (array_count_values($_SESSION['cart']))) {
  display_cart($_SESSION['cart']);
} else {
  echo "<p>There are no items in your cart</p><hr/>";
}
```

可以从这段代码看出，如果购物车里有一些物品，我们就调用display_cart()函数。如果购物车是空的，会给用户发送一条消息，告诉他们购物车为空。

display_cart()函数只以可读的HTML格式打印出购物车的内容，这点可从图28-6和图28-7看出。尽管函数的代码可以在output_fns.php中找到，但我们还是给出了output_fns.php的源代码，如程序清单28-10所示。尽管该函数只是一个显示函数，但是它比较复杂，因此在这里我们讨论一下。

程序清单28-10 output_fns.php文件中的display_cart()函数——该函数将格式化并打印购物车的内容

```
function display_cart($cart, $change = true, $images = 1) {
  // display items in shopping cart
  // optionally allow changes (true or false)
  // optionally include images (1 - yes, 0 - no)

   echo '<table border=\"0\" width=\"100%\" cellspacing=\"0\">
         <form action=\"show_cart.php\" method=\"post\">
         <tr><th colspan=\"".(2 + $images).'\" bgcolor=\"#cccccc\">Item</th>
         <th bgcolor=\"#cccccc\">Price</th>
         <th bgcolor=\"#cccccc\">Quantity</th>
         <th bgcolor=\"#cccccc\">Total</th>
         </tr>";

  //display each item as a table row
  foreach ($cart as $isbn => $qty) {
    $book = get_book_details($isbn);
    echo "<tr>";
    if($images == true) {
      echo '<td align=\"left\">";
      if (file_exists("images/'.$isbn.'.jpg')) {
        $size = GetImageSize("images/".$isbn.".jpg");
        if(($size[0] > 0) && ($size[1] > 0)) {
          echo '<img src=\"images/".$isbn.'.jpg\"
```

```
                    style=\"border: 1px solid black\"
                    width=\"".($size[0]/3)."\"
                    height=\"".($size[1]/3)."\"/>";
            }
        } else {
            echo ' ";
        }
        echo "</td>";
    }
    echo "<td align=\'left\">
          <a href=\'show_book.php?isbn=".$isbn."\'>".$book['title']."</a>
          by ".$book['author']."</td>
          <td align=\"center\">\$".number_format($book['price'], 2)."</td>
          <td align=\"center\">";

    // if we allow changes, quantities are in text boxes
    if ($change == true) {
      echo "<input type=\"text\" name=\"".$isbn."\" value=\"".$qty."\"
size=\'3\">";
    } else {
      echo $qty;
    }
    echo "</td>
          <td align=\'center\">\$".number_format($book['price']*$qty,2)."</td>
        </tr>\n";
}
// display total row
echo '<tr>
      <th colspan=\"".(2+$images)."\" bgcolor-\"#cccccc\"> </td>
      <th align=\"center\" bgcolor-\"#cccccc\'>".$_SESSION[ items'].'</th>
      <th align=\'center\" bgcolor=\'#cccccc\'>
          \$".number_format($_SESSION['total_price'], 2)."
      </th>
      </tr>';

// display save change button
if($change == true) {
  echo "<tr>
        <td colspan-\"".(2+$images).'\"> </td>
        <td align=\'center\">
           <input type-\"hidden\" name=\"save\' value=\"true\"/>
           <input type=\"image\" src=\'images/save-changes.gif\"
                  border=\'0\" alt=\"Save Changes\"/>
        </td>
        <td> </td>
        </tr>";
```

```
    echo '</form></table>';
}
```

该函数的基本流程如下所示：

1. 对购物车中所有物品执行循环，将每个物品的ISBN传递给get_book_details()函数，这样，可以总结每本书的详细信息。

2. 如果有图片存在，可以为每本书提供一个图片。我们使用HTML图像的高度标记和宽度标记将图片改得小了一点。这意味着这些图片可能有一些扭曲，但是因为它们很小，扭曲一点不会有太大问题。（如果看起来还是不舒服，可以使用第22章“创建图像”中介绍的gd库来修改它的大小，或者手动为每个产品生成一个不同大小的图片。）

3. 让每个购物车包含一个指向适当图书的链接，也就是以ISBN作为show_book.php的参数显示图书。

4. 如果以change作参数调用函数，而且该change设置为true（或不设置——其默认值是true），则看到的是以表单形式显示的购物车中所有物品的数量，点击“Save Changes”按钮结束（在结账之后再使用该函数的时候，我们希望用户不能修改订单）。

该函数中没有特别复杂的逻辑，但是它完成了许多操作，因此，仔细阅读它，会觉得它非常有用。

28.5.3 将物品添加到购物车

如果用户点击“Add to Cart”按钮进入show_cart.php页面，在显示其购物车内容之前，我们要做一些工作。特别地，需要将适当的物品添加到购物车中，如下所示。

首先，如果该用户此前没有在购物车中添加任何物品，那么该用户没有一个购物车，需要为其创建一个购物车：

```
if(!isset($_SESSION['cart'])) {
  $_SESSION['cart'] = array();
  $_SESSION['items'] = 0;
  $_SESSION['total_price'] = '0.00';
}
```

初始状态下，购物车是空的。

其次，建立了一个购物车后，可以将物品添加到购物车内：

```
if(isset($_SESSION['cart'][$new])) {
  $_SESSION['cart'][$new]++;
} else {
  $_SESSION['cart'][$new] = 1;
}
```

在这里，我们检查了该物品是否已经在购物车中存在，如果是，我们将该物品的数量增1，否则，添加新物品到购物车。

再次，我们必须计算购物车中所有物品的总价格和数量。要完成这些操作，我们使用了calculate_price()函数和calculate_items()函数，如下所示：

```
$_SESSION['total_price'] = calculate_price($_SESSION['cart']);
$_SESSION['items'] = calculate_items($_SESSION['cart']);
```

这些函数位于book_fns.php函数库中。其代码如程序清单28-11和程序清单28-12所示。

程序清单28-11　book_fns.php文件中的calculate_price()函数——计算和返回购物车中物品的总价格

```
function calculate_price($cart) {
  // sum total price for all items in shopping cart
  $price = 0.0;
  if(is_array($cart)) {
    $conn = db_connect();
    foreach($cart as $isbn => $qty) {
      $query = "select price from books where isbn='".$isbn."'";
      $result = $conn->query($query);
      if ($result) {
        $item = $result->fetch_object();
        $item_price = $item->price;
        $price +=$item_price*$qty;
      }
    }
  }
  return $price;
}
```

可以看到，calculate_price()函数工作的时候查询了数据库中每个物品的价格。该操作可能会有些慢，因此如果没有必要，应该避免这样的计算。我们将保存此价格（还有物品总数）到会话变量中，当购物车改变的时候才重新计算。

程序清单28-12　book_fns.php文件中的calculate_items()函数——该函数计算并返回购物车中物品的总数

```
function calculate_items($cart) {
  // sum total items in shopping cart
  $items = 0;
  if(is_array($cart)) {
    foreach($cart as $isbn => $qty) {
      $items += $qty;
    }
  }
  return $items;
}
```

calculate_items()函数相当简单；它只是扫描了购物车，使用array_sum()函数将每个物品的数量加起来得到总物品数量。如果还没有数组（或者购物车为空），它将返回0。

28.5.4 保存更新后的购物车

如果我们通过点击“Save Changes”按钮进入show_cart.php脚本，那么它的处理过程与前面有些不同。在这种情况下，我们是通过提交一个表单进入的。如果仔细查看这些代码，可以发现“Save Changes”按钮是一个表单的提交按钮。该表单包含了隐含变量save。如果该变量被赋了值，我们就可以知道是从“Save Changes”按钮的点击事件进入此页面的。这就意味着，该用户可能已经编辑了购物车中物品的数量值，我们需要更新它们。

在output_fns.php脚本中，如果查看display_cart()函数“Save Changes”表单部分的文本框，你将发现它们的命名是与ISBN相关联的，如下所示：

```
echo "<input type=\"text\" name=\"".$isbn."\" value=\"".$qty."\" size=\"3\">";
```

现在，让我们来看看保存修改部分的脚本：

```
if(isset($_POST['save'])) {
  foreach ($_SESSION['cart'] as $isbn -> $qty) {
    if($_POST[$isbn] == '0') {
      unset($_SESSION['cart'][$isbn]);
    } else {
      $_SESSION['cart'][$isbn] = $_POST[$isbn];
    }
  }

  $_SESSION['total_price'] = calculate_price($_SESSION['cart']);
  $_SESSION['items'] = calculate_items($_SESSION['cart']);
}
```

可以看出，我们通过购物车完成了工作，对购物车中的每一个isbn，检查了该变量名称的POST变量。这些都是“Save Changes”表单的表单域。

如果将任何一个域设置为0，我们将使用unset()函数将购物车中该物品完全删除。否则，就更新该购物车，使之与该表单域匹配，如下所示：

```
if($_POST[$isbn] == '0') {
  unset($_SESSION['cart'][$isbn]);
} else {
  $_SESSION['cart'][$isbn] = $_POST[$isbn];
}
```

在完成购物车更新之后，我们又调用calculate_price()函数和calculate_items()函数计算出会话变量total_price和items的新值。

28.5.5 打印标题栏摘要

网站中每一页的标题栏都简要总结了购物车中的物品。这是通过打印会话变量total_price和items的值来实现的。函数do_html_header()能够完成此功能。

当用户第一次访问show_cart.php页面的时候，我们就注册了这些变量。我们还需要一些代码来解决用户未访问该页的情形。这个逻辑同样包含在do_html_header()函数中，如

下代码所示：

```
if (!$_SESSION[ items ]) {
  $_SESSION['items'] = '0';
}
if (!$_SESSION['total_price']) {
  $_SESSION['total_price'] = '0.00';
}
```

28.5.6 结账

当用户点击购物车的“Go to Checkout”按钮时，将触发checkout.php脚本。该结账脚本及其后面的脚本必须通过SSL访问到，但是，我们给出的示例应用程序并不要求这么做（要了解更多关于SSL的信息，请参阅第18章“通过PHP和MySQL实现安全事务”）。

结账页面如图28-8所示。

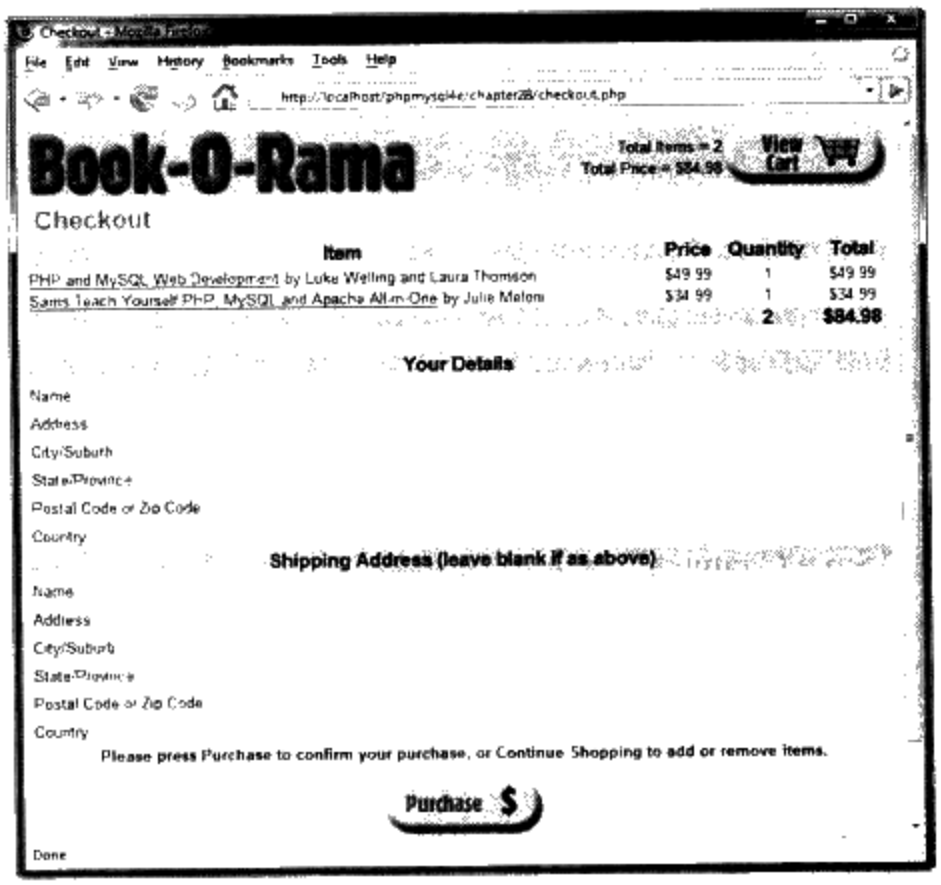

图28-8　checkout.php脚本获取顾客的详细信息

结账脚本要求顾客输入地址（如果运送地址与该地址不同，需要输入运送地址）。该脚本非常简单，其代码如程序清单28-13所示。

程序清单28-13　checkout.php——该脚本获取用户的详细信息

```
<?php
  //include our function set
  include ('book_sc_fns.php');

  // The shopping cart needs sessions, so start one
```

```
session_start();

do_html_header("Checkout");

if(($_SESSION['cart']) && (array_count_values($_SESSION['cart']))) {
  display_cart($_SESSION['cart'], false, 0);
  display_checkout_form();
} else {
  echo "<p>There are no items in your cart.</p>";
}

display_button("show_cart.php", "continue-shopping", "Continue Shopping");

do_html_footer();
?>
```

在以上脚本中，并没有什么奇怪的事情发生。如果购物车是空的，脚本将通知该用户；否则，它将显示如图28-8所示的表单。

如果用户继续点击该表格下面的“Purchase”按钮，将进入purchase.php脚本。在图28-9中，可以看到该脚本的输出。

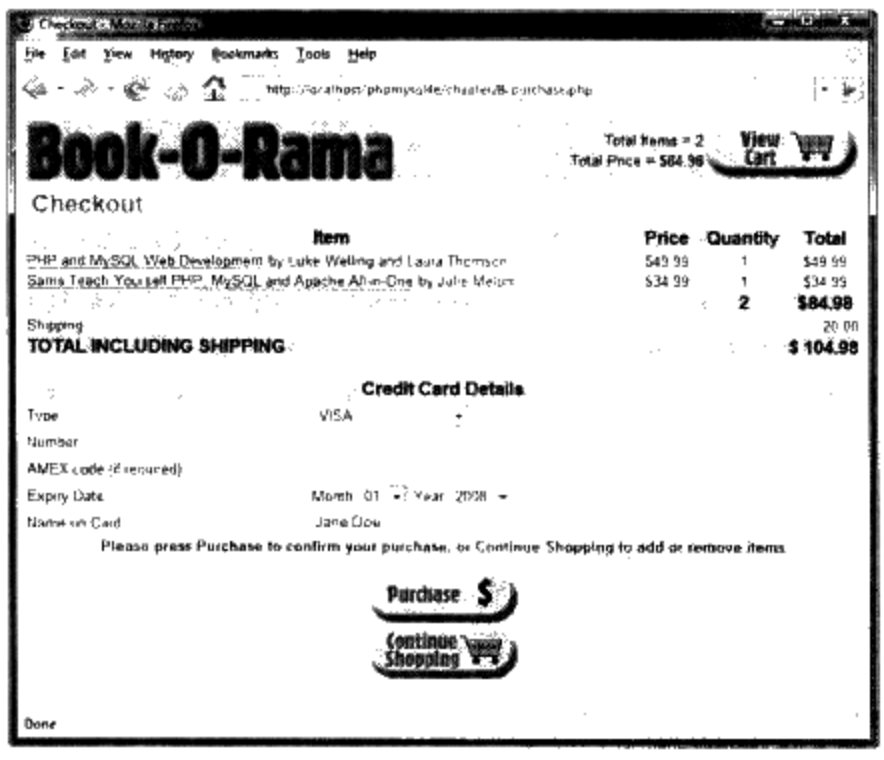

图28-9 purchase.php脚本计算客户购买的商品数量和最终的订单价格，并且获得客户的付款详细信息

以上脚本代码比checkout.php脚本逻辑稍微复杂一些，如程序清单28-14所示。

程序清单28-14 purchase.php——该脚本将用户订单的详细信息保存到数据库中并获取付款细节

```
<?php

  include ('book_sc_fns.php');
```

```
// The shopping cart needs sessions, so start one
session_start();

do_html_header("Checkout");

// create short variable names
$name = $_POST['name'];
$address = $_POST['address'];
$city = $_POST['city'];
$zip = $_POST['zip'];
$country = $_POST['country'];

// if filled out
if (($_SESSION['cart']) && ($name) && ($address) && ($city)
        && ($zip) && ($country)) {
  // able to insert into database
  if(insert_order($_POST) != false ) {
    //display cart, not allowing changes and without pictures
    display_cart($_SESSION['cart'], false, 0);

    display_shipping(calculate_shipping_cost());

    //get credit card details
    display_card_form($name);
    display_button('show_cart.php', 'continue-shopping', 'Continue Shopping');
  } else {
    echo "<p>Could not store data, please try again.</p>";
    display_button('checkout.php', 'back', 'Back');
  }
} else {
  echo '<p>You did not fill in all the fields, please try again.</p><hr />';
  display_button('checkout.php', 'back', 'Back');
}

do_html_footer();
?>
```

以上代码的逻辑简单明了：它首先检查用户是否填好了表单，并调用insert_order()函数，这是一个简单的函数，将用户填写的详细信息插入数据库。其代码如程序清单28-15所示。

该函数相当长，因为我们需要插入用户所有的细节，包括订单细节以及他们要买的每一本书的细节。

程序清单28-15　order_fns.php文件中的insert_order()函数——该函数将所有的用户订单详细信息插入到数据库中

```
<?php
function process_card($card_details) {
```

```
  // connect to payment gateway or
  // use gpg to encrypt and mail or
  // store in DB if you really want to

  return true;
}

function insert_order($order_details) {
  // extract order_details out as variables
  extract($order_details);

  // set shipping address same as address
  if((!$ship_name) && (!$ship_address) && (!$ship_city)
      && (!$ship_state) && (!$ship_zip) && (!$ship_country)) {
    $ship_name = $name;
    $ship_address = $address;
    $ship_city = $city;
    $ship_state = $state;
    $ship_zip = $zip;
    $ship_country = $country;
  }

  $conn = db_connect();
// we want to insert the order as a transaction
// start one by turning off autocommit
$conn->autocommit(FALSE);

// insert customer address
$query = "select customerid from customers where
          name = '".$name."' and address = '".$address."'
          and city = '".$city."' and state = '".$state."'
          and zip = '".$zip."' and country = '".$country."'";

$result = $conn->query($query);

if($result->num_rows>0) {
  $customer = $result->fetch_object();
  $customerid = $customer->customerid;
} else {
  $query = "insert into customers values
           ('', '".$name."','".$address."', '".$city."',
            '".$state."','".$zip."','".$country."')";
  $result = $conn->query($query);

  if (!$result) {
    return false;
```

```
    }
}

$customerid = $conn->insert_id;

$date = date("Y-m-d");

$query = "insert into orders values
          ('', '".$customerid."', '".$_SESSION['total_price']."',
           '".$date."', '".PARTIAL."', '".$ship_name."',
           '".$ship_address."', '".$ship_city."',
           '".$ship_state."', '".$ship_zip."',
           '".$ship_country."')";

$result = $conn->query($query);
if (!$result) {
  return false;
}

$query = "select orderid from orders where
              customerid = '".$customerid."' and
              amount > (".$_SESSION['total_price']."-.001) and
              amount < (".$_SESSION['total_price']."+.001) and
              date = '".$date."' and
              order_status = 'PARTIAL' and
              ship_name = '".$ship_name."' and
              ship_address = '".$ship_address."' and
              ship_city = '".$ship_city."' and
              ship_state = '".$ship_state."' and
              ship_zip = '".$ship_zip."' and
              ship_country = '".$ship_country."' ";

$result = $conn->query($query);

if($result->num_rows>0) {
  $order = $result->fetch_object();
  $orderid = $order->orderid;
} else {
  return false;
}

// Insert each book
foreach($_SESSION['cart'] as $isbn => $quantity) {
  $detail = get_book_details($isbn);
  $query = "delete from order_items where
            orderid = '".$orderid."' and isbn = '".$isbn."'";
  $result = $conn->query($query);
  $query = "insert into order_items values
```

```
                ('".$orderid."', '".$isbn."', ".$detail['price'].", $quantity)";
        $result = $conn->query($query);
        if(!$result) {
          return false;
        }
      }

      // end transaction
      $conn->commit();
      $conn->autocommit(TRUE);

      return $orderid;
    }

    ?>
```

插入操作的不同部分被包括在一个事务中，它以如下语句开始：

```
$conn->autocommit(FALSE);
```

以如下语句结束：

```
$conn->commit();
$conn->autocommit(TRUE);
```

这是应用程序中唯一需要使用事务的地方。如何避免在其他情况下也使用它呢？请查看db_connect()函数的代码，如下所示：

```
function db_connect() {
    $result = new mysqli('localhost', 'book_sc', 'password', 'book_sc');
    if (!$result) {
       return false;
    }
    $result->autocommit(TRUE);
    return $result;
}
```

很明显，以上代码与在其他章节中使用的相同函数名称的代码不同。在创建了到MySQL的连接后，必须开启自动提交模式。

正如我们前面介绍的，这可以确保每一个SQL语句将自动被提交。当需要使用一个多语句事务时，必须关闭自动提交模式，执行一系列的插入操作，提交数据，然后再重新启用自动提交模式。

然后我们根据顾客地址计算出运送费用，并通过如下所示代码告诉他们需要的费用：

```
display_shipping(calculate_shipping_cost());
```

在这里，我们使用的calculate_shipping_cost()函数将始终返回$20。当真正建立一个购物网站的时候，必须选择一个送货方式，计算出不同的目的地需要的运送费用，并据此计算出总费用。

接下来，我们将向每个用户显示一个表单，用来获得信用卡详细信息，此过程中，我们调用了output_fns.php函数库中的display_card_form()函数。

28.6 实现付款

当用户点击“Purchase”按钮时，我们将调用process.php脚本来处理付款细节。可以在图28-10中看到成功付款的结果。

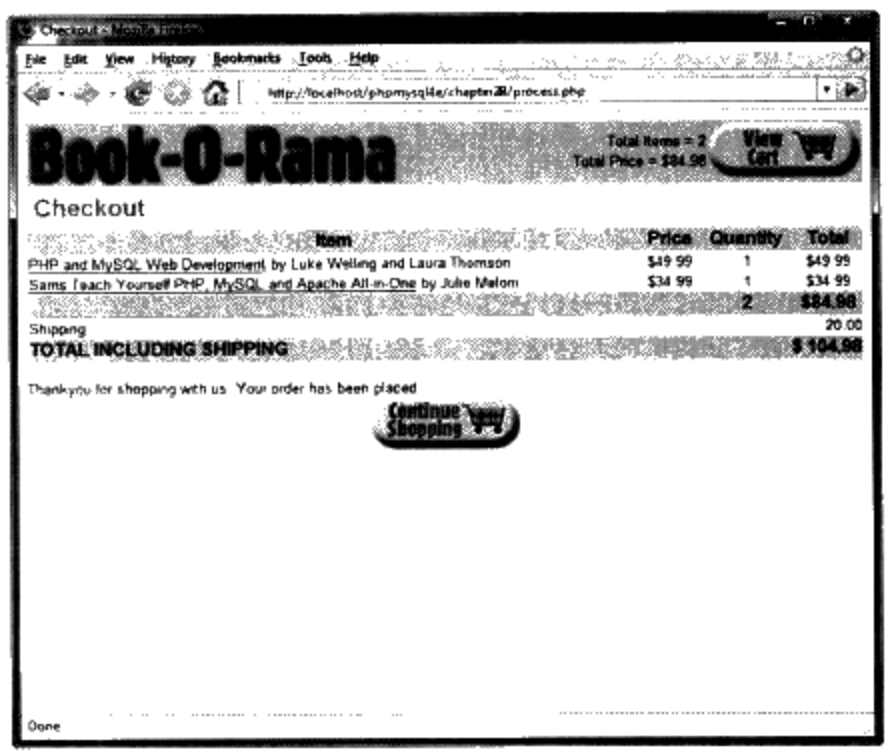

图28-10　付款操作成功，我们将发送顾客订购的商品

process.php脚本如程序清单28-16所示。

我们将处理用户的信用卡，如果所有操作成功完成，再销毁用户的会话。

我们编写的信用卡处理函数将简单地返回true。在具体实现过程中，可能需要先执行一些验证（检查信用卡的过期日期是否仍然有效以及卡号是否正确），然后再执行真正的付款操作。

程序清单28-16　process.php——该脚本处理顾客的付款并显示处理结果

```
<?php
  include ('book_sc_fns.php');
  // The shopping cart needs sessions, so start one
  session_start();

  do_html_header('Checkout');

  $card_type = $_POST['card_type'];
  $card_number = $_POST['card_number'];
  $card_month = $_POST['card_month'];
  $card_year = $_POST['card_year'];
  $card_name = $_POST['card_name'];

  if(($_SESSION['cart']) && ($card_type) && ($card_number) &&
```

```
    ($card_month) && ($card_year) && ($card_name)) {
    //display cart, not allowing changes and without pictures
    display_cart($_SESSION['cart'], false, 0);

    display_shipping(calculate_shipping_cost());

    if(process_card($_POST)) {
      //empty shopping cart
      session_destroy();
      echo "<p>Thank you for shopping with us. Your order has been placed.</p>";
      display_button("index.php", "continue-shopping", "Continue Shopping");
    } else {
      echo '<p>Could not process your card. Please contact the card
            issuer or try again.</p>";
      display_button("purchase.php", "back", "Back");
    }
  } else {
    echo "<p>You did not fill in all the fields, please try again.</p><hr />";
    display_button("purchase.php", "back", "Back");
  }

  do_html_footer();
?>
```

实际建立站点时，必须决定采用哪种事务清除机制。我们可以：

- 与一个事务清理提供商签约。根据所在地的不同，可以选择许多服务商。一些服务商提供实时的数据清理，一些服务商则不是。是否需要实时的数据清理取决于向用户提供的服务。如果提供的是在线服务，可能最希望实时清理；如果发送货物，实时处理就不是很重要了。无论选择哪种方式，这些服务商都承担了保存信用卡号码的责任。
- 通过加密邮件将信用卡号码发送给自己，加密算法可以采用PGP或GPG等，这些内容已经在第18章中详细介绍。当接收并解密这些邮件之后，就可以手动处理这些事务。
- 在数据库中保存信用卡号。如果没有慎重考虑这样做对系统安全性能的潜在危害，那么我们不推荐采用这种选择。可以参阅第18章的详细介绍，明白为什么这是一个糟糕的主意。

以上就是购物车模块和付款模块。

28.7 实现一个管理界面

我们实现的管理界面非常简单。创建一个网站界面，该页面可以连接数据库，并且执行一些前台的身份验证。这些代码与第27章中用到的代码非常类似。出于完整性考虑，我们还是在这里给出了这些代码，但对它不做具体的介绍。

管理界面需要用户通过`login.php`文件进行登录，该文件会将用户带入到管理菜单`admin.php`中。登录界面如图28-11所示。（为了简洁起见，我们没有给出`login.php`的代码——该文件与第27章中介绍的几乎一样，如果要查看这个文件，可以在附带的文件中找到

这个文件。）管理菜单如图28-12所示。

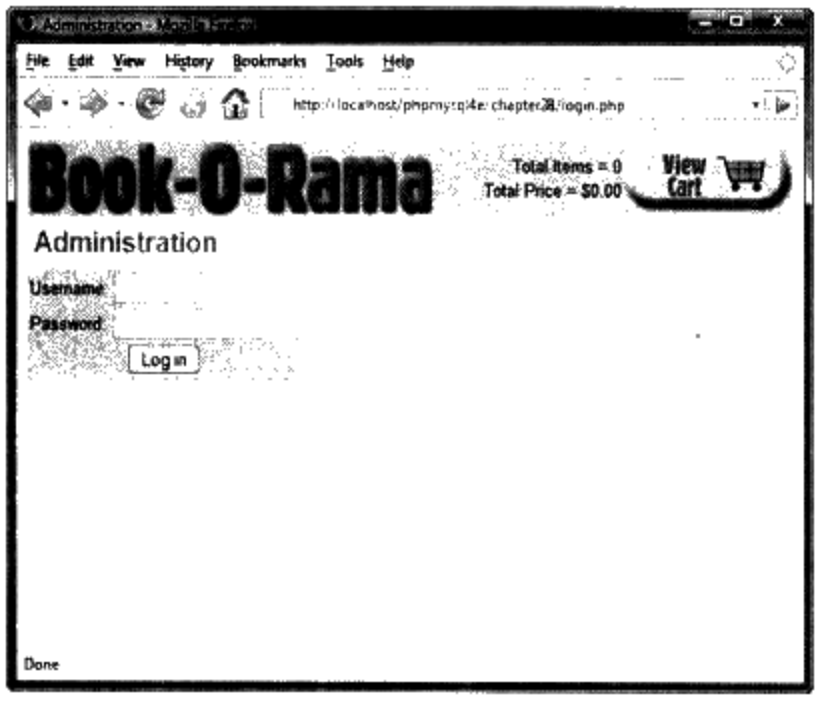

图28-11　用户通过登录页才能访问管理功能

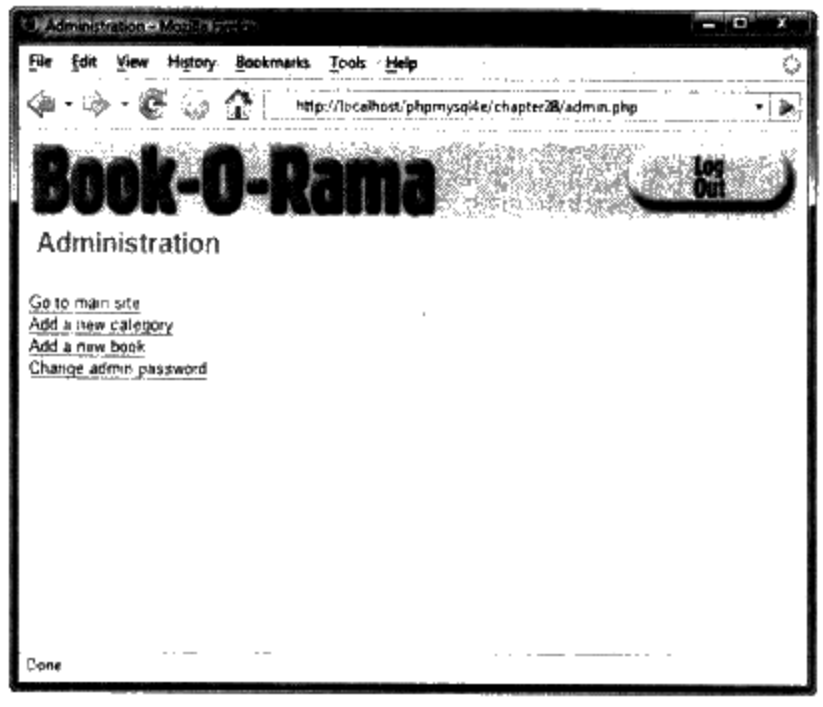

图28-12　管理菜单允许用户访问管理函数

管理菜单的代码如程序清单28-17所示。

程序清单28-17　admin.php——该脚本将验证管理员身份并允许他访问管理功能

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


if (($_POST['username']) && ($_POST['passwd'])) {
    // they have just tried logging in

    $username = $_POST['username'];
    $passwd = $_POST['passwd'];

    if (login($username, $passwd)) {
      // if they are in the database register the user id
      $_SESSION['admin_user'] = $username;

  } else {
     // unsuccessful login
     do_html_header('Problem:');
     echo '<p>You could not be logged in.<br/>
           You must be logged in to view this page.</p>';
     do_html_url('login.php', 'Login');
     do_html_footer();
     exit;
```

```
    }
}

do_html_header('Administration');
if (check_admin_user()) {
  display_admin_menu();
} else {
  echo '<p>You are not authorized to enter the administration area.</p>';
}
do_html_footer();
?>
```

这段代码看起来有点熟悉；它与第27章中的一段脚本类似。管理员到达这个页面之后，可以修改密码或者退出——该代码与第27章中的代码完全一样，因此在这里不再介绍。

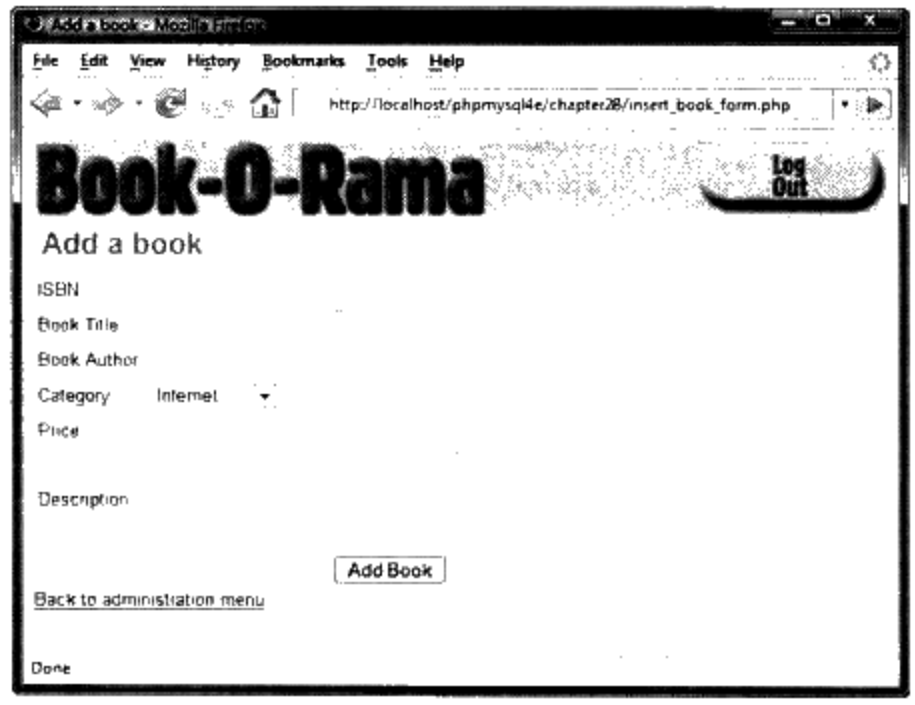

图28-13 该表单允许管理员输入新书到在线目录

我们在管理员登录之后通过会话变量admin_user和check_admin_user()函数来识别其身份。该函数和其他被管理员脚本调用的函数一样，都可以在函数库admin_fns.php中找到。

如果管理员选择添加一个新的目录或图书，可以根据具体情况进入insert_category_form.php或insert_book_form.php文件。每个脚本都会向管理员提供一个表单，该表单必须由管理员填写。每个表单都由相应的脚本来处理（insert_category.php和insert_book.php），这些脚本将检查表单是否填好并将数据插入到数据库中。在这里，我们只讨论添加图书的脚本，因为添加图书的脚本和添加目录的脚本类似。

insert_book_form.php脚本的输出结果如图28-13所示。

注意图书的目录域是一个HTML的SELECT元素。SELECT的选项来自对get_categories()函数的调用，前面我们已经介绍了该函数。

点击“Add Book”按钮之后，将触发insert_book.php脚本的执行。该脚本的代码如程序清单28-18所示。

程序清单28-18 insert_book.php——该脚本将验证新书数据并将它添加到数据库

```
<?php

// include function files for this application
require_once('book_sc_fns.php');
```

```
session_start();

do_html_header("Adding a book");
if (check_admin_user()) {
  if (filled_out($_POST)) {
    $isbn = $_POST['isbn'];
    $title = $_POST['title'];
    $author = $_POST| author'];
    $catid = $_POST['catid'];
    $price = $_POST['price'];
    $description = $_POST['description'];

    if(insert_book($isbn, $title, $author, $catid, $price, $description)) {
      echo "<p>Book <em>".stripslashes($title).'</em> was added to the
           database.</p>";
    } else {
      echo "<p>Book <em>".stripslashes($title)."</em> could not be
           added to the database.</p>";
    }
  } else {
    echo '<p>You have not filled out the form. Please try again.</p>';
  }

  do_html_url("admin.php", "Back to administration menu");
} else {
  echo "<p>You are not authorised to view this page.</p>";
}

do_html_footer();
?>
```

可以看到，该脚本调用了函数insert_book()。该函数和其他管理脚本调用的函数一样，都可以在函数库admin_fns.php中找到。

除了添加新目录和新书，管理员还可以编辑和删除它们。我们已经重用了尽可能多的脚本来实现这些功能。当管理员点击管理菜单页面中的“Go to Main Site”链接时，将回到index.php中的目录索引，并且与普通用户一样使用同样的脚本接受该索引的导航。

然而，管理导航之间不完全相同：管理员将看到不同的选项，这些选项是根据他们已经注册的会话变量admin_user来确定的。例如，如果我们查看一下本章前面介绍的show_book.php页，将看到一些不同的菜单选项，如图28-14所示。

管理员可以访问该页面中的两个新增选项：“Edit Item”和“Admin Menu”。注意在页面的右上角我们看不到购物车—取而代之的是一个“Log Out”按钮。

这些代码都在程序清单28-8中列出，如下所示：

```
if(check_admin_user()) {
  display_button("edit_book_form.php?isbn=".$isbn, "edit-item", "Edit Item");
```

```
  display_button("admin.php", "admin-menu", "Admin Menu");
  display_button($target, "continue", "Continue");
}
```

如果回头看看show_cat.php脚本，将发现它也有这些内置选项。

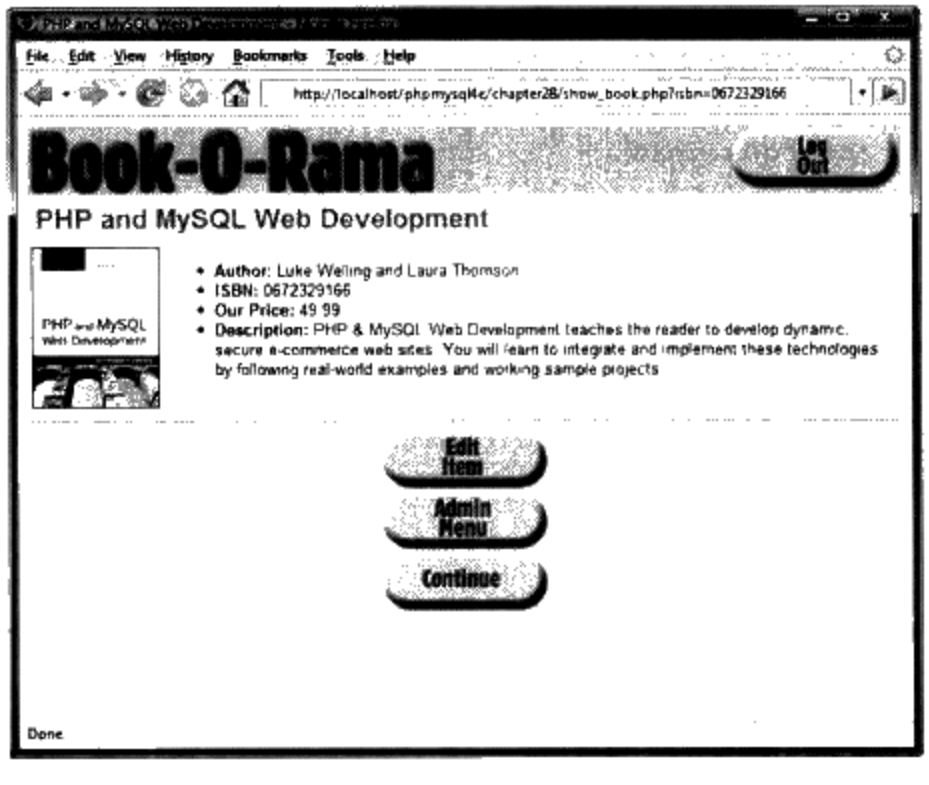

图28-14 show_book.php脚本为管理员用户生成不同的输出

如果管理员点击“Edit Item”按钮，将进入edit_book_form.php脚本。该脚本的输出如图28-15所示。

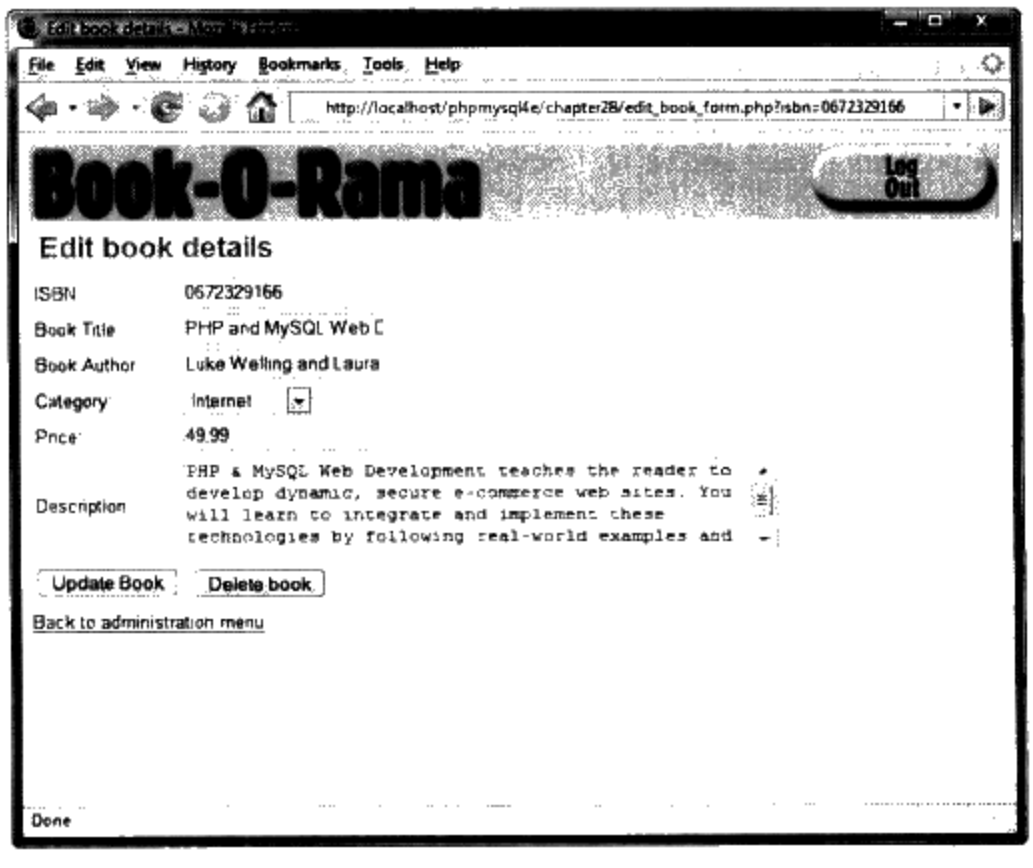

图28-15 edit_book_form.php脚本使管理员可以编辑图书的详细信息或者删除一本书

实际上，这是我们前面用来获取图书详细信息的同一个表单。我们建立了一个指向该表单的选项用来传入和显示现存的图书数据。对于目录，同样如此。要明白为什么目录操作也是如此，请参阅程序清单28-19。

程序清单28-19　admin_fns.php文件中的display_book_form()函数——该表单完成了两项工作：插入图书和编辑图书

```
function display_book_form($book = '') {
// This displays the book form.
// It is very similar to the category form.
// This form can be used for inserting or editing books.
// To insert, don't pass any parameters. This will set $edit
// to false, and the form will go to insert_book.php.
// To update, pass an array containing a book. The
// form will be displayed with the old data and point to update_book.php.
// It will also add a "Delete book" button.


  // if passed an existing book, proceed in "edit mode"
  $edit = is_array($book);
  // most of the form is in plain HTML with some
  // optional PHP bits throughout
?>
<form method="post"
      action="<?php echo $edit ? 'edit_book.php' : 'insert_book.php';?>">
<table border="0">
<tr>
  <td>ISBN:</td>
  <td><input type="text" name="isbn"
       value="<?php echo $edit ? $book['isbn'] : ''; ?>" /></td>
</tr>
<tr>
  <td>Book Title:</td>
  <td><input type="text" name="title"
       value="<?php echo $edit ? $book['title'] : ''; ?>" /></td>
</tr>
<tr>
  <td>Book Author:</td>
  <td><input type="text" name="author"
       value="<?php echo $edit ? $book['author'] : ''; ?>" /></td>
</tr>
<tr>
   <td>Category:</td>
   <td><select name="catid">
   <?php
```

```
        // list of possible categories comes from database
        $cat_array=get_categories();
        foreach ($cat_array as $thiscat) {
             echo "<option value=\"".$thiscat['catid']."\"";
             // if existing book, put in current catgory
             if (($edit) && ($thiscat[ catid'] == $book['catid'])) {
                echo   selected";
             }
             echo ">".$thiscat['catname']."</option>";
        }
        ?>
        </select>
      </td>
</tr>
<tr>
 <td>Price:</td>
 <td><input type="text" name='price"
             value='<?php echo $edit ? $book['price'] : ''; ?>" /></td>
</tr>
<tr>
  <td>Description:</td>
  <td><textarea rows='3' cols="50"
        name='description">
        <?php echo $edit ? $book['description'] : ''; ?>
        </textarea></td>
</tr>
<tr>
  <td <?php if (!$edit) { echo "colspan=2"; }?> align="center">
     <?php
        if ($edit)
        // we need the old isbn to find book in database
        // if the isbn is being updated
        echo '<input type=\"hidden\' name-\"oldisbn\"
               value=\"".$book['isbn']."\" />";
     ?>
  <input type="submit"
         value='<?php echo $edit ? 'Update' : 'Add'; ?> Book' />
  </form></td>
  <?php
     if ($edit) {
       echo "<td>
             <form method=\"post\' action=\"delete_book.php\">
             <input type=\'hidden\" name-\'isbn\"
              value=\"".$book['isbn']."\" />
             <input type=\"submit\" value=\"Delete book\"/>
```

```
                    </form></td>';
                }
            ?>
            </td>
        </tr>
    </table>
    </form>
<?php
}
```

如果我们传入一个包含图书数据的数组，该表单将显示为编辑模式，并用数组中的数据填充其中的域，如下所示：

```
<input type="text" name="price"
            value="<?php echo $edit ? $book[ price'] : ''; ?>" /><
```

我们甚至可以得到一个不同的提交按钮。实际上，对编辑表单，我们有两个脚本——一个用来更新图书，而另一个用来删除图书。这两个脚本分别称为edit_book.php和delete_book.php，它们都要相应地更新数据库。

从工作原理来说，目录版本与图书版本的脚本是相同的，但有一点不同的是，当管理员删除一个目录的时候，如果该目录仍然包含有图书，那么该目录不可删除（这需要通过数据库查询进行检查），这避免了任何不正常删除的问题。在第8章中我们讨论过这样的问题。在这种情况下，如果一个目录还有图书包含在内，就被直接删除，那么这些书将成为“孤儿”。我们不知道它们属于哪个目录，也不知道如何查找它们！

以上就是对管理界面的概述。要了解更多信息，请参阅代码——它们都可在附带的文件里找到。

28.8 扩展该项目

本项目创建了一个相当简单的购物车系统。我们还可以对它进行许多改进和提高：

- 在真正的在线商店，可能必须建立一些订单记录和实施系统——在我们这个系统中，用户无法看到已经预订了的订单。
- 顾客希望在不必与我们联系的前提下就能检查到他们的订单处理情况。我们觉得重要的是用户不必登录进来就能看到这些信息。但是，为已有客户提供一种验证身份的方法可以使他们能够查看自己以前的订单，并且也可以将操作与个人情况紧密地结合起来。
- 如今，图书的图片可以通过FTP传输到该网站的图像目录并给它们取一个合适的名字。可以把文件上载到图书插入页，以使该操作方便一些。
- 可以添加用户登录、个性化设置以及书目推荐、在线评论、会员制度、库存级别检查等。可能要添加的物品是无穷无尽的。

28.9 使用一个已有系统

如果希望得到一个很有特点而且运行速度也很快的购物车，可以尝试使用一个已有的购物

车系统。一个著名的使用PHP编写的开放源代码购物车叫FishCartSQL，可以通过如下站点获得：http://www.fishcart.org/。

该系统有很多高级的特性，例如，用户记录、即时交易、多语言支持、信用卡处理以及支持在一个服务器上部署多个在线商店。当然，当使用一个已有系统的时候，经常会发现有许多功能自己不需要使用，而需要使用的功能可能没有。开放源代码产品的好处就是可以随时进入源代码改掉不喜欢的东西。

28.10 下一章

在下一章中，我们将讨论如何构建一个能够收发电子邮件的Web界面，这个界面将使用IMAP协议收发电子邮件。