第33章　使用XML和SOAP来连接Web服务

在最近几年中，XML（可扩展的置标语言）已经成为通信的重要方法。在本章中，我们使用亚马逊公司最新的Web服务接口在本地Web站点中创建一个以Amazon为后台的购物车（我们将这个应用程序命名为Tahuayo，它是亚马逊河区印第安人的名称）。我们将使用两种不同的方法来创建购物车：SOAP和REST。REST也就是基于HTTP协议之上的XML。我们将使用PHP内置的SimpleXML库和NuSOAP库来实现这两种方法。

在本章中，我们将主要介绍以下内容：

- 理解XML和Web服务的基础知识
- 使用XML与Amazon进行交互
- 使用PHP的SimpleXML库解析XML
- 缓存响应
- 使用NuSOAP与Amazon进行交互

33.1　项目概述：使用XML和Web服务

在这个项目中，我们要实现两个目标：第一，理解什么是XML和SOAP并且掌握如何在PHP中使用它们。第二，使用这两种技术与外界进行通信。我们之所以选择Amazon的Web服务程序作为例子，是因为它对我们自己的Web站点将会非常有帮助。

很早以前，Amazon就提供了允许在Web站点给其产品做广告的相关程序。用户可以点击这些链接进入到Amazon站点上每一个产品的页面。如果某些用户通过某站点访问Amazon站点并且成功购买其产品，该站点将获得一定的现金奖励。

Amazon的这些Web服务程序允许将它作为一个引擎：我们可以搜索Amazon站点并且通过自己的站点显示搜索结果，或者在用户浏览我们站点时将用户选择的产品直接放入购物车中。换句话说，客户在付费之前可以一直使用我们的站点，就像在Amazon站点上的购买操作一样。

我们和Amazon站点之间的交互可以按两种可能的方式进行。第一种方法是使用基于HTTP的XML，也就是表示状态转换（REST）。例如，如果我们希望使用这种方法执行一个搜索操作，可以向Amazon发送一个关于所查询信息的普通HTTP请求，Amazon将用包含请求信息的XML文档响应查询。于是，我们可以使用PHP的XML库对这个XML文档进行解析并且使用我们所选择的接口向终端用户显示搜索结果。通过HTTP发送和接收数据的过程非常简单，但是解析结果文档的难易程度则是由文档的复杂度决定的。

第二种方法是使用SOAP。SOAP是Web服务的标准协议之一。它是简单对象访问协议的缩写，但是最近该协议变得不再简单，因此其名称就有一点名不符实。最后的结果是该协议仍然叫做SOAP，但是它不再是一个缩写词了。

在这个项目中，我们将创建一个可以向Amazon SOAP服务器发送请求并获得响应的SOAP

客户端。这些将包含我们使用基于HTTP的XML方法从Amazon服务器获得的响应信息相同的信息，但是我们将使用不同的方法来获取数据，也就是NuSOAP库。

在这个项目中，我们最终的目标是创建自己的、使用Amazon作为后台的图书销售Web站点。我们将创建两个版本：一个是使用REST，而另一个是使用SOAP。

在真正进入这个项目之前，我们需要熟悉XML和Web服务的常规结构和使用。

33.1.1 理解XML

下面，如果我们不熟悉XML和Web服务的概念，那么花一些时间来了解它们。

正如前面已经提到的，XML是可扩展的置标语言。在W3C的站点上，提供了其规范。在W3C的XML站点上，可以找到大量关于XML的信息。W3C的XML站点URL如下所示：http://www.w3.org/XML/。

XML源自SGML（标准通用置标语言）。如果我们已经了解了HTML（假设我们的确已经了解了），这对快速理解XML概念是非常有帮助的。

XML是针对文档的，基于标记的文本格式。作为XML文档的一个例子，程序清单33-1给出了Amazon站点返回的、针对一个基于特定请求参数的HTTP XML请求的响应内容。

程序清单33-1　描述了本书第一版的XML文档

```
<?xml version="1.0" encoding="UTF-8"?>
<ItemLookupResponse
   xmlns="http://webservices.amazon.com/AWSECommerceService/2005-03-23">
  <Items>
    <Request>
      <IsValid>True</IsValid>
        <ItemLookupRequest>
        <IdType>ASIN</IdType>
        <ItemId>0672317842</ItemId>
        <ResponseGroup>Similarities</ResponseGroup>
        <ResponseGroup>Small</ResponseGroup>
      </ItemLookupRequest>
    </Request>
   <Item>
      <ASIN>0672317842</ASIN>
      <DetailPageURL>http://www.amazon.com/PHP-MySQL-Development-Luke-Welling/
dp/0672317842%3Fp%26linkCode%3Dsp1%26camp%3D2025%26creative%3D165953%26crea
tiveASIN%3D0672317842
      </DetailPageURL>
      <ItemAttributes>
        <Author>Luke Welling</Author>
        <Author>Laura Thomson</Author>
        <Manufacturer>Sams</Manufacturer>
        <ProductGroup>Book</ProductGroup>
        <Title>PHP and MySQL Web Development</Title>
      </ItemAttributes>
```

```
<SimilarProducts>
   <SimilarProduct>
      <ASIN>1590598628</ASIN>
      <Title>Beginning PHP and MySQL: From Novice to Professional,
      Third Edition (Beginning from Novice to Professional)</Title>
   </SimilarProduct>
   <SimilarProduct>
      <ASIN>032152599X</ASIN>
      <Title>PHP 6 and MySQL 5 for Dynamic Web Sites:
          Visual QuickPro Guide</Title>
   </SimilarProduct>
   <SimilarProduct>
      <ASIN>B00005UL4F</ASIN>
      <Title>JavaScript Definitive Guide</Title>
   </SimilarProduct>
   <SimilarProduct>
      <ASIN>1590596145</ASIN>
      <Title>CSS Mastery: Advanced Web Standards Solutions</Title>
   </SimilarProduct>
   <SimilarProduct>
      <ASIN>0596005431</ASIN>
      <Title>Web Database Applications with PHP &amp;amp; MySQL,
         2nd Edition</Title>
   </SimilarProduct>
      </SimilarProducts>
   </Item>
</Items>
```

该文档是以如下语句行为开始的：

```
<?xml version= "1.0" encoding= "UTF-8"?>
```

这是一个标准的声明，它告诉我们接下来的文档将会是使用UTF-8字符编码的XML。

让我们看看该文档的正文。整个文档由打开和关闭的标记对组成，例如，你可以从以下代码的开放和关闭的Item标记看到：

```
<Item>
...
```

</Item>是一个元素，就像其在HTML中一样。而且，就像在HTML中，我们可以嵌套元素，例如，以上代码的ItemAttributes元素，它为Item元素内，同样，Author元素却位于ItemAttributes元素内：

```
<ItemAttributes>
   <Author>Luke Welling</Author>
   <Author>Laura Thomson</Author>
   <Manufacturer>Sams</Manufacturer>
   <ProductGroup>Book</ProductGroup>
   <Title>PHP and MySQL Web Development</Title>
```

还像HTML中的一样，元素可以具有属性。就像这个例子中，例如：

Details元素只有一个属性url。由于URL非常长，在这里，它被分成了3行。

当然，XML和HTML还是有一些不同的。首先，在XML中，所有开始的标记都必须有一个对应的结束标记。这条规则的例外就是空元素，空元素的开始和结束标记都在一个标记内，因为它没有包括任何文本。如果我们熟悉XHTML，将会看到，我们可以在使用
的地方使用
标记。此外，所有元素必须正确地嵌套。在HTML解析器中，Text可能是合法的，但是在XML或XHTML中，它却是不合法的，这些标记必须正确地嵌套，Text。

注意XML和HTML之间最主要的区别在于我们可以在文档中设置自己的标记！这就是XML的灵活性。我们可以根据希望保存的数据来设计文档的结构。我们可以通过编写一个DTD（文档类型定义）或XML模式来规范化XML文档的结构。这两个文档都是用来描述一个给定的XML文档的结构。如果你愿意，我们可以将DTD或模式看作是一个类声明，而XML文档则是类的实例。在这个例子中，没有使用DTD或模式。

可以通过如下URL给出的Web服务查阅Amazon站点的XML模式：http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.xsd。

你应该可以直接在浏览器中打开这个XML模式。

需要注意的是，除了最初的XML声明外，文档的所有正文都是包含在ItemLookupResponse元素中。这个元素就叫做该文档的根元素。下面，我们仔细查看这个元素：

```
<ItemLookupResponse
xmlns="http://webservices.amazon.com/AWSECommerceService/2005-03-23">
```

该元素具有一些特殊的属性。这些属性是XML的名称空间（namespace）。我们并不需要理解名称空间，因为在这个项目中，这并不重要，但是这将会是非常有用的。该元素的基本功能是确保元素和属性名称的正确性，这样当处理来自不同组织的文档时，这些通用名称不会相互冲突。

如果想要了解更多关于名称空间的内容，可以在如下URL找到“XML标准建议的名称空间”文档：http://www.w3.org/TR/REC-xml-names/。

如果想要了解更多关于XML的内容，在Internet上存在了大量的资料。W3C站点就是了解XML的一个非常不错的入门站点，在该站点上，介绍了关于XML和Web入门的海量图书。ZVON.org提供了关于XML的最佳入门教程。

33.1.2　理解Web服务

Web服务是通过Internet可供使用的应用程序接口。我们可以将Web服务看作是一个通过Web开放了其公有方法的类。现在，Web服务随处可见，而且商业领域的一些著名企业也正在通过Web服务逐步开放它们的功能。

例如，Google、Amazon、eBay和PayPal现在就可以提供一系列的Web服务。在学习了本章关于如何创建一个能够访问Amazon接口的客户端后，我们会发现创建Google的客户端也是非常简单而又直观的。可以在http://code.google.com/apis/找到更多信息。

在如下站点给出了不断发展的公用Web服务列表：http://www.xmethods.net。

在这种远程函数调用方法中，还涉及了一些核心协议。最重要的两个协议是SOAP和WSDL。

1. SOAP

SOAP是一个以请求和响应为驱动的消息传递协议，它允许客户端调用Web服务，服务器对客户端的调用进行响应。每一个SOAP消息，无论是请求的还是响应的，都是一个简单的XML文档。程序清单33-2给出了一个我们可能发送给Amazon的SOAP示例请求。事实上，这个请求产生的XML响应如程序清单33-1所示。

程序清单33-2　一个基于ASIN的SOAP搜索请求

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <m:ItemLookup>
      <m:Request>
        <m:AssociateTag>webservices-20</m:AssociateTag>
        <m:IdType>ASIN</m:IdType>
        <m:ItemId>0672317842</m:ItemId>
        <m:AWSAccessKeyId>0XKKZBBJHE7GNBWF2ZG2</m:AWSAccessKeyId>
        <m:ResponseGroup>Similarities</m:ResponseGroup>
        <m:ResponseGroup>Small</m:ResponseGroup>
      </m:Request>
    </m:ItemLookup>
  </SOAP-ENV:Body>
```

SOAP消息以一个XML文档的声明为开始。所有SOAP消息的根元素是SOAP“信封”。在这个“信封”中，我们可以找到包含真正请求的Body元素。

这个请求是一个ItemLookup，在这个例子中，它请求Amazon服务器在其数据库中基于ASIN（表示Amazon.com 标准条目号）搜索特定项。这个号码是Amazon数据库中每一个产品的唯一标识符。

我们可以将ItemLookup看作是对一个远程计算机的函数调用，而且包含在该元素中的所有元素和属性就是我们传递给这个函数的参数。在这个例子中，通过IdType元素，我们传递了“ASIN”参数，而真正的ASIN值（0672317842）通过ItemId元素进行了传递。这个ASIN是本书第一版的唯一标识符。我们还需要传递另一个参数，那就是AssociateTag,，这是你的Amazon会员ID；此外，还需要的参数有：希望的响应类型（通过ResponseGroup元素）；以及AWSAccessKeyId——Amazon分配的开发人员令牌。

这个请求的响应非常类似于程序清单33-1中的XML文档，但是它是封闭在一个SOAP信封中的。

当使用SOAP时，无论使用何种编程语言，通常要生成SOAP请求并且使用SOAP库通过程序来解释响应。这样做是非常不错的，因为它可以节省大量手动构建SOAP请求并解释响应所需的操作。

2. WSDL

WSDL是Web服务描述语言的缩写（通常，这个词的发音是“wiz-dul”）。这个协议是用来描述特定Web站点上可供使用的接口的。如果想了解用来描述本章中使用的Amazon站点所提供Web服务的WSDL文档，可以访问http://soap.amazon.com/schemas2/AmazonWebServices.wsdl。

如果点击这个链接，将发现WSDL文档明显比SOAP消息要复杂。如果让我们选择的话，肯定会选择使用程序来生成请求并解释响应。

如果希望了解更多关于WSDL的内容，可以访问如下URL：http://www.w3.org/TR/wsdl20/。

33.2 解决方案的组成

实现这个解决方案需要不同的组件。最明显的部分是一个能够向顾客显示的购物车接口和通过REST或SOAP连接到Amazon的代码，除此之外，还需要一些辅助的部分。在接收一个XML文档后，代码必须解析它，并且提取购物车将要显示的信息。要满足Amazon的要求并且提高性能，必须考虑缓存机制。最后，由于付费结账操作必须在Amazon完成，需要一些能够向Amazon提交用户购物车内容以及将用户提交给该服务的功能。

很明显，我们需要创建一个购物车作为系统的前台。在第28章中已经介绍了如何创建一个购物车。由于购物车并不是该项目的主要部分，我们将使用一个简化的应用程序。我们只需要提供一个基本的购物车，这样就可以记录客户想要购买哪些产品，并且在用户结账时将这些产品报告给Amazon。

33.2.1 使用Amazon的Web服务接口

要使用Amazon（亚马逊）的Web服务接口，我们需要注册一个开发人员令牌。可以在http://aws.amazon.com站点完成开发人员的注册。当请求到达Amazon站点时，这个令牌可以用在站点上标识开发人员本身。

我们可能还会希望注册一个Amazon会员ID。当任何客户通过我们的接口在Amazon购买产品时，通过这个ID，Amazon会给我们返回一些现金奖励。

http://developer.amazonwebservices.com/提供的Amazon Web服务（AWS）开发人员资源中心包含了大量关于使用SOAP和REST连接Amazon所有Web服务的文档、教程以及示例代码。本章给出的示例程序将生成一个可用的系统，并且提供连接AWS并获取信息的基本知识，但是如果你打算构建一个类似本章的应用，你就应该花些时间阅读这些文档。例如，你可能希望通过浏览和直接搜索接口搜索并获得不同商品的信息。根据所需元素的不同，所返回的数据也有不同的结构。AWS开发人员指南给出了所有这些信息。

> **提示**　另一个有价值的资源是AWSZone.com（http://www.awszone.com）。在这个网站上，你可以测试SOAP和REST查询，并且了解请求以及响应的结构，这样你就可以知道如何处理返回的数据。此外，测试响应可以有助于确定确切的ResponseGroup对象，从而获得最佳的速度。

在注册开发人员令牌时，你必须同意这个许可协议。这个协议值得一读，因为它不是普通

的软件协议。在实现时，许可的某些条件还是非常重要的，这些条件如下所示：

- 作为客户端，每秒钟不能发出多个请求。
- 必须缓存来自Amazon的数据。
- 可以24小时缓存大多数数据，而一些稳定属性可以缓存至3个月。
- 如果缓存价格或库存信息超过1小时，必须提供一些声明。
- 必须将本地所拥有的所有Amazon数据链接到Amazon站点的页面，严禁将从Amazon下载的文本或图形链接至其他商业网站。

由于没有一个拼写简单的域名，没有推广政策以及其他使用Tahuayo.com的明显理由，所以我们还是直接使用Amazon网站，我们不需要采取任何特殊的措施来保障每秒不超过一个的请求。

在这个项目中，我们已经实现了缓存技术来保障协议条件的第2至4点。我们可以将图像缓存24小时，而产品数据（包含了价格信息和库存信息）则缓存1小时。

我们的应用程序也遵循该协议的第5点。我们希望主页上的产品链接到我们站点的详细信息页面，而且只有当完成交易时才链接到Amazon站点。

33.2.2　XML的解析：REST响应

Amazon为其Web服务提供的第一个最受欢迎的接口是通过REST实现的。这个接口可以接受一个普通的HTTP请求并且返回一个XML文档。要使用这个接口，需要解析Amazon返回给我们的XML响应。可以通过PHP的SimpleXML库来实现XML的解析。

33.2.3　在PHP中使用SOAP

另一种可以提供相同Web服务的接口是SOAP。要使用SOAP访问这些服务，我们需要使用许多不同的PHP SOAP库中的其中之一。PHP内置有SOAP库，但是由于该库并不是一直可用的，因此可以使用NuSOAP库。由于NuSOAP是用PHP编写的，它并不需要进行编译。它只是一个可以通过`require_once()`进行调用的文件。

在http://dietrich.ganx4.com/nusoap上，可以找到NuSOAP。NuSOAP可以在Lesser的开放源代码项目中找到，也就是说，我们可以在任何应用程序中使用它，包括非免费的应用程序。

33.2.4　缓存

正如我们前面提到的，亚马逊对开发人员提出的条件和约束之一就是必须缓存通过Web服务下载的数据。在我们的解决方案中，仍然需要找到一个保存和重复使用所下载数据的方法，当然必须在这些数据失效之前。

33.3　解决方案概述

对于这个项目，我们还将使用事件驱动的方法来编写代码，正如第29章和第30章所介绍的。在这个例子中，我们不会绘制系统流程图，因为系统中只出现非常少的几个界面，而且这些页面之间的链接是非常简单的。

如图33-1所示的是用户访问Tahuayo时看到的主页面。

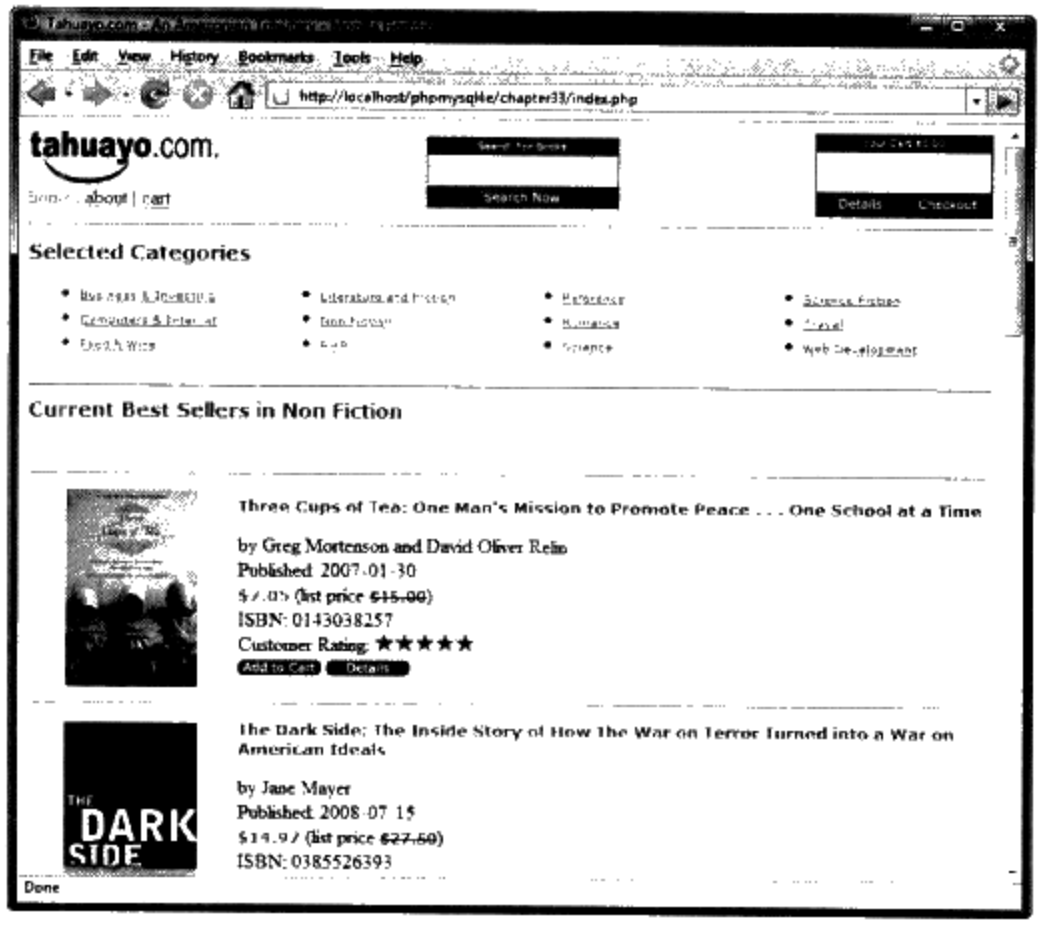

图33-1　Tahuayo的第一个页面显示了该站点的所有主要特性：目录浏览、搜索和购物车

可以看到，该站点的主要特性就是选中目录的显示以及显示这些目录中的物品。在默认情况下，第一页上显示当前销售情况最好的产品目录。如果一个用户点击其他目录，将看到该目录下相似的显示。

在我们进一步开始项目之前，需要掌握一些简单的术语：Amazon将目录当作是浏览节点。在贯穿整个代码和正式的文档中，我们都将发现这种表达方式。

文档提供了所有流行的浏览节点列表。此外，如果希望看到特定的节点，可以浏览Amazon站点并且从URL中读入它们，你可以在http://www.browsernodes.com/获得Browse Nodes资源。

奇怪的是，某些重要的目录，例如销售最佳的图书，无法通过浏览节点进行访问。

在这个页面的下方，还有许多图书和链接，但是无法从上图中看到（上图只是屏幕的截图）。我们在每一页上显示10本图书，同时显示链接到30多本相关图书。这种每页显示10本图书的设置是由Amazon确定的。而每页显示30本图书的设置是我们自己的选择。

从这个页面，用户可以点击并查看每一本图书的详细信息。这个页面如图33-2所示。

虽然无法在一个截图中显示整个页面，但是我们已经尽量显示了更多的内容，但是并不是全部的，因为在这个页面上显示的信息是一个heavy查询得来的。我们选择过滤了图书以外的产品，以及不适合图书目录的其他产品列表。

如果点击图书的封面，可以看到一个扩大后的图书封面。

我们可能已经注意到了位于上图右上方的搜索文本框。这个搜索操作可以对站点的关键字

进行搜索，同样，它也可以通过Web服务接口搜索Amazon的目录。一个搜索操作的输出结果示例如图33-3所示。

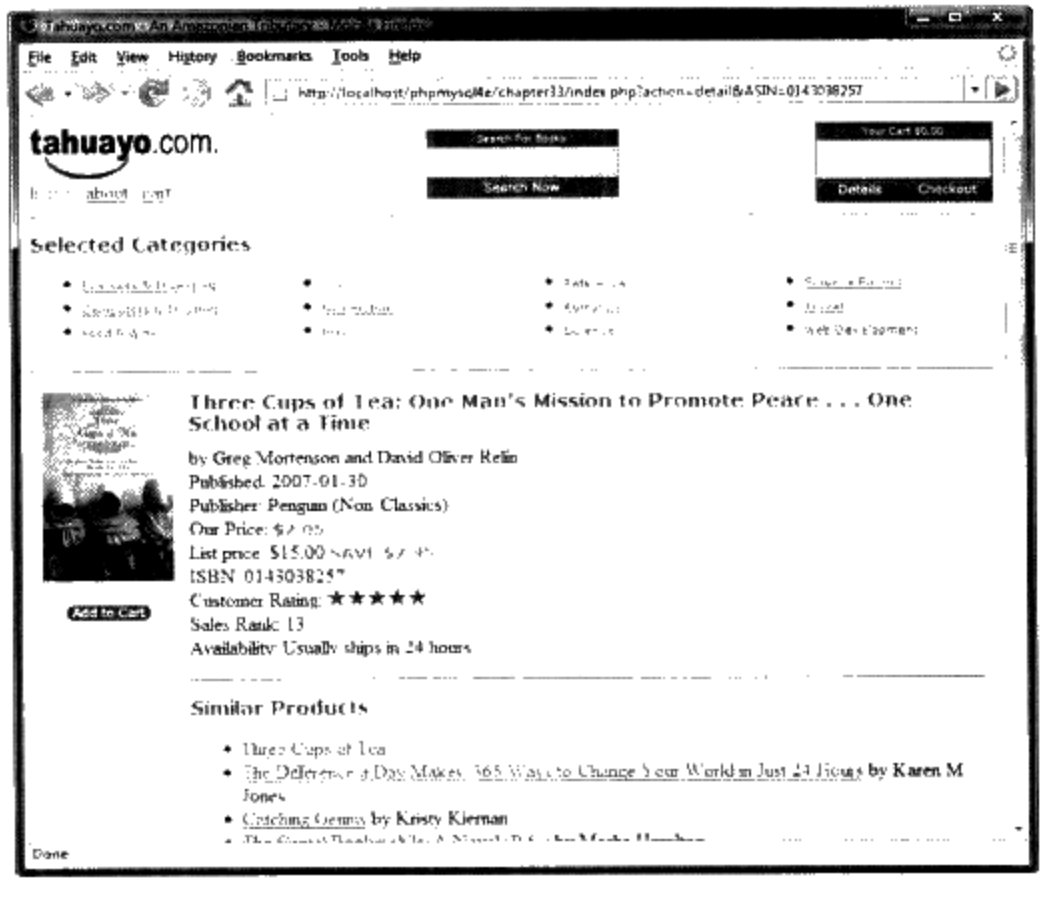

图33-2　详细信息页面显示了关于特定图书的详细信息，同时还包括类似产品和概述

虽然我们只是列出了少量的目录，但是客户可以通过这个搜索工具找到任何想要的图书，并且浏览特定的图书。

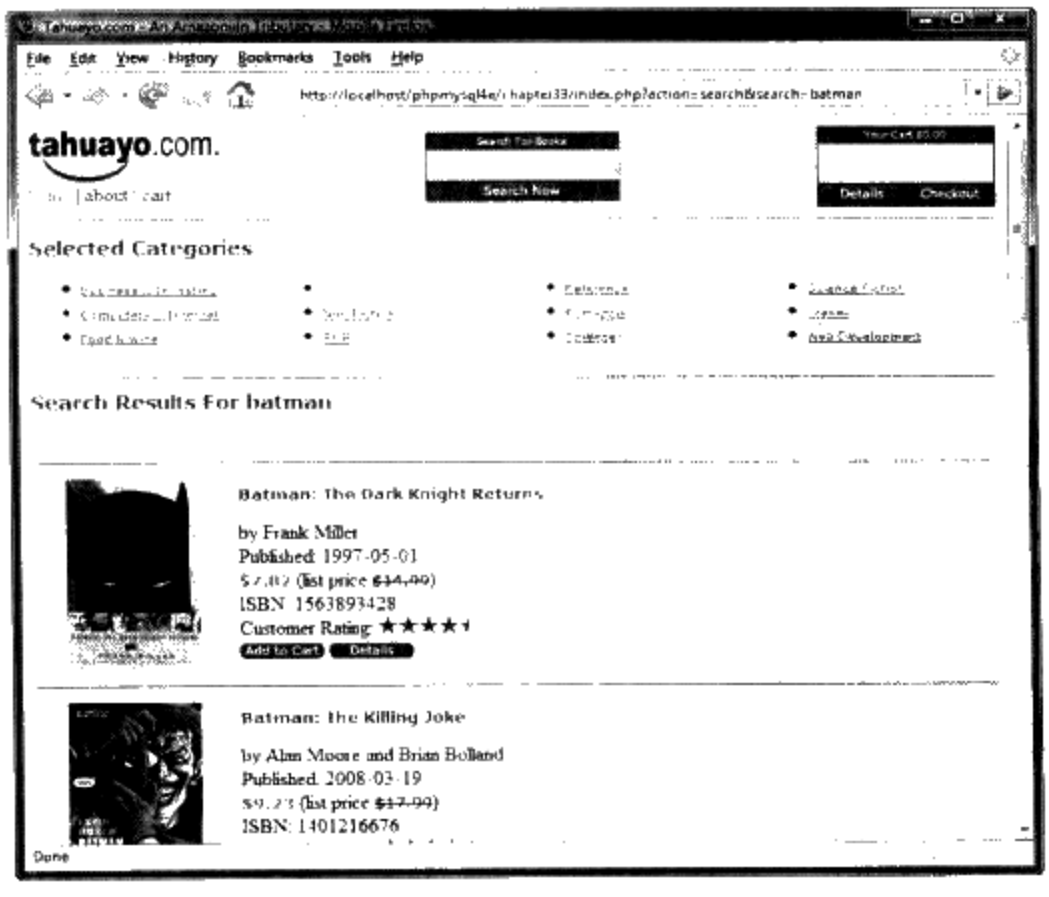

图33-3　搜索batman的结果输出

每一本图书都有一个“添加到购物车（Add to Cart）”的链接。点击购物车汇总中的这个链接或“详细信息（Details）”链接，将显示购物车中的详细内容，如图33-4所示。

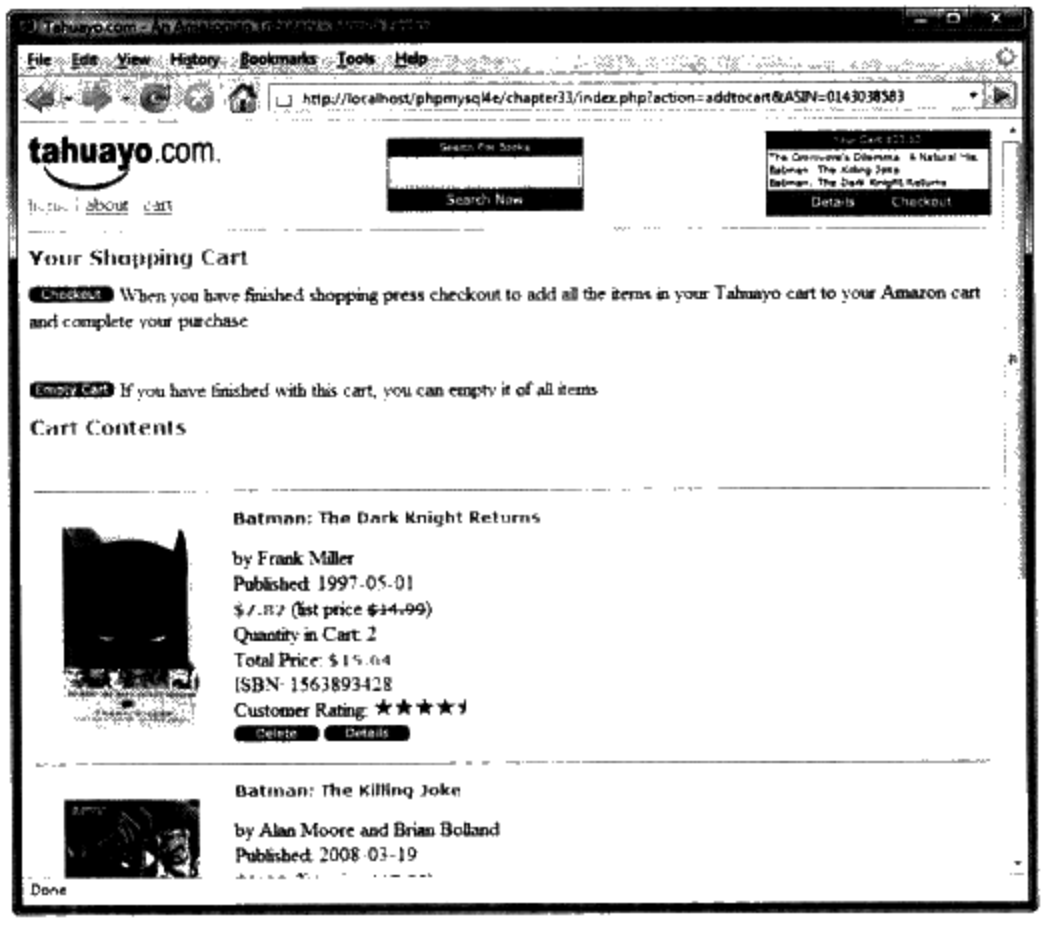

图33-4　在购物车页面，可以删除物品，清空购物车或付账

最后，当用户点击其中任何一个“付账（Checkout）”链接，都会将购物车中的详细信息发送给Amazon并且转到Amazon站点。这样，客户就将看到如图33-5所示的页面。

通过构建我们自己的前台应用程序和使用Amazon作为后台，可以理解本项目的含义。

由于这个项目仍然使用了事件驱动的方法，所以这个应用程序中的核心程序逻辑都是在一个文件中实现的——index.php。在该应用程序中，所用到的文件概述如表33-1所示。

表33-1　Tahuayo应用程序中用到的文件

文件名称	类　型	描　述
index.php	应用程序	主要应用程序文件
about.php	应用程序	显示“关于”页面
constants.php	包含文件	设置一些全局变量
topbar.php	包含文件	生成每一个页面和CSS之间的信息栏
bottom.php	包含文件	生成每一个页面的注脚
AmazonResultSet.php	类文件	包含保存每一个Amazon查询结果的PHP类
Product.php	类文件	包含保存特定图书信息的PHP类
bookdisplayfunctions.php	函数	包含用来显示图书和图书列表的函数
cachefunctions.php	函数	包含执行Amazon所需的缓存操作相关的函数
cartfunctions.php	函数	包含与购物车相关的函数
categoryfunctions.php	函数	包含用来检索和显示一个目录的函数
utilityfunctions.php	函数	包含在整个应用程序中要使用到的工具函数的集合

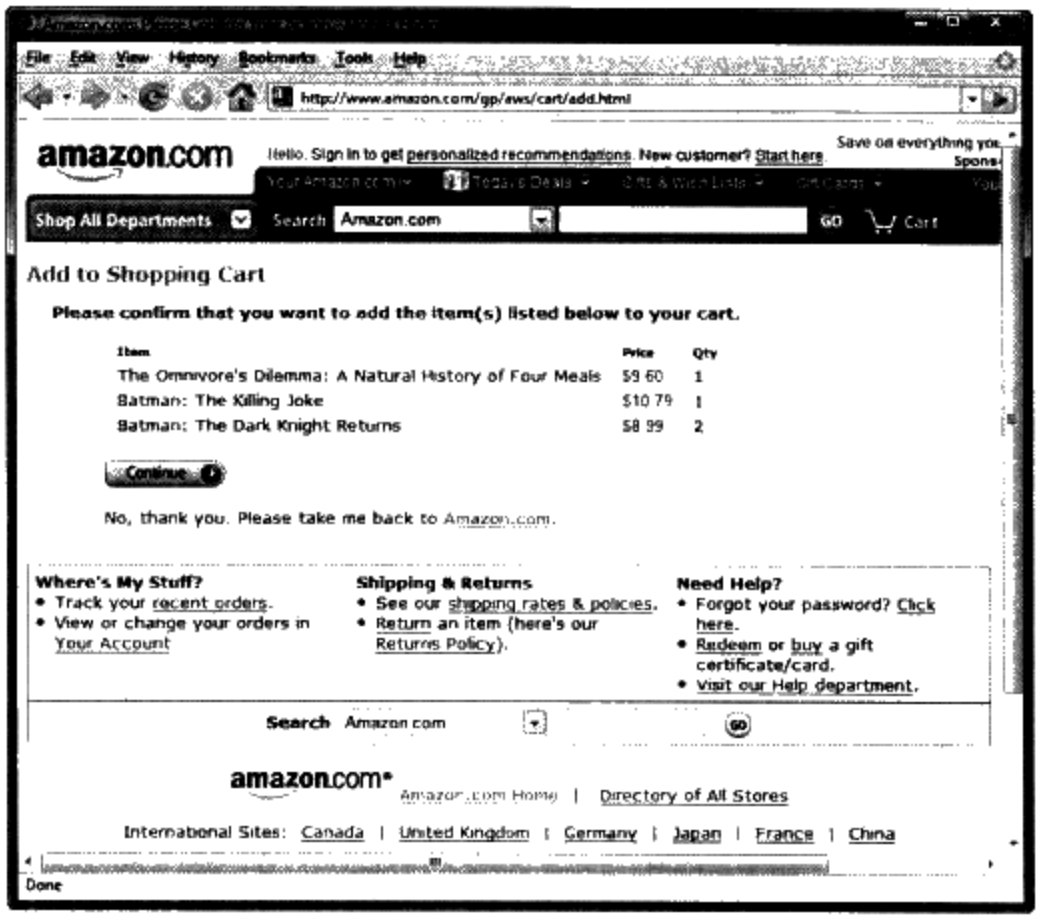

图33-5　在将客户选购物品保存在Amazon购物车之前，系统将确认该交易，并显示Tahuayo购物车中的所有物品

我们还需要前面所介绍的nusoap.php文件，因为在以上文件中，这个文件是必需的。NuSOAP文件可以在附带的文件中找到，具体目录位于chapter33中，但是也可以从http://dietrich.ganx4.com/nusoap/index.php中找到该文件的最新版本（如果发布了新版本）。

下面，我们开始了解核心应用程序index.php文件。

33.3.1　核心应用程序

程序清单33-3所示的就是核心应用程序index.php文件。

程序清单33-3　index.php——核心应用程序文件

```
<?php
//we are only using one session variable 'cart' to store the cart contents
session_start();

require_once('constants.php');
require_once('Product.php');
require_once('AmazonResultSet.php');
require_once('utilityfunctions.php');
require_once('bookdisplayfunctions.php');
require_once('cartfunctions.php');
require_once('categoryfunctions.php');

// These are the variables we are expecting from outside.
```

```
// They will be validated and converted to globals
$external = array('action', 'ASIN', 'mode', 'browseNode', 'page', 'search');

// the variables may come via Get or Post
// convert all our expected external variables to short global names
foreach ($external as $e) {
  if(@$_REQUEST[$e]) {
    $$e = $_REQUEST[$e];
  } else {
    $$e = '';
  }

  $$e = trim($$e);
}

// default values for global variables
if($mode=='') {
  $mode = 'Books'; // No other modes have been tested
}
if($browseNode=='') {
  $browseNode = 53; //53 is bestselling non-fiction books
}
if($page=='') {
  $page = 1; // First Page - there are 10 items per page
}

//validate/strip input
if(!eregi('^[A-Z0-9]+$', $ASIN)) {
  // ASINS must be alpha-numeric
  $ASIN ='';
}
if(!eregi('^[a-z]+$', $mode)) {
  // mode must be alphabetic
  $mode = 'Books';
}
$page=intval($page); // pages and browseNodes must be integers
$browseNode = intval($browseNode);
// it may cause some confusion, but we are stripping characters out from
// $search it seems only fair to modify it now so it will be displayed
// in the heading
$search = safeString($search);

if(!isset($_SESSION['cart'])) {
  session_register('cart');
  $_SESSION['cart'] = array();
}

// tasks that need to be done before the top bar is shown
```

```
if($action == 'addtocart') {
  addToCart($_SESSION['cart'], $ASIN, $mode);
}
if($action == 'deletefromcart') {
  deleteFromCart($_SESSION['cart'], $ASIN);
}
if($action == 'emptycart') {
  $_SESSION['cart'] = array();
}

// show top bar
require_once ('topbar.php');

// main event loop. Reacts to user action on the calling page
switch ($action) {
  case 'detail' :
    showCategories($mode);
    showDetail($ASIN, $mode);
  break;

  case 'addtocart':
  case 'deletefromcart':
  case 'emptycart':
  case 'showcart' :
    echo '<h1 /><h1>Your Shopping Cart</h1>';
    showCart($_SESSION['cart'], $mode);
break;

case 'image' :
  showCategories($mode);
  echo "<h1>Large Product Image</h1>" ;
  showImage($ASIN, $mode);
break;

case 'search':
  showCategories($mode);
  echo "<h1>Search Results For ".$search."</h1>";
  showSearch($search, $page, $mode);
break;

case 'browsenode' :
default:

  showCategories($mode);
  $category = getCategoryName($browseNode);
  if(!$category || ($category=='Best Selling Books')) {
    echo '<h1>Current Best Sellers</h1>';
```

```
    } else {
      echo '<h1>Current Best Sellers in  .$category.'</h1>";
    }
    showBrowseNode($browseNode, $page, $mode) ;
  break;
}
require ( bottom.php );
```

下面，我们用自己的方法来了解这个文件。首先，我们创建了一个会话。就像前面所介绍的，将客户的购物车保存为一个会话变量。

接着，我们包括并引入了几个文件。这些文件都是下面将要介绍的函数，但是在介绍它们之前，我们必须先介绍第一个被包括进来的文件。constants.php文件定义了一些重要的常量，这些常量将在整个应用程序中使用。程序清单33-4给出了constants.php的所有代码。

程序清单33-4 constants.php——声明重要的全局常量和变量

```
<?php
// this application can connect via REST (XML over HTTP) or SOAP
// define one version of METHOD to choose.
// define('METHOD',  SOAP );
define('METHOD', 'REST');

// make sure to create a cache directory an make it writable
define( CACHE',  cache'); // path to cached files
define('ASSOCIATEID', 'XXXXXXXXXXXXXX'); //put your associate id here
define('DEVTAG ,  XXXXXXXXXXXXXX'); // put your developer tag here

//give an error if software is run with the dummy devtag
if(DEVTAG--'XXXXXXXXXXXXXX') {
  die ("You need to sign up for an Amazon.com developer tag at
       <a href=\"https://aws.amazon.com/ >Amazon</a>
       when you install this software. You should probably sign up
       for an associate ID at the same time. Edit the file constants.php.');
}

// (partial) list of Amazon browseNodes.
$categoryList = array(5-> Computers & Internet , 3510->'Web Development',
                      295223->'PHP', 17->'Literature and Fiction',
                      3->'Business & Investing , 53=>'Non Fiction',
                      23=>'Romance , 75 = Science , 21=>'Reference',
                      6 ->'Food & Wine', 27 > Travel',
                      16272=>'Science Fiction
                     );
```

这个应用程序可以使用REST或SOAP进行开发。修改METHOD常量值，可以选择所使用的开发方法。

CACHE常量定义了保存我们从Amazon站点上下载的数据的路径。可以修改该常量值，使

其指向系统上任何希望的地方。

ASSOCIATEID常量定义了会员ID。如果是在事务中发送会员ID，将得到现金奖励。修改该常量，使其保存会员ID。

DEVTAG常量定义了Amazon在注册时为我们分配的开发人员令牌。必须将其修改为我们自己的开发人员令牌，否则该应用程序将无法正常运行。可以在如下URL注册一个开发人员令牌：http://aws.amazon.com。

现在，让我们回头看看index.php文件。它包含一些初始设置，以及主要的事件循环。首先，我们将从通过GET或POST方法传递进来的$_REQUEST超级全局变量中获取所需的变量。接着，将为这些标准的全局变量设置默认值，这些全局变量可以确定以后的页面显示，如下所示：

```
// default values for global variables
if($mode=='') {
  $mode = 'Books'; // No other modes have been tested
}
if($browseNode=='') {
  $browseNode = 53; //53 is bestselling non-fiction books
}
if($page=='') {
  $page = 1; // First Page - there are 10 items per page
```

在以上代码中，我们将mode变量设置为“books”。Amazon支持许多模式（产品的类型），但是对于这个应用程序，我们只考虑图书。修改本章代码使其适用于其他产品目录不会太困难，只要重新设置$mode变量就可以了。此外，可能还需要查看Amazon文档，确认对于非图书类产品是否还具有其他属性，同时从用户界面上去除只与图书相关的文字。

browseNode变量用来指定要显示的图书种类。如果用户点击了“Selected Categories”链接，就可以自动设置图书种类。如果还没有设置（例如，当用户第一次来到该站点）我们仍然需要对其进行设置，设置为53。Amazon的浏览节点都是整数，用整数来标识一个种类。53表示非科幻图书类别，该类别也是一个非常不错的节点，就像那些出现在初始页面的类别，虽然有些最好的常见目录（例如，最佳销售）无法通过浏览节点进行访问。

page变量用来告诉Amazon我们希望在给定的种类中要显示的结果子集。page 1包含了1～10的结果，page 2包含了11～20的结果等。Amazon可以设置一个页面上的结果数，我们不用对其进行控制。当然，也可以在一页上显示Amazon站点上两页或更多页上的数据，但是对Amazon和我们的站点来说，10是都可以接受的，而且这样可以使我们的站点与Amazon保持一致。

接下来，我们将整理接收到的输入数据，可以通过搜索文本框或者是GET和POST参数：

```
//validate/strip input
if(!eregi('^[A-Z0-9]+$', $ASIN)) {
  // ASINS must be alpha-numeric
  $ASIN ='';
}
if(!eregi('^[a-z]+$', $mode)) {
  // mode must be alphabetic
  $mode = 'Books';
```

```
}
$page=intval($page); // pages and browseNodes must be integers
$browseNode = intval($browseNode);
// it may cause some confusion, but we are stripping characters out from
// $search it seems only fair to modify it now so it will be displayed
// in the heading
```

以上代码没有什么新内容。safeString()函数源自utilityfunctions.php函数库。它只是通过一个正则表达式的替换操作，从输入字符串中删除任何非字母字符。由于我们在前面已经介绍它，这里不再进行描述。

在应用程序中，我们对用户输入数据进行校验的主要原因在于，我们将使用客户的输入在缓存中创建文件名称。如果允许客户在输入中使用“..”或“/”，可能会遇到非常严重的问题。

接下来，如果客户还没有购物车的话，将为其设置一个购物车：

```
if(!isset($_SESSION['cart'])) {
  session_register('cart');
  $_SESSION['cart'] = array();
```

在页面最上方的信息栏中显示信息之前（参阅图33-1），还需要完成一些操作。购物车将出现在每一个页面最上方的信息栏中。因此，在显示购物车内容之前，保持购物车变量为最新的是非常重要的：

```
// tasks that need to be done before the top bar is shown
if($action == 'addtocart') {
  addToCart($_SESSION['cart'], $ASIN, $mode);
}
if($action == 'deletefromcart') {
  deleteFromCart($_SESSION['cart'], $ASIN);
}
if($action ==  emptycart') {
  $_SESSION[ cart'] = array();
```

在这里，我们将在显示购物车之前添加或删除一些物品。讨论购物车和付账时，我们还将介绍这些函数。如果想现在就了解这些函数，可以在cartfunctions.php文件中找到它们。现在，我们先将它们放到一边，因为首先必须理解Amazon的接口。

下一步，我们将包含并引入topbar.php文件。这个文件只包含了HTML和样式单，以及一个对ShowSmallCart()函数的调用（源自cartfunctions.php）。这个函数将显示购物车的总结信息，我们将在每个页面右上方看到它们。介绍购物车函数时，我们还将介绍这些函数。

最后，我们介绍事件处理的主循环。表33-2给出了可能出现的事件总结。

可以看到，表33-2中的前4个事件都与获取和显示信息相关。而后4个事件都与管理购物车相关。

从Amazon获取数据的事件具有相似的工作方式。我们将以browsenode（种类）事件为例介绍如何获取关于图书的数据。

表33-2　可能出现的事件主循环

事　件	描　述
browsenode	显示特定种类中的图书。这是默认的事件
detail	显示特定图书的详细信息
image	显示图书封面的图像
search	显示用户搜索结果
addtocart	将一个物品添加到用户的购物车中
deletefromcart	从购物车中删除一个物品
emptycart	清空购物车
showcart	显示购物车中的物品

33.3.2　显示特定种类的图书

当browsenode（查看一个种类）事件被执行时，将运行如下代码：

```
showCategories($mode);
$category = getCategoryName($browseNode);
if(!$category || ($category=='Best Selling Books')) {
  echo "<h1>Current Best Sellers</h1>";
} else {
  echo '<h1>Current Best Sellers in '.$category.'</h1>";
}
```

showCategories()函数将显示选中种类的列表，我们可以在页面的最上方看到这个列表。getCategoryName()函数返回给定browsenode号条件下的当前种类名称。showBrowseNode()函数将在一个页面中显示该目录下的图书。

下面，我们先来分析showCategories()函数。该函数代码如程序清单33-5所示。

程序清单33-5　categoryfunctions.php函数库中的showCategories()函数——种类列表

```
//display a starting list of popular categories
function showCategories($mode) {
  global $categoryList;
  echo '<hr/><h2>Selected Categories</h2>";

  if($mode == 'Books' ) {

  asort($categoryList);

   $categories = count($categoryList);
   $columns = 4;
   $rows = ceil($categories/$columns);

   echo '<table border=\"0\" cellpadding=\"0\" cellspacing=\"0\"
         width=\"100%\"><tr>';

   reset($categoryList);
```

```
    for($col = 0; $col<$columns; $col++) {
      echo "<td width=\"".(100/$columns).'%" valign=\"top\"><ul>";
      for($row = 0; $row<$rows; $row++) {
        $category = each($categoryList);
        if($category) {
          $browseNode = $category['key'];
          $name = $category['value'];
          echo '<li><span class=\"category\">
           <a href=\"index.php?action=browsenode&browseNode="
           .$browseNode.'\">'.$name.'</a></span></li>";
        }
      }
      echo "</ul></td>";
    }
    echo "</tr></table><hr/>";
  }
```

以上函数使用categoryList数组将browsenode号映射到种类名称，该数组在constants.php函数库中声明。所需的browsenode将被硬编码到这个数组中。该函数还将对这个数组进行排序，并且显示不同的种类。

在主事件循环中，接下来调用的getCategoryName()函数可以用来查询当前正在查看的browsenode名称，这样我们就可以在页面上显示一个标题，例如，“Current Best Sellers in Business & Investing（当前商业和投资领域中最热销的产品）”。它将在前面提到的categoryList数组中查询这个关键字。

在这里，真正有趣的地方是从调用showBrowseNode()函数开始的。这个函数如程序清单33-6所示。

程序清单33-6 bookdisplayfunctions.php函数库中的showBrowseNode()函数——种类列表

```
// For a particular browsenode, display a page of products
function showBrowseNode($browseNode, $page, $mode) {
  $ars = getARS( 'browse' , array('browsenode'=>$browseNode,
          'page' => $page, 'mode'=>$mode));
  showSummary($ars->products(), $page, $ars->totalResults(),
        $mode, $browseNode);
```

这个函数将实现两个功能。首先，它将调用cachefunctions.php函数库中的getARS()函数。getARS()函数将获得并返回一个AmazonResultSet对象（稍后将详细介绍这个对象）。接下来，该函数将调用bookdisplayfunctions.php函数库中的showSummary()函数来显示所获得的信息。

getARS()函数绝对是整个应用程序的关键部分。如果我们按部就班了解其他操作（查看详细信息、图像以及搜索）我们将发现这个函数经常用到。

33.3.3 获得一个AmazonResultSet类

现在，我们来仔细了解getARS()函数，如程序清单33-7所示。

程序清单33-7　cachefunctions.php函数库中的getARS()函数——一个查询的结果集

```
// Get an AmazonResultSet either from cache or a live query
// If a live query add it to the cache
function getARS($type, $parameters) {
  $cache = cached($type, $parameters);
  if ($cache) {
    // if found in cache
    return $cache;
  } else {
    $ars = new AmazonResultSet;
    if($type == 'asin') {
      $ars->ASINSearch(padASIN($parameters['asin']), $parameters['mode']);
    }
    if($type == 'browse') {
      $ars->browseNodeSearch($parameters['browsenode'],
            $parameters['page'], $parameters['mode']);
    }
    if($type == 'search') {
      $ars->keywordSearch($parameters['search'], $parameters['page'],
            $parameters['mode']);
    }
    cache($type, $parameters, $ars);
  }
  return $ars;
}
```

这个函数是用来驱动从Amazon获得数据的操作。它可以以两种方式完成数据获取：从缓存，或者从Amazon实时获得。由于Amazon要求开发人员缓存已经下载的数据，因此这个函数将首先查看保存在缓存中的数据。我们将在稍后的内容详细介绍缓存。

如果我们还没有执行特定的查询操作，就必须从Amazon站点实时获得数据。通过创建一个AmazonResultSet类的实例，并且调用该类中对应于我们需要执行的查询的方法，可以获得实时数据。查询的类型是由$type参数决定的。以种类（或者浏览节点）查询为例，我们将browse传递给该参数——参阅程序清单33-6。如果我们希望执行一个针对特定图书的查询，就应该传递该图书的asin，而如果希望执行一个关键字查询，该参数就必须设置为search。

该参数的每一个不同值都将调用AmazonResultSet类中的不同方法。单个物品搜索将调用ASINSearch()方法。种类搜索将调用browseNodeSearch()方法。而关键字搜索将调用keywordSearch()方法。

下面，我们来仔细了解AmazonResultSet类。该类的所有代码如程序清单33-8所示。

程序清单33-8　AmazonResultSet.php——处理与Amazon站点连接的类

```
<?php
// you can switch between REST and SOAP using this constant set in
// constants.php
if(METHOD--'SOAP') {
```

```
  include_once('nusoap/lib/nusoap.php');
}

// This class stores the result of queries
// Usually this is 1 or 10 instances of the Product class
class AmazonResultSet {
  private $browseNode;
  private $page;
  private $mode;
  private $url;
  private $type;
  private $totalResults;
  private $currentProduct = null;
  private $products = array(); // array of Product objects

  function products() {
    return $this->products;
  }

  function totalResults() {
    return $this->totalResults;
  }

  function getProduct($i) {
  if(isset($this->products[$i])) {
    return $this->products[$i];
  } else {
    return false;
  }
}

// Perform a query to get a page full of products from a browse node
// Switch between XML/HTTP and SOAP in constants.php
// Returns an array of Products
function browseNodeSearch($browseNode, $page, $mode) {

  $this->Service = "AWSECommerceService";
  $this->Operation = "ItemSearch";
  $this->AWSAccessKeyId = DEVTAG;
  $this->AssociateTag = ASSOCIATEID;
  $this->BrowseNode = $browseNode;
  $this->ResponseGroup = 'Large';
  $this->SearchIndex= $mode;
  $this->Sort= 'salesrank';
  $this->TotalPages= $page;

  if(METHOD=='SOAP') {
```

```
        $soapclient = new nusoap_client(
        'http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl',
        'wsdl');

        $soap_proxy = $soapclient->getProxy();

        $request = array ('Service' => $this->Service,
        'Operation' => $this->Operation, 'BrowseNode' => $this->BrowseNode,
        'ResponseGroup' => $this->ResponseGroup, 'SearchIndex' =>
         $this->SearchIndex, 'Sort' => $this->Sort, 'TotalPages' =>
         $this->TotalPages);

        $parameters = array('AWSAccessKeyId' => DEVTAG,
         'AssociateTag' => ASSOCIATEID, 'Request'=>array($request));

        // perform actual soap query
        $result = $soap_proxy->ItemSearch($parameters);

        if(isSOAPError($result)) {
          return false;
        }

        $this->totalResults = $result['TotalResults'];
        foreach($result['Items']['Item'] as $product) {
          $this->products[] = new Product($product);
        }
        unset($soapclient);
        unset($soap_proxy);

      } else {
        // form URL and call parseXML to download and parse it
        $this->url = "http://ecs.amazonaws.com/onca/xml?".
                     "Service=".$this->Service.
                     "&Operation=".$this->Operation.
                     "&AssociateTag=".$this->AssociateTag.
                     "&AWSAccessKeyId=".$this->AWSAccessKeyId.
                     "&BrowseNode=".$this->BrowseNode.
                     "&ResponseGroup=".$this->ResponseGroup.
                     "&SearchIndex=".$this->SearchIndex.
                     "&Sort=".$this->Sort.
                     "&TotalPages=".$this->TotalPages;

        $this->parseXML();
      }

      return $this->products;
    }
```

```
  // Given an ASIN, get the URL of the large image
  // Returns a string
  function getImageUrlLarge($ASIN, $mode) {
    foreach($this->products as $product) {
      if( $product->ASIN()== $ASIN) {
      return $product->imageURLLarge();
    }
  }
  // if not found
  $this->ASINSearch($ASIN, $mode);
  return $this->products(0)->imageURLLarge();
}

// Perform a query to get a products with specified ASIN
// Switch between XML/HTTP and SOAP in constants.php
// Returns a Products object
function ASINSearch($ASIN, $mode = 'books') {
  $this->type = 'ASIN';
  $this->ASIN=$ASIN;
  $this->mode = $mode;
  $ASIN = padASIN($ASIN);

  $this->Service = "AWSECommerceService";
  $this->Operation = 'ItemLookup';
$this->AWSAccessKeyId = DEVTAG;
$this->AssociateTag = ASSOCIATEID;
$this->ResponseGroup = 'Large';
$this->IdType = 'ASIN';
$this->ItemId = $ASIN;

if(METHOD== SOAP') {

  $soapclient = new nusoap_client(
  'http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl',
  'wsdl');

  $soap_proxy = $soapclient->getProxy();

  $request = array ('Service' => $this->Service, 'Operation' =>
  $this->Operation, 'ResponseGroup' => $this->ResponseGroup,
  'IdType' => $this->IdType, 'ItemId' => $this->ItemId);

  $parameters = array('AWSAccessKeyId' => DEVTAG,
   AssociateTag' => ASSOCIATEID, 'Request'=>array($request));

  // perform actual soap query
  $result = $soap_proxy->ItemLookup($parameters);
```

```
    if(isSOAPError($result)) {
      return false;
    }

    $this->products[0] = new Product($result['Items']['Item']);

    $this->totalResults=1;
    unset($soapclient);
    unset($soap_proxy);

  } else {
    // form URL and call parseXML to download and parse it
    $this->url = 'http://ecs.amazonaws.com/onca/xml?'.
                 "Service=".$this->Service.
                 "&Operation=".$this->Operation.
                 "&AssociateTag=".$this->AssociateTag.
                 "&AWSAccessKeyId=".$this->AWSAccessKeyId.
                 "&ResponseGroup=".$this->ResponseGroup.
                 "&IdType=".$this->IdType.
                 "&ItemId=".$this->ItemId;

    $this->parseXML();
  }
  return $this->products[0];
}

// Perform a query to get a page full of products with a keyword search
// Switch between XML/HTTP and SOAP in index.php
// Returns an array of Products
function keywordSearch($search, $page, $mode = 'Books') {

  $this->Service = "AWSECommerceService";
  $this->Operation = "ItemSearch";
  $this->AWSAccessKeyId = DEVTAG;
  $this->AssociateTag = ASSOCIATEID;
  $this->ResponseGroup = "Large";
  $this->SearchIndex= $mode;
  $this->Keywords= $search;

  if(METHOD=='SOAP') {
    $soapclient = new nusoap_client(
     'http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl',
     'wsdl');

    $soap_proxy = $soapclient->getProxy();

    $request = array ('Service' => $this->Service, 'Operation' =>
    $this->Operation, 'ResponseGroup' => $this->ResponseGroup,
```

```
    'SearchIndex  => $this->SearchIndex, 'Keywords' => $this->Keywords);

    $parameters = array('AWSAccessKeyId' => DEVTAG,
    'AssociateTag' => ASSOCIATEID, 'Request'=>array($request));

    // perform actual soap query
    $result = $soap_proxy->ItemSearch($parameters);

    if(isSOAPError($result)) {
      return false;
    }

    $this->totalResults = $result['TotalResults'];

    foreach($result['Items']['Item'] as $product) {
      $this->products[] = new Product($product);
    }
    unset($soapclient);
    unset($soap_proxy);

  } else {

    $this->url = "http://ecs.amazonaws.com/onca/xml?".
                 "Service=".$this->Service.
                 "&Operation=".$this->Operation.
                 "&AssociateTag=".$this->AssociateTag.
                 "&AWSAccessKeyId=".$this->AWSAccessKeyId.
                 "&ResponseGroup=".$this->ResponseGroup.
                 "&SearchIndex=".$this->SearchIndex.
                 "&Keywords=".$this->Keywords;


      $this->parseXML();
    }
    return $this->products;
  }

  // Parse the XML into Product object(s)
  function parseXML() {
    // suppress errors because this will fail sometimes
    $xml = @simplexml_load_file($this->url);
    if(!$xml) {
      //try a second time in case just server busy
      $xml = @simplexml_load_file($this->url);
      if(!$xml) {
      return false;
    }
```

```
        }

        $this->totalResults = (integer)$xml->TotalResults;
        foreach($xml->Items->Item as $productXML) {
          $this->products[] = new Product($productXML);
        }
      }
    }
```

这个类是非常有用的。它以非常好的黑盒方式封装了与Amazon站点的接口。在这个类中，与Amazon站点的连接可以通过REST方法或SOAP方法来实现。所使用的方法通过在最开始的constants.php文件设置的全局常量METHOD来确定。

下面，让我们回到前面的种类搜索例子。我们按如下方式使用AmazonResultSet：

```
$ars = new AmazonResultSet;
$ars->browseNodeSearch($parameters['browsenode'],
                       $parameters['page'],
                       $parameters['mode']);
```

这个类没有构造函数，因此我们直接介绍browseNodeSearch()方法。这里，我们将向该方法传递3个参数：我们所感兴趣的browsenode号（相应地，例如，“Business & Investing”或者“Computers & Internet”）；表示要获得多少记录的页数；以及表示我们所感兴趣的商业类型的模式。该方法的代码如程序清单33-9所示。

程序清单33-9　browseNodeSearch()方法——执行一个种类查询

```
// Perform a query to get a page full of products from a browse node
// Switch between XML/HTTP and SOAP in constants.php
// Returns an array of Products
function browseNodeSearch($browseNode, $page, $mode) {

  $this->Service = "AWSECommerceService";
  $this->Operation = "ItemSearch";
  $this->AWSAccessKeyId = DEVTAG;
  $this->AssociateTag = ASSOCIATEID;
  $this->BrowseNode = $browseNode;
  $this->ResponseGroup = 'Large";
  $this->SearchIndex= $mode;
  $this->Sort= 'salesrank";
  $this->TotalPages= $page;

  if(METHOD=='SOAP') {

    $soapclient = new nusoap_client(
    'http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl',
     wsdl');

    $soap_proxy = $soapclient->getProxy();
```

```
$request = array ('Service' => $this->Service,
'Operation' => $this->Operation, 'BrowseNode' => $this->BrowseNode,
'ResponseGroup' => $this->ResponseGroup, 'SearchIndex' =>
$this->SearchIndex, 'Sort'  => $this->Sort, 'TotalPages' =>
$this->TotalPages);

$parameters = array('AWSAccessKeyId' => DEVTAG,
'AssociateTag' => ASSOCIATEID, 'Request'=>array($request));

// perform actual soap query
$result = $soap_proxy->ItemSearch($parameters);

if(isSOAPError($result)) {
  return false;
}

$this->totalResults = $result['TotalResults'];

foreach($result['Items']['Item'] as $product) {
  $this->products[] = new Product($product);
}
unset($soapclient);
unset($soap_proxy);
  } else {
    // form URL and call parseXML to download and parse it
    $this->url = 'http://ecs.amazonaws.com/onca/xml?'.
                 'Service='.$this->Service.
                 '&Operation='.$this->Operation.
                 '&AssociateTag='.$this->AssociateTag.
                 "&AWSAccessKeyId=".$this->AWSAccessKeyId.
                 "&BrowseNode=".$this->BrowseNode.
                 "&ResponseGroup=".$this->ResponseGroup.
                 "&SearchIndex=".$this->SearchIndex.
                 '&Sort='.$this->Sort.
                 "&TotalPages=".$this->TotalPages;

    $this->parseXML();
  }

  return $this->products;
}
```

根据METHOD常量值的不同，这个方法将执行通过基于SOAP或REST的查询。但是，这两种请求中发送的信息是相同的。如下代码行显示的是请求的变量及其值：

```
$this->Service = "AWSECommerceService";
$this->Operation = "ItemSearch";
```

```
$this->AWSAccessKeyId = DEVTAG;
$this->AssociateTag = ASSOCIATEID;
$this->BrowseNode = $browseNode;
$this->ResponseGroup = "Large";
$this->SearchIndex= $mode;
$this->Sort= "salesrank";
$this->TotalPages= $page;
```

以上代码中的某些变量值是在应用的其他部分设置，例如，$browseNode、$mode和$page的变量值。其他值是常量，例如，DEVTAG和ASSOCIATEID.。而另外一部分的变量$this->Service和$this->Operation以及$this->Sort都是静态变量。

不同的请求类型对变量的要求是不同。以上示例用来浏览根据销售排名的特定节点。查询特定物品和关键字的变量是不同的。在AmazonResultSet.php文件的browseNodeSearch()函数、ASINSearch()函数以及keywordSearch()函数中，你可以找到这些变量的列表。所有请求类型所需变量的详细信息可以在AWS开发人员指南找到。

接下来，我们分别介绍browseNodeSearch()函数中的REST和SOAP查询请求的创建。ASINSearch()函数和keywordSearch()函数的请求创建格式在概念上都是相似的。

33.3.4 使用REST发送和接收请求

在browseNodeSearch()函数（或ASINSearch()，keywordSearch()）设置了所需的类成员变量后，通过HTTP使用REST/XML还要做的就是格式化数据并发送给URL，如下所示：

```
$this->url = 'http://ecs.amazonaws.com/onca/xml?".
             "Service=".$this->Service.
             "&Operation=".$this->Operation.
             '&AssociateTag='.$this->AssociateTag.
             "&AWSAccessKeyId=".$this->AWSAccessKeyId.
             '&BrowseNode='.$this->BrowseNode.
             "&ResponseGroup=".$this->ResponseGroup.
             "&SearchIndex='.$this->SearchIndex.
             "&Sort='.$this->Sort.
             "&TotalPages=".$this->TotalPages;
```

在上面的例子中，URL是http://ecs.amazonaws.com/onca/xml。对这个URL，你可以在URL字符串后附加变量名称及其值来组成一个GET查询字符串。关于相关信息以及其他可能变量的完整文档可以在AWS开发人员指南找到。在设置所有变量后，你可以调用：

```
$this->parseXML();
```

来完成实际的操作。parseXML()方法如程序清单33-10所示。

程序清单33-10　parseXML()方法——分析由查询返回的XML

```
// Parse the XML into Product object(s)
function parseXML() {
  // suppress errors because this will fail sometimes
```

```
$xml = @simplexml_load_file($this->url);
if(!$xml) {
  //try a second time in case just server busy
  $xml = @simplexml_load_file($this->url);
  if(!$xml) {
    return false;
  }
}

$this->totalResults = (integer)$xml->TotalResults;
foreach($xml->Items->Item as $productXML) {
  $this->products[] = new Product($productXML);
}
```

simplexml_load_file()函数完成了大部分的操作。它从一个文件读入XML内容，在这个例子中，是从一个URL读入的。它为数据和XML文档中的结构提供了一个面向对象接口。对数据来说，这是一个非常有用的接口，但是由于我们希望一个接口函数集能够处理来自REST或SOAP方法的数据，所以可以为Product类示例中的相同数据创建自己的面向对象接口。请注意，在REST版本中，可以将XML中的属性转换成PHP变量类型。在PHP中，不必使用cast操作符，但是在这个例子中，如果不给出cast操作符，将看到每一个数据的对象表示，这是没有用的。

Product类包含了大多数的访问器函数，这些函数可以用来访问保存在私有成员中的数据，因此打印整个文件并没有很大意义。Product类的结构和构造函数还是值得介绍的。程序清单33-11包含了Product类的部分定义。

程序清单33-11 Product类封装了一个Amazon产品的所有信息

```
class Product {
  private $ASIN;
  private $productName;
  private $releaseDate;
  private $manufacturer;
  private $imageUrlMedium;
  private $imageUrlLarge;
  private $listPrice;
  private $ourPrice;
  private $salesRank;
  private $availability;
  private $avgCustomerRating;
  private $authors = array();
  private $reviews = array();
  private $similarProducts = array();
  private $soap; // array returned by SOAP calls

  function __construct($xml) {
    if(METHOD=='SOAP') {
```

```
$this->ASIN = $xml['ASIN'];
$this->productName = $xml['ItemAttributes']['Title'];

if (is_array($xml['ItemAttributes']['Author']) != "") {
    foreach($xml['ItemAttributes']['Author'] as $author) {
      $this->authors[] = $author;
    }
} else {
    $this->authors[] = $xml['ItemAttributes']['Author'];
}
$this->releaseDate = $xml['ItemAttributes']['PublicationDate'];
$this->manufacturer = $xml['ItemAttributes']['Manufacturer'];
$this->imageUrlMedium = $xml['MediumImage']['URL'];
$this->imageUrlLarge = $xml['LargeImage']['URL'];

$this->listPrice = $xml['ItemAttributes']['ListPrice']['FormattedPrice'];
$this->listPrice = str_replace('$', '', $this->listPrice);
$this->listPrice = str_replace(',', '', $this->listPrice);
$this->listPrice = floatval($this->listPrice);

$this->ourPrice = $xml['OfferSummary']['LowestNewPrice']['FormattedPrice'];
$this->ourPrice = str_replace('$', '', $this->ourPrice);
$this->ourPrice = str_replace(',', '', $this->ourPrice);
$this->ourPrice = floatval($this->ourPrice);

    $this->salesRank = $xml['SalesRank'];
    $this->availability =
$xml['Offers']['Offer']['OfferListing']['Availability'];
    $this->avgCustomerRating = $xml['CustomerReviews']['AverageRating'];

    $reviewCount = 0;

    if (is_array($xml['CustomerReviews']['Review'])) {
        foreach($xml['CustomerReviews']['Review'] as $review) {
          $this->reviews[$reviewCount]['Rating'] = $review['Rating'];
          $this->reviews[$reviewCount]['Summary'] = $review['Summary'];
          $this->reviews[$reviewCount]['Content'] = $review['Content'];
          $reviewCount++;
        }
    }

    $similarProductCount = 0;

    if (is_array($xml['SimilarProducts']['SimilarProduct'])) {
        foreach($xml['SimilarProducts']['SimilarProduct'] as $similar) {
          $this->similarProducts[$similarProductCount]['Title'] =
$similar['Title'];
          $this->similarProducts[$similarProductCount]['ASIN'] =
```

```
$review['ASIN'];
            $similarProductCount++;
          }
        }


    } else {
      // using REST

      $this->ASIN = (string)$xml->ASIN;
      $this->productName = (string)$xml->ItemAttributes->Title;
      if($xml->ItemAttributes->Author) {
        foreach($xml->ItemAttributes->Author as $author) {
          $this->authors[] = (string)$author;
        }
      }
      $this->releaseDate = (string)$xml->ItemAttributes->PublicationDate;
      $this->manufacturer = (string)$xml->ItemAttributes->Manufacturer;
      $this->imageUrlMedium = (string)$xml->MediumImage->URL;
      $this->imageUrlLarge = (string)$xml->LargeImage->URL;

      $this->listPrice = (string)$xml->ItemAttributes->ListPrice->FormattedPrice;
      $this->listPrice = str_replace('$', '', $this->listPrice);
      $this->listPrice = str_replace(',', '', $this->listPrice);
      $this->listPrice = floatval($this->listPrice);

      $this->ourPrice = (string)$xml->OfferSummary->LowestNewPrice->
FormattedPrice;
      $this->ourPrice = str_replace('$', '', $this->ourPrice);
      $this->ourPrice = str_replace(',', '', $this->ourPrice);
      $this->ourPrice = floatval($this->ourPrice);

      $this->salesRank = (string)$xml->SalesRank;
      $this->availability = (string)$xml->Offers->Offer->OfferListing->
Availability;
      $this->avgCustomerRating = (float)$xml->CustomerReviews->AverageRating;

      $reviewCount = 0;

      if($xml->CustomerReviews->Review) {
        foreach ($xml->CustomerReviews->Review as $review) {
          $this->reviews[$reviewCount]['Rating'] = (float)$review->Rating;
          $this->reviews[$reviewCount]['Summary'] = (string)$review->Summary;
          $this->reviews[$reviewCount]['Content'] = (string)$review->Content;
          $reviewCount++;
        }
      }
```

```
    $similarProductCount = 0;

    if($xml->SimilarProducts->SimilarProduct) {
      foreach ($xml->SimilarProducts->SimilarProduct as $similar) {
        $this->similarProducts[$similarProductCount]['Title'] =
             (string)$similar->Title;
        $this->similarProducts[$similarProductCount]['ASIN'] =
             (string)$similar->ASIN;
        $similarProductCount++;
      }
    }

  }
}

// most methods in this class are similar
// and just return the private variable
function similarProductCount() {
  return count($this->similarProducts);
}

function similarProduct($i) {
  return $this->similarProducts[$i];
}

function customerReviewCount() {
  return count($this->reviews);
}

function customerReviewRating($i) {
  return $this->reviews[$i]['Rating'];
}

function customerReviewSummary($i) {
  return $this->reviews[$i]['Summary'];
}

function customerReviewComment($i) {
  return $this->reviews[$i]['Content'];
}

function valid() {
  if(isset($this->productName) && ($this->ourPrice>0.001) &&
          isset($this->ASIN)) {
    return true;
  } else {
    return false;
  }
```

```
}

function ASIN() {
  return padASIN($this->ASIN);
}

function imageURLMedium() {
  return $this->imageUrlMedium;
}

function imageURLLarge() {
  return $this->imageUrlLarge;
}

function productName() {
  return $this->productName;
}

function ourPrice() {
  return number_format($this->ourPrice,2, '.', '');
}

function listPrice() {
  return number_format($this->listPrice,2, '.', '');
}

function authors() {
  if(isset($this->authors)) {
    return $this->authors;
  } else {
    return false;
  }
}

function releaseDate() {
  if(isset($this->releaseDate)) {
    return $this->releaseDate;
  } else {
    return false;
  }
}

function avgCustomerRating() {
  if(isset($this->avgCustomerRating)) {
    return $this->avgCustomerRating;
  } else {
    return false;
  }
```

```
}

function manufacturer() {
  if(isset($this->manufacturer)) {
    return $this->manufacturer;
  } else {
    return false;
  }
}

function salesRank() {
  if(isset($this->salesRank)) {
    return $this->salesRank;
  } else {
    return false;
  }
}

function availability() {
  if(isset($this->availability)) {
    return $this->availability;
  } else {
    return false;
  }
}
```

这个类的构造函数需要两个不同形式的输入数据并且能够创建一个应用程序接口。请注意，虽然某些处理代码可以更普通，但是根据方法的不同，某些重要的属性，例如评价，具有不同的名称。

了解获得数据的全部步骤后，我们回到getARS()函数，接着就是showBrowseNode()。下一步是：

```
showSummary($ars->products(), $page,
            $ars->totalResults(), $mode,
            $browseNode);
```

showSummary()函数只显示了AmazonResultSet的信息，如程序清单33-1所示的。因此，这里我们没有再次包含该函数。

33.3.5　使用SOAP发送和接收请求

下面，我们来了解SOAP版本的browseNodeSearch()函数。如下所示，我们将再次给出其代码：

```
$soapclient - new nusoap_client(
  'http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl',
```

```
    'wsdl );

$soap_proxy = $soapclient->getProxy();

$request = array ('Service'  => $this->Service, 'Operation' => $this->Operation,
    'BrowseNode' => $this->BrowseNode, 'ResponseGroup' => $this->ResponseGroup,
    'SearchIndex'  -> $this->SearchIndex, 'Sort' => $this->Sort,
    'TotalPages' -> $this->TotalPages);

$parameters = array('AWSAccessKeyId' => DEVTAG, 'AssociateTag' => ASSOCIATEID,
    'Request'=>array($request));

// perform actual soap query
$result = $soap_proxy->ItemSearch($parameters);

if(isSOAPError($result)) {
    return false;
}

$this->totalResults = $result['TotalResults'];

foreach($result['Items']['Item'] as $product) {
    $this->products[] = new Product($product);
}
unset($soapclient);
```

在以上代码中，没有其他特殊的函数——SOAP客户端为我们完成了所有“工作”。

我们以创建一个SOAP客户端实例为开始：

```
$soapclient = new nusoap_client(
    'http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl',
    'wsdl' );
```

这里，我们向客户端提供了两个参数。第一个就是服务的WSDL描述，而第二个参数可以告诉SOAP客户端这是一个WSDL URL。或者，我们也可以只提供一个参数：服务的终点，这是SOAP服务器的直接URL。

我们选择提供一个参数是有充分原因的，如下代码所示：

```
$soap_proxy = $soapclient->getProxy();
```

这一行代码将根据WSDL文档中的信息创建了一个类，这个类就是SOAP的代理，它拥有对应于Web服务中的方法。这可以使我们的工作变得简单一些。我们可以像与一个本地的PHP类进行交互一样与Web服务进行交互。

接下来，我们设置一个需要传递给browsenode查询的参数数组：

```
$request = array ('Service' => $this->Service, 'Operation' => $this->Operation,
    'BrowseNode' -> $this->BrowseNode,  'ResponseGroup' => $this->ResponseGroup,
    'SearchIndex' => $this->SearchIndex, 'Sort'  => $this->Sort,
    'TotalPages' -> $this->TotalPages);
```

发送给该请求还需要其他两个元素：AWSAccessKeyID和and AssociateTag。这些元素以及$Request对象中的元素数组将保存在$parameters数组参数中。

```
$parameters = array('AWSAccessKeyId' => DEVTAG, 'AssociateTag' => ASSOCIATEID,
  'Request'=>array($request));
```

使用proxy类，你可以只调用Web服务的方法，将该参数数组传递给Web服务的方法：

```
$result = $soap_proxy->ItemSearch($parameters);
```

保存在$result中的数据是一个数组，我们可以将其直接保存为AmazonResultSet类的products数组中的一个Product对象。

33.3.6 缓存请求返回的数据

现在，我们回到getARS()函数，介绍一些关于缓存的内容。回忆一下，这个函数如下所示：

```
// Get an AmazonResultSet either from cache or a live query
// If a live query add it to the cache
function getARS($type, $parameters) {

  $cache = cached($type, $parameters);

  if ($cache) {
    // if found in cache
    return $cache;
  } else {
    $ars = new AmazonResultSet;
    if($type == 'asin') {
      $ars->ASINSearch(padASIN($parameters['asin']), $parameters['mode']);
    }
    if($type == 'browse') {
      $ars->browseNodeSearch($parameters['browsenode'], $parameters['page'],
          $parameters['mode']);
    }
    if($type == 'search') {
      $ars->keywordSearch($parameters['search'], $parameters['page'],
          $parameters['mode']);
    }
    cache($type, $parameters, $ars);
  }
  return $ars;
```

所有应用程序的SOAP缓存和XML缓存都是通过这个函数实现的。我们还有另一种缓存方法。首先，我们调用cached()方法判断所需的AmazonResultSet对象是否已经被缓存。如果已经被缓存，将直接返回数据，而不是生成一个新的请求：

```
$cache = cached($type, $parameters);
if($cache) // if found in cache{
  return $cache;
```

```
}
```

如果没有被缓存，就必须从Amazon获得数据，然后再将其添加到缓存中：

```
cache($type, $parameters, $ars);
```

下面，让我们来看看这两个函数：cached()和cache()，如程序清单33-12所示。这些函数实现了Amazon所要求的缓存技术。

程序清单33-12　cached()和cache()函数——cachefunctions.php的缓存函数

```
// check if Amazon data is in the cache
// if it is, return it
// if not, return false
function cached($type, $parameters) {
  if($type == 'browse') {
    $filename = CACHE.'/browse.'.$parameters['browsenode'].'.'
                .$parameters['page'].'.'.$parameters['mode'].'.dat';
  }
  if($type == 'search') {
    $filename = CACHE.'/search.'.$parameters['search'].'.'
                .$parameters['page'].'.'.$parameters['mode'].'.dat';
  }
  if($type == 'asin') {
    $filename = CACHE.'/asin.'.$parameters['asin'].'.'
                .$parameters['mode'].'.dat';
  }

  // is cached data missing or > 1 hour old?
  if(!file_exists($filename) ||
      ((mktime() - filemtime($filename)) > 60*60)) {
    return false;
  }
  $data = file_get_contents($filename);
  return unserialize($data);
}

// add Amazon data to the cache
function cache($type, $parameters, $data) {
  if($type == 'browse') {
    $filename = CACHE.'/browse.'.$parameters['browsenode'].'.'
                .$parameters['page'].'.'.$parameters['mode'].'.dat';
  }
  if($type == 'search') {
    $filename = CACHE.'/search.'.$parameters['search'].'.'
                .$parameters['page'].'.'.$parameters['mode'].'.dat';
  }
  if($type == 'asin') {
    $filename = CACHE.'/asin.'.$parameters['asin'].'.'
                .$parameters['mode'].'.dat';
```

```
}

$data = serialize($data);

$fp = fopen($filename, 'wb');
if(!$fp || (fwrite($fp, $data)==-1)) {
  echo ('<p>Error, could not store cache file');
}
fclose($fp);
```

仔细查看以上代码，可以发现，缓存文件保存在一个由查询类型以及查询参数组成的文件名称的文件中。cache()函数通过对这些数据进行序列化操作，从而保存了这些数据。cached()函数也将在一个小时后覆盖这些数据，这是Amazon协议所规定的。

serialize()函数将所保存的程序数据转换成一个可以单独保存的字符串。在这个例子中，我们为AmazonResultSet对象创建了一个表示给对象的、可以用于保存的格式。调用unserialize()函数可以完成相反的操作，它可以将保存的数据转换成内存中的数据结构。请注意，像这样的反序列化一个对象意味着必须拥有该类的文件定义，这样在该类被载入后，该类就可以被理解并使用。

在我们的应用程序中，从缓存获取结果只需要很短的时间，而执行一个实时查询则要十秒钟的时间。

33.3.7　创建购物车

到这里，我们了解了Amazon的所有查询功能，可以用它们来做些什么呢？很明显，我们可以使用这些功能来创建一个购物车。由于我们已经在第28章中详细介绍了它，因此在这里不再详细介绍。

购物车函数的代码如程序清单33-13所示。

程序清单33-13　cartfunctions.php——实现购物车

```
<?php
require_once('AmazonResultSet.php');

// Using the function showSummary() in the file bookdisplay.php display
// the current contents of the shopping cart
function showCart($cart, $mode) {
  // build an array to pass
  $products = array();
  foreach($cart as $ASIN=>$product) {
    $ars = getARS('asin', array('asin'=>$ASIN, 'mode'=>$mode));
    if($ars) {
      $products[] = $ars->getProduct(0);
    }
  }
  // build the form to link to an Amazon.com shopping cart
```

```
echo "<form method=\'POST\"
       action=\"http://www.amazon.com/gp/aws/cart/add.html\">";

foreach($cart as $ASIN=>$product) {
  $quantity = $cart[$ASIN]['quantity'];
  echo "<input type=\'hidden\" name=\'ASIN.".$ASIN."\"
        value=\'".$ASIN."\">";
  echo '<input type=\'hidden\' name=\"Quantity.".$ASIN."\'
        value=\'".$quantity."\">";
}

echo "<input type=\"hidden\" name=\"SubscriptionId\'
        value=\'".DEVTAG."\'>
      <input type=\"hidden\" name=\'AssociateTag\"
        value=\'".ASSOCIATEID."\'>
      <input type=\"image\" src=\'images/checkout.gif\"
        name=\'submit.add-to-cart\" value=\'Buy
        From Amazon.com\'>
      When you have finished shopping press checkout to add all
      the items in your Tahuayo cart to your Amazon cart and
      complete your purchase.
      </form>
      <br/><a href=\'index.php?action=emptycart\"><img
        src=\"images/emptycart.gif\' alt=\"Empty Cart\"
        border=\"0\'></a>
      If you have finished with this cart, you can empty it
      of all items.
      </form>
      <br />
      <h1>Cart Contents</h1>";

  showSummary($products, 1, count($products), $mode, 0, true);

}

// show the small overview cart that is always on the screen
// only shows the last three items added
function showSmallCart() {
  global $_SESSION;

  echo "<table border=\'1\" cellpadding=\"1\' cellspacing=\"0\">
        <tr><td class=\"cartheading\'>Your Cart $'.
      number_format(cartPrice(), 2).'</td></tr>
      <tr><td class=\"cart\">".cartContents()."</td></tr>";

  // form to link to an Amazon.com shopping cart
  echo "<form method=\"POST\"
```

```
                action=\"http://www.amazon.com/gp/aws/cart/add.html\'>
        <tr><td class=\"cartheading\"><a
                href=\"index.php?action=showcart\"><img
                src=\"images/details.gif\' border=\'0\"></a>";

  foreach($_SESSION['cart'] as $ASIN=>$product) {
    $quantity = $_SESSION['cart'][$ASIN]['quantity'];
    echo '<input type=\"hidden\" name=\"ASIN.".$ASIN."\"
            value=\"".$ASIN."\">";
    echo "<input type=\'hidden\' name=\"Quantity.".$ASIN."\"
              value=\"'.$quantity."\">";
  }
  echo "<input type=\"hidden\" name=\'SubscriptionId\"
            value=\"".DEVTAG."\">
        <input type=\"hidden\" name=\'AssociateTag\'
            value=\'".ASSOCIATEID."\">
        <input type=\"image\" src=\"images/checkout.gif\"
              name=\"submit.add-to-cart\" value=\"Buy From
              Amazon.com\">
        </td></tr>
        </form>
        </table>";
}

// show last three items added to cart
function cartContents() {
  global $_SESSION;

  $display = array_slice($_SESSION['cart'], -3, 3);
  // we want them in reverse chronological order
  $display = array_reverse($display, true);

  $result = '';
  $counter = 0;

  // abbreviate the names if they are long
  foreach($display as $product) {
    if(strlen($product['name'])<=40) {
      $result .= $product['name'].'<br />';
    } else {
      $result .= substr($product['name'], 0, 37).'...<br />';
    }
    $counter++;
  }

  // add blank lines if the cart is nearly empty to keep the
  // display the same
```

```
  for(;$counter<3; $counter++) {
    $result .= "<br />";
  }
  return $result;
}

// calculate total price of items in cart
function cartPrice() {
  global $_SESSION;
  $total = 0.0;
  foreach($_SESSION['cart'] as $product) {
    $price = str_replace('$', '', $product['price']);
    $total += $price*$product['quantity'];
  }

  return $total;
}

// add a single item to cart
// there is currently no facility to add more than one at a time
function addToCart(&$cart, $ASIN, $mode) {
  if(isset($cart[$ASIN] )) {
    $cart[$ASIN]['quantity'] +=1;
  } else {
    // check that the ASIN is valid and look up the price
    $ars = new AmazonResultSet;
    $product = $ars->ASINSearch($ASIN, $mode);

    if($product->valid()) {
      $cart[$ASIN] = array('price'=>$product->ourPrice(),
            'name' => $product->productName(), 'quantity' => 1) ;
    }
  }

}

// delete all of a particular item from cart
function deleteFromCart(&$cart, $ASIN) {
  unset ($cart[$ASIN]);
}
```

在这里，我们实现购物车的方法与前面稍微有些不同，例如addToCart()函数。当我们向购物车添加一个物品时，可以检查它是否具有一个有效的ASIN并且查找当前的价格（至少是缓存的价格）。

真正有趣的地方是这个问题：当客户付账时，如何让数据到达Amazon?

33.3.8　到Amazon付账

请仔细阅读程序清单33-13所示的showCart()函数。如下是相关的一部分：

```
// build the form to link to an Amazon.com shopping cart
echo "<form method=\'POST\'
       action=\"http://www.amazon.com/gp/aws/cart/add.html\">";
foreach($cart as $ASIN=>$product) {
  $quantity = $cart[$ASIN]['quantity'];
  echo '<input type=\"hidden\' name=\'ASIN.".$ASIN."\'
        value=\"".$ASIN."\">";
  echo '<input type=\"hidden\' name=\'Quantity.".$ASIN."\'
        value=\"".$quantity."\">";
}

echo '<input type=\"hidden\' name=\'SubscriptionId\'
        value=\"".DEVTAG."\">
      <input type=\"hidden\' name=\"AssociateTag\'
        value=\"".ASSOCIATEID."\">
      <input type=\"image\" src=\"images/checkout.gif\'
            name=\'submit.add-to-cart\" value=\"Buy
            From Amazon.com\">
       When you have finished shopping press checkout to add all
       the items in your Tahuayo cart to your Amazon cart and
       complete your purchase.
      </form>";
```

付账按钮是一个表单按钮，它可以将购物车连接到客户在Amazon站点上的购物车。我们通过POST变量发送ASIN、数量和我们的会员ID。通过点击本章开始处的图33-5所示的每一个链接，可以看到最终结果。

这个接口的一个困难在于这是一个单向的交互。我们将物品添加到Amazon的购物车，但不能从Amazon购物车中删除物品。这就意味着人们无法在我们的站点和Amazon站点之间来回浏览，并且删除购物车中重复的物品。

33.4　安装项目代码

如果希望安装本章的项目代码，必须进行一些必要的操作。将代码保存在服务器的合适位置后，我们还必须完成以下操作：

- 创建一个缓存目录。
- 设置这个缓存目录的读写权限，这样脚本就可以对其进行写操作。
- 编辑constants.php代码，使其提供缓存的位置。
- 注册一个Amazon开发人员令牌。
- 编辑constants.php代码，使其包括开发人员令牌，此外，还可以选择是否包括会员ID。
- 确认已经安装了NuSOAP。我们将其安装在Tahuayo目录下，但是也可以将其安装在不同位置，并对其进行修改。

■ 检查PHP5是否是在带有XML支持的条件下进行编译的。

33.5 扩展这个项目

你可以很容易对这个项目进行如下方面的扩展：扩展Tahuayo站点所提供的搜索类型。更多信息，请参阅Amazon的Web服务资源中心提供的关于创新性示例应用的链接以及“文章与教程”和“社区代码”部分。

购物车是使用Amazon Web服务的最基本功能，但它并不是唯一的功能。

33.6 进一步学习

关于XML和Web服务，互联网上有大量的图书和在线资源。通常，W3C是一个非常不错的起点，可以在该站点找到关于XML工作组页面，其URL如下所示：http://www.w3.org/XML/Core/。

以及Web服务事件页：http://www.w3.org/2002/ws/。

这里仅仅是一个开始。