第34章　使用Ajax构建Web 2.0应用

互联网是以一系列包含文本和图像、音频以及视频文件链接的静态页面组成。如今，尽管大多数页面都加入了通过服务器端脚本生成的文本和多媒体信息，但在很大程度上互联网还是处于这样的状况。这些通过服务器端脚本生成的文本和多媒体信息也就是本书所介绍的应用所创建的。但是，Web 2.0的出现可以引领开发人员探索与Web服务器和数据库进行用户交互的新方法，而正是这些Web服务器和数据库保存了我们所需要的信息。使用Ajax（异步JavaScript和XML）编程进行交互是如今日益流行的方法，它在减少读取静态元素时间的同时改进了交互性。

注意　要更好理解Web 2.0的概念，请参阅Tim O'Reilly的文章：http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html。

本章将介绍Ajax编程的基础以及可以集成到Web应用的Ajax示例元素。本章并不会全面介绍Ajax编程，只是为在未来工作中使用该技术提供一个基础。本章将包括如下：

- 整合脚本和置标语言创建Ajax应用。
- 一个Ajax应用的基本组成部分，包括发送请求以及解析来自服务器端的响应。
- 如何修改前面章节介绍应用的元素来创建支持Ajax的页面。
- 代码库以及如何需求帮助信息。

34.1　Ajax是什么

Ajax本身不是一个编程语言，也不是一个单一的技术。通常，Ajax编程是将处理XML格式数据传输的客户端JAVASCRIPT编程与某种服务器端编程语言（例如，PHP）的结合。此外，XHTML和CSS也被用来展示支持Ajax的元素。

典型的，Ajax编程的结果是为一个交互式应用提供更加清晰和快捷的用户接口——例如，连接到FACEBOOK，FLICKER以及其他处于Web 2.0前沿的社交网站的接口。这些应用支持用户不需要重新刷新或载入整个页面来执行许多任务，这也正是Ajax的用武之地。客户端的编程将调用许多服务器端的编程，但是只有用户浏览器所显示的特定区域才是会被从新刷新的。这种动作结果模拟了在单个应用才会产生的动作结果，但却是在Web环境下发生的。

一个常见的例子是，处理工作单的脱机应用和查看网站上信息丰富的表格。在脱机应用中，用户可以修改一个单元格并将公式应用到其他单元格，或者点击一个列进行排序，所有这些操作不需要离开最初的界面。在一个静态的Web环境中，点击一个链接对一列进行排序可能需要给服务器发送一个新请求，服务器返回一个新结果给浏览器，浏览器重新刷新该页面并展示给用户。在支持Ajax的Web环境中，表格可以根据用户的请求进行排序，不需要重新载入整个页面。

接下来的内容将介绍使用Ajax时需要涉及的不同技术。这些信息也不是全面的，文中将给

出一些相关的资源。

34.1.1 HTTP请求和响应

超文本传输协议，或者HTTP，是一个Internet标准，它定义了Web服务器和Web浏览器之间相互交流的方式。当用户在Web浏览器的地址栏输入一个URL请求Web页面，或者点击一个链接，提交一个表单，或者执行任何能将用户带到新页面的时候，浏览器将发送一个HTTP请求。

这个请求将发送给Web服务器，服务器将返回许多可能响应的一种。要获得来自Web服务器的可理解响应，必须正确地创建请求。当使用Ajax的时候，了解请求和相应的格式是非常关键的，因为开发人员必须在Ajax应用中编写正确的请求以及期望特定的返回结果。

当发送HTTP请求时，客户端将用以下格式发送信息。

- 开始行，包含了发送方法，资源路径以及使用的HTTP版本，如下示例：

```
GET http://server/phpmysql4e/chapter34/test.html HTTP/1.1
```

其他常见的方法包括POST和HEAD。

- 可选的报头行，以“参数:值”格式出现，如下示例：

```
User agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.0.1)
Gecko/2008070208 Firefox/3.0.1
```

以及/或者：

```
Accept: text/plain, text/html
```

关于HTTP报头，请参阅：http://www.w3.org/Protocols/rfc2616/。

- 空白行
- 可选的消息正文（Message Body）

发送HTTP请求后，客户端将接收到一个HTTP响应。

HTTP响应的格式如下所示：

- 开始行，或者状态行，包含了所使用的HTTP版本以及响应码，如下示例：

```
HTTP/1.1 200 OK
```

状态码的第一个数字（这里是200中的2）给出响应的类型。以1为开始的状态码表示响应为信息性的，2表示请求成功，3表示请求被重定向，4表示客户端错误，例如404表示客户端请求缺少相关内容项，5表示服务器端错误，例如，500表示非正常的脚本。

关于HTTP状态码的完整列表，请参阅：http://www.w3.org/Protocols/rfc2616/。

- 可选的报头行，以“参数:值”格式出现，如下示例：

```
Server: Apache/2.2.9
Last-Modified: Fri, 1 Aug 2008 15:34:59 GMT
```

34.1.2 DHTML和XHTML

动态HTML，或者DHTML，是一个用来描述将静态HTML，级联样式单（CSS）以及JavaScript结合，并且在载入一个静态Web页面所有元素后，通过文档对象模型修改页面外观的

术语。表面上看，这个功能非常类似支持Ajax的站点页面，事实上，在一定程度的确如此。它们的区别就在于服务器和客户端之间的异步连接，也就是，Ajax中的字母“A”（异步）。

尽管DHTML驱动的站点可以在导航的下拉列表显示动态变化或者根据用户选择改变表单元素，但是这些元素的所有数据都是已经从服务器获得的。例如，如果要设计一个用户将鼠标滑过一个链接或按钮时显示文本第一部分并且滑过另一个链接或按钮时显示文本第二部分的DHTML网站，第一部分和第二部分的文本事实上已经载入到用户的浏览器中。开发人员可以通过JavaScript编程根据用户鼠标的动作设置CSS的可见性（Visibility）属性来显示或隐藏文本。在支持Ajax的站点中，根据远程调用服务器端的脚本执行结果，填充为第一部分和第二部分预留的区域，而页面的其他区域保持不变。

可扩展的超文本置标语言，或者XHTML，通过客户端设备（Web浏览器，手机或者其他手持设备）来显示标记的内容，并且支持集成CSS来提供显示的额外控制。这些功能与HTML和DHTML是类似的。XHTML和HTML的不同之处在于XHTML遵循XML的语法，以及除了标准的Web浏览工具外，XHTML还可以被XML工具来解析。

XHMTL的元素和属性全部是小写字母（例如，`<head></head>`，而不是`<HEAD></HEAD>`；`href`而不是`HREF`）。此外，所有属性值必须封闭在单引号或双引号内，所有元素必须显式封闭——通过标记对里的结束标记或者在单一元素内，例如，`<img />`或`<br />`。

关于XHTML的详细信息，请参阅：http://www.w3.org/TR/xhtml1/。

34.1.3　级联样式单

级联样式单（CSS）用来进一步定义静态、动态以及支持Ajax页面的显示。使用CSS允许开发人员修改一个文档内部（该样式单）的标记、类以及ID的定义，从而使该修改立即在所有连接到该样式单的页面体现。通过选择器（Selector），声明（Declaration）以及值（Value），这些定义，或者规则遵循如下特定格式：

- 选择器是HTML标记的名称，例如body或h1（标题级别1）。
- 声明是样式单本身的属性，例如background或font-size。
- 值与声明相对应，例如white或12pt。

如下所示的是一个样式单示例，它定义了文档正文背景颜色为白色，文档内所有文本为常规粗细，12磅大小，Verdana或sans-serif字体：

```
body {
     background: white [or #fff or #ffffff];
     font-family: Verdana, sans-serif;
     font-size: 12pt;
     font-weight: normal;
}
```

当要展示该页面的某个元素，如果该元素的样式在该样式单定义，这些值就会产生效果。例如，当遇到h1，客户端将显示h1文本，然而，h1的样式已经被定义——可能使用大于12pt的字体大小并且使用Bold属性。

除了定义选择器，也可以在样式单中定义自定义的类和ID。使用类（可以在一个页面的多

个元素使用）或ID（只能在一个页面中使用一次），还可以进一步定义站点内所显示元素的外观以及功能。这种重定义对于支持Ajax的站点尤为重要，因为文档的预定义区域可以显示通过远程脚本动作获取的新信息。

类的定义类似于选择器的定义——名称定义后需要花括号，定义内容需要分号间隔。如下所示的是ajaxarea类的定义：

```
.ajaxarea {
     width: 400px;
     height: 400px;
     background: #fff;
     border: 1px solid #000;
}
```

在该例中，当将该类应用到div容器中，将产生一个400像素（宽）×400像素（高）的矩形，其背景颜色为白色，并具有细的黑边框。其使用如下所示：

```
<div class="ajaxarea"> some text</div>
```

使用样式单的最常见方法是创建一个包含了所有样式定义的单独文件，将该文件链接到HTML文档，如下所示：

```
<head>
<link rel="stylesheet' href='the_style_sheet.css" type="text/css">
</head>
```

关于CSS的详细信息，请参阅：http://www.w3.org/TR/CSS2/。

34.1.4　客户端编程

客户端编程通常在整个页面从Web服务器获取之后，在客户端Web浏览器中发生。所有的程序函数都包含在从Web服务器获取的数据中，等待客户端的调用。在客户端，常见的动作包括显示或隐藏部分的文本或图像，修改文本或图像的颜色、大小或位置，执行计算或者在用户将表单发送给Web服务器处理之前验证用户的表单输入。

最常见的客户端脚本编程语言是JavaScript——Ajax中的字母“J”。VBScript也是一种客户端脚本编程语言，但它是微软特有的，因此它并不是针对所有操作系统和Web浏览器的好选择。

34.1.5　服务器端编程

服务器端编程包括所有存在于Web服务器并且在给客户端发送响应之前需要解释或编译的脚本。通常，服务器端编程包括服务器端与数据库的连接；因此，与数据库之间的请求和响应也成为脚本本身一部分。

这些脚本可以用任何服务器端语言编写，例如，Perl、JSP、ASP或PHP——显然，本章将使用PHP作为服务器端编程的语言。通常，由于服务器端的响应是在某些标准HTML的标记区域显示数据，因此需要考虑终端用户的环境。

34.1.6 XML和XSLT

在本书的第33章介绍了XML，其中涵盖了XML的格式、结构和使用。在Ajax应用的上下文中，XML就是Ajax的字母“x”，它主要用于交换数据；XSLT用来操作数据。数据本身将通过所创建的Ajax应用发送或获取。

关于XML的详细信息，请参阅：http://www.w3.org/XML/。关于XSLT的详细信息，请参阅：http://www.w3.org/TR/xslt20/。

34.2 Ajax基础

到这里，已经了解了一个Ajax应用的可能组成部分，本节将使用这些组成部分创建一个可用的示例程序。请记住，使用Ajax的一个主要原因：创建一个能够响应用户动作的交互式网站，而这些响应并不需要刷新整个页面。

要实现此功能，Ajax应用需要包括一个能够处理发生在所请求的Web页面与负责生成该页面的Web服务器之间额外层。这个额外层通常就是指Ajax框架（也就是Ajax引擎）。该框架用来处理终端用户和Web服务器之间的请求，它可以不需要额外的动作就交流请求和响应，例如重新刷新页面，或者中断用户正在执行的动作（例如，滚动、点击或阅读一段文本）。

接下来的内容将介绍使用Ajax应用的不同组成部分创建一个流畅的用户体验。

34.2.1 XMLHTTPRequest对象

本章前面的内容已经介绍了HTTP请求和响应以及如何在一个Ajax应用中使用客户端编程。当连接Web服务器以及发送一个不需要重新载入整个原始页面的请求时，XMLHTTPReuqest对象是非常重要的，它也是JavaScript的特有对象。

注意　由于安全的原因，该对象只能调用同一域的URL；不能直接调用远程服务器。

XMLHTTPRequest对象也被看作是Ajax应用的“通道（guts）”，因为它是客户端请求和服务器端响应的“通道”。虽然接下来将介绍创建和使用XMLHTTPRequest对象的基础知识，请参阅http://www.w3.org/TR/XMLHttpRequest/获取详细信息。

XMLHTTPRequest对象具有一些属性，如表34-1所示。

表34-1　XMLHTTPRequest对象的属性

属　性	描　述
Onreadystatechange	指定当readyState属性变化时应该调用的函数
readyState	请求的状态。整数0表示未初始化，1表示正在载入，2表示已载入，3表示交互，4表示已完成
responseText	包含以字符串形式返回的数据
responseXml	包含以XML格式文档对象返回的数据
status	服务器返回的HTTP状态码，例如200
statusText	服务器返回的HTTP状态字符串，例如OK

XMLHTTPRequest对象具有一些方法，如表34-2所示。

表34-2　XMLHTTPRequest对象的方法

方　法	描　述
abort()	停止一个请求
getAllResponseHeaders()	以字符串的形式返回响应的所有报头
getResponseHeader(*header*)	以字符串的形式返回响应的报头“header”的值
open('*method*', '*URL*', '*a*')	指定HTTP请求的方法（参数值包括POST、GET和HEAD），目标URL以及该请求是否为异步（参数值为“true”或“false”）
send(*content*)	使用POST方法发送该请求，参数为可选
setRequestHeader('x', 'y')	设置一个参数对（x为参数名，y为参数值），并且作为报头参数发送请求

在使用XMLHTTPRequest对象之前，必须创建该对象的实例。如下语句所示：

```
var request = new XMLHTTPRequest();
```

尽管以上代码可以在非IE的浏览器中工作，但是理想情况下，该代码应该可以支持所有用户。因此，如下所示的代码可以在所有浏览器下创建XMLHTTPRequest对象的实例：

```
function getXMLHTTPRequest() {
   var req = false;
   try {
      /* for Firefox */
      req = new XMLHttpRequest();
   } catch (err) {
      try {
         /* for some versions of IE */
         req = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (err) {
         try {
            /* for some other versions of IE */
            req = new ActiveXObject("Microsoft.XMLHTTP");
         } catch (err) {
            req = false;
         }
      }
   }
   return req;
}
```

如果将以上代码保存在一个文件ajax_functions.js中并且将其保存在Web服务器上，这样就创建了一个Ajax函数库。

当需要在Ajax应用中创建一个XMLHTTPRequest对象的实例，可以在代码中引入（include）包含所需函数的文件。

```
<script src="ajax_functions.js" type="text/javascript"></script>
```

接下来调用这个新对象以及继续编写应用的代码，如下所示：

```
<script type="text/javascript">
```

```
var myReq = getXMLHTTPRequest();
</script>
```

接下来的内容将介绍在Ajax函数文件中添加新的函数。

34.2.2 与服务器通信

在前面内容介绍的例子中，它的主要功能是创建一个新的XMLHTTPRequest对象；还没有执行任何使用该对象进行真正通信的操作。在接下来的例子中，将创建一个能够向服务器发送请求的JavaScript函数，具体地说是向servertime.php脚本发送请求。

```
function getServerTime() {
  var thePage = 'servertime.php';
  myRand = parseInt(Math.random()*999999999999999);
  var theURL = thePage +"?rand="+myRand;
  myReq.open("GET", theURL, true);
  myReq.onreadystatechange = theHTTPResponse;
  myReq.send(null);
}
```

函数的第一行创建一个名为thePage的变量，变量值为servertime.php。这是位于服务器端的脚本文件名称。

第二行代码作用不重要，因为它只是创建一个随机数。可能会有问题“一个随机数与服务器时间有什么关系？”答案是它对脚本本身并没有任何直接的影响。创建随机数并将其附加在URL中（如第三行代码所示）的原因是避免浏览器（或者代理服务器）缓存该请求带来的任何问题。如果URL只是http://yourserver/yourscript.php，服务器返回的结果可能会被缓存。然而，如果URL是http://yourserver/yourscript.php?rand=randval，浏览器或代理服务器就没法缓存，因为每次的URL都是不同的，而函数的实际功能是没有变化的。

该函数的最后三行代码使用了XMLHTTPRequest对象实例的三个方法（open，onreadystatechange和send），该XMLHTTPRequest对象是通过前面示例的getXMLHTTPRequest()方法获得。

在调用open()方法的代码行中，该方法的参数是请求类型（“GET”），URL（“theURL”），以及“true”表示该请求为异步的。

在调用onreadystatechange()方法的代码行中，当对象状态发生变化时，该函数调用一个新的函数——theHTTPResponse()。

在调用send()方法的代码行中，该函数向服务器端脚本发送了空的内容（NULL）。

到这里，创建一个名为servertime.php的文件，并且输入如程序清单34-1所示代码：

程序清单34-1　servertime.php代码

```
<?php
header('Content-Type: text/xml');
echo '<?xml version="1.0" ?>
      <clock>
        <timestring>It is ".date('H:i:s').' on '.date('M d, Y').'.</timestring>
```

```
        </clock>";
?>
```

该脚本将通过PHP的date()函数的调用获得服务器的当前时间，并且以XML编码的字符串返回。需要注意的是，date()函数被调用了两次；一次是date('H:i:s')，它将以24小时格式返回服务器当前时间的小时，分钟和秒。而以date('M d, Y')格式调用将返回脚本被调用时的月份、日期和年份。

返回结果的XML字符串如下示例所示，其中方括号中的字符将被真实值所代替：

```
<?xml version='1.0' ?>
<clock>
   <timestring>
   It is [time] on [date].
   </timestring>
</clock>
```

接下来的内容将介绍如何创建剩下的函数——theHTTPResponse()，并且通过服务器上的脚本对响应进行一些处理。

34.2.3 处理服务器响应

前面介绍的getServerTime()函数可以调用theHTTPResponse()函数并且对返回的字符串数据进行一些操作。如下例将解释响应并向终端用户显示一个字符串：

```
function theHTTPResponse() {
  if (myReq.readyState == 4) {
    if(myReq.status == 200) {
       var timeString =
              myReq.responseXML.getElementsByTagName("timestring")[0];
       document.getElementById('showtime').innerHTML =
              timeString.childNodes[0].nodeValue;
    }
  } else {
    document.getElementById('showtime').innerHTML =
           '<img src="ajax-loader.gif"/>';
  }
}
```

函数中的if...else语句将检查对象的状态；如果对象状态不是4（已完成），函数将显示一段动画（()。然而，如果myReq的readState是4，将进一步检查来自服务器的状态是不是200（OK）。

如果状态码是200，将创建一个新的变量：timeString。该变量将被赋值为从服务器端脚本返回的XML字符串中的timestring元素值，而该元素值是通过调用myReq对象的responseXML属性的getElementByTagname()方法获得，如下代码所示：

```
var timeString = myReq.responseXML.getElementsByTagName("timestring")[0];
```

下一步就是在HTML文件的CSS预定义区域显示该值。在这个例子中，时间值将显示在名

为showtime的文档元素中：

```
document.getElementById('showtime').innerHTML =
   timeString.childNodes[0].nodeValue;
```

到这里，就完成了ajax_functions.js脚本，如程序清单34-2所示。

程序清单34-2　ajax_functions.js代码

```
function getXMLHTTPRequest() {
   var req = false;
   try {
      /* for Firefox */
      req = new XMLHttpRequest();
   } catch (err) {
      try {
         /* for some versions of IE */
         req = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (err) {
         try {
            /* for some other versions of IE */
            req = new ActiveXObject("Microsoft.XMLHTTP");
         } catch (err) {
            req = false;
         }
      }
   }

   return req;
}

function getServerTime() {
  var thePage = 'servertime.php';
  myRand = parseInt(Math.random()*999999999999999);
  var theURL = thePage +"?rand="+myRand;
  myReq.open("GET", theURL, true);
  myReq.onreadystatechange = theHTTPResponse;
  myReq.send(null);
}

function theHTTPResponse() {
  if (myReq.readyState == 4) {
    if(myReq.status == 200) {
       var timeString =
            myReq.responseXML.getElementsByTagName("timestring")[0];
        document.getElementById('showtime').innerHTML =
            timeString.childNodes[0].nodeValue;
    }
  } else {
```

```
        document.getElementById('showtime').innerHTML =
                '<img src="ajax-loader.gif"/>';
    }
}
```

接下来的内容将完成该HTML并且结合所有部分创建一个Ajax应用。

34.2.4 整合应用

本章前面已经介绍，Ajax是一个整合的技术。在上一节，已经介绍了如何使用JavaScript和PHP（客户端和服务器端编程）发送HTTP请求并获取响应。没有介绍的部分是关于页面的显示：使用XHTML和CSS产生用户所见到的结果。

程序清单34-3给出了ajaxServerTime.html的代码，该文件包含了样式单以及能够调用远程PHP脚本并获取服务器响应的JavaScript脚本。

程序清单34-3　ajaxServerTime.html的代码

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
'http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" dir="ltr" lang="en">
<head>

<style>
body {
  background: #fff;
  font-family: Verdana, sans-serif;
  font-size: 12pt;
  font-weight: normal;
}

.displaybox {
  width: 300px;
  height: 50px;
  background-color:#ffffff;
  border:2px solid #000000;
  line-height: 2.5em;
  margin-top: 25px;
  font-size: 12pt;
  font-weight: bold;
}
</style>

<script src="ajax_functions.js" type="text/javascript"></script>
<script type="text/javascript">
var myReq = getXMLHTTPRequest();
</script>
```

```
</head>

<body>

<div align="center">
   <h1>Ajax Demonstration</h1>
   <p align='center'>Place your mouse over the box below
   to get the current server time.<br/>
   The page will not refresh; only the contents of the box
   will change.</p>
   <div id="showtime" class="displaybox'
         onmouseover='javascript:getServerTime();"></div>
</div>

</body>
</html>
```

以上代码以XHTML声明为开始，后面跟着<html>和<head>标记。在文档的head区域，插入了以<style></style>封闭的样式单定义。这里只定义了两项：boday标记内所有内容的格式以及使用displaybox类的元素格式。Displaybox类定义为300像素宽，50像素高的白色背景矩形，并且带有黑色边框。此外，该矩形内所有内容将是12 pt的粗体字。

定义样式单内容后，在head元素中给出指向JavaScript函数库的链接。

```
<script src="ajax_functions.js' type="text/javascript'></script>
```

如下所示代码创建了一个新XMLHTTPRequest对象并赋值给myReq变量：

```
<script type='text/javascript">
var myReq = getXMLHTTPRequest();
</script>
```

这样就结束了head元素的定义，开始定义body元素。在body元素内，只是给出了XHTML内容。在一个对齐属性为居中的div元素中，可以找到该页面的标题文本（“Ajax Demonstration”）以及指导用户将鼠标移至文本框来获得服务器当前时间的信息。

在div元素中的showtime属性是真正定义了脚本应该执行的动作，具体的是，onmouseover事件处理器。

```
<div id="showtime' class="displaybox'
onmouseover-"javascript:getServerTime();'></div>
```

onmouseover事件处理器的使用意味着，当用户鼠标进入名为showtime的div元素，调用getServerTime()函数。调用该函数将初始化发送给服务器的请求，服务器响应以及出现在div元素中的结果文本。

注意　JavaScript函数可以通过多种方法调用，例如，通过表单按钮的onclick事件。

图34-1、图34-2以及图34-3所示的是脚本实际执行时这些事件的调用顺序。ajaxServerTime.html页面的内容不会重新载入，只有showtime这个div元素中的内容将重新载入。

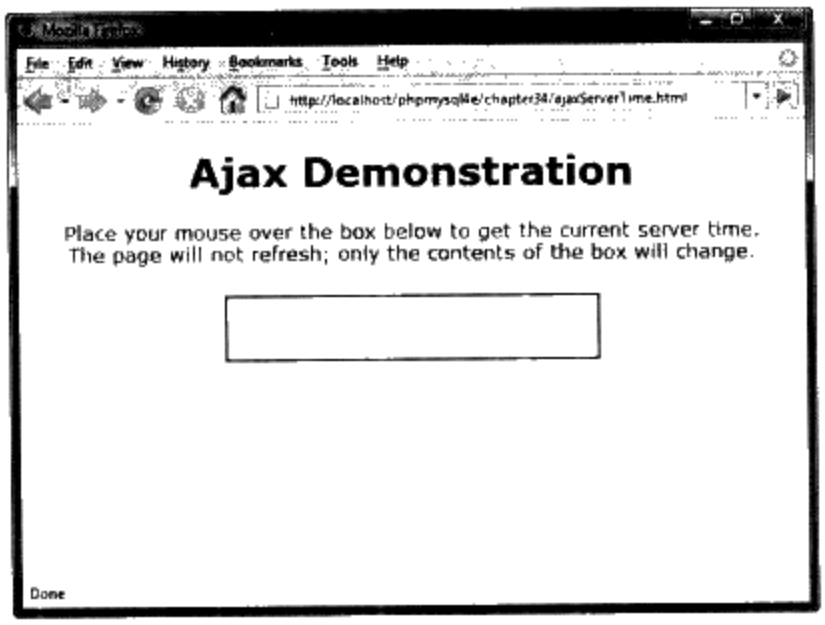

图34-1　最初载入ajaxServerTime.html页面所显示的指示信息以及空白文本框

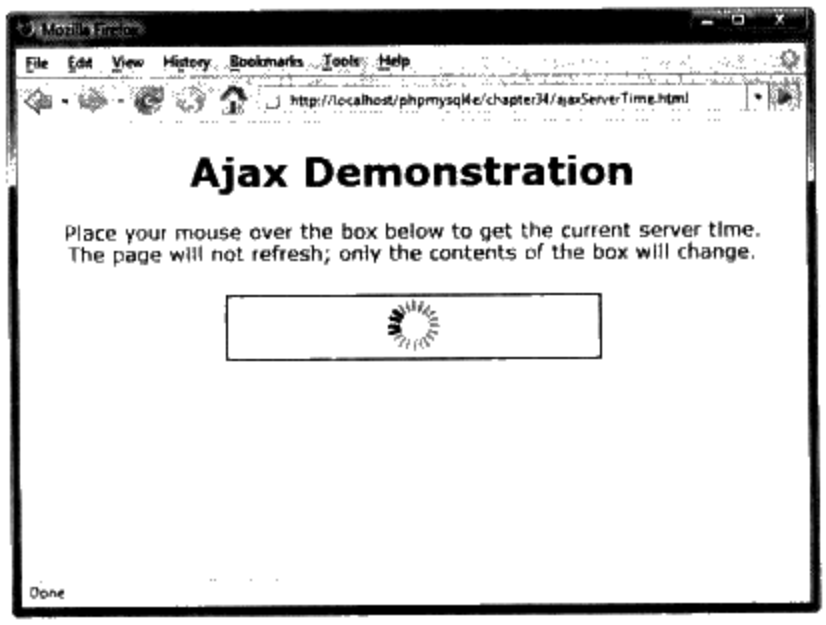

图34-2　用户鼠标移过该区域将开始请求；图中所示图标表示对象正在载入

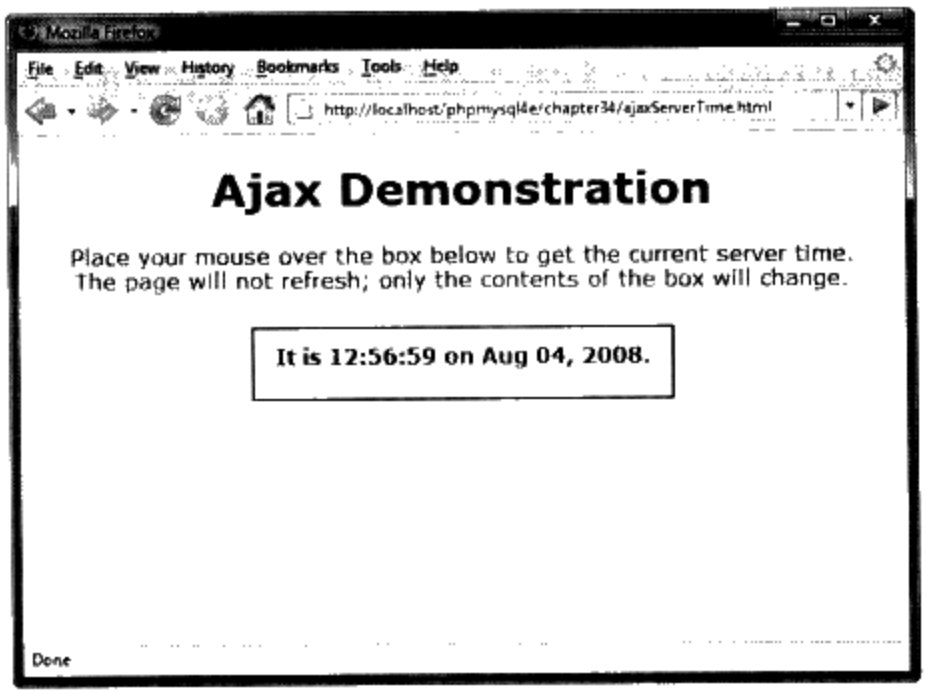

图34-3　服务器端的结果显示在名为showtime的div元素中；再次将鼠标移过该区域将产生脚本的再次调用

34.3 在以前的项目添加Ajax元素

本书第五篇所介绍的任何项目都不是支持Ajax的。这些项目都包含了一系列的表单提交以及页面的重新载入。尽管页面包含动态元素，但是没有一个提供了Web 2.0应用带来的流畅的用户体验。

当然，在这些项目中添加Ajax元素可能会将重点从使用PHP和MySQL创建Web应用转移到Ajax。换句话说，必须先学会走，才能开始学跑。但是，既然已经知道如何跑（或者能蹦一点）可以开始考虑修改这些应用的元素来引入Ajax，或者开始考虑在未来将要创建的新应用引入Ajax元素。

Ajax开发人员的思考过程大概是这样的：什么是独有的用户操作，什么样的页面动作将调用这些操作？例如，是否期望用户通过点击一个按钮元素提交一个表单，然后再继续下一步操作，或者是否可以通过改变一个表单元素（文本输入框，单选框或复选框）的焦点来调用一个发送给服务器的异步请求？在确定了需要包括的动作类型后，可以编写这些调用PHP脚本的JavaScript函数，而PHP脚本将处理这些发送给服务器的请求或来自服务器的结果。

接下来的内容将介绍如何在本书前面章节创建的脚本中添加Ajax元素。

在PHPBookmark应用中添加Ajax元素

在第27章中，创建了一个名为PHPbookmark应用。该应用要求用户在保存书签并查看其他用户所保存的书签推荐之前进行用户注册和登录。

由于这个应用已经创建并且由一些联系紧密的PHP脚本和函数库组成，要做的第一步就是考虑如何在已有结构中添加额外的文件——是否某些文件是样式单、JavaScript函数或者处理服务器端动作的PHP脚本。回答是非常简单的：为样式单创建一个单独文件，为所有的JavaScript函数创建一个单独文件。然后，在第27章的已有PHP脚本添加一些代码来引入这两个外部文件，如果必要，也可以直接加入JavaScript函数本身的调用。任何新创建的PHP脚本将区别于已有的应用文件，单独存在。

在确定如何管理新创建的文件后，可以开始考虑哪些用户函数需要支持Ajax。尽管该应用的用户注册和登录是支持Ajax的主要部分，但是这里，主要将围绕添加和编辑用户保存的书签部分转换成支持Ajax的部分进行介绍。

此外，还需要对已有的应用文件进行修改。将第27章的应用文件复制到一个新目录是必要的，这样可以方便本章的使用。本章所做的任何修改都将上传到这个目录，而不是PHPBookmark应用的工作目录中。

注意　如果要将应用的用户注册部分支持Ajax，可以通过也能够调用PHP脚本的JavaScript函数来验证用户的电子邮件以及用户名称是否已经存在于系统中。如果已经存在，该函数将显示一个错误信息，因此在这些错误被修改之前，不会继续注册的提交过程。

1. 创建额外的文件

正如前面提到的，我们将在已有的应用结构中添加新的文件。尽管在接下来的内容会逐步将新加文件添加到应用中，但记住这一点也是有必要的。

假设至少有两个新文件：一个样式单和一个JavaScript函数库。现在先创建两个名为new_ss.css和new_ajax.js的新文件。由于新样式单还没有定义任何新样式，因此该文件为空。而new_ajax.js文件包含了本章前面所创建的`getXMLHTTPRequest()`函数，该函数可以在所有浏览器中创建一个`XMLHTTPRequest`对象实例。虽然还要在该文件添加更多的代码，但是现在就可以将该脚本上传到Web服务器。

接下来需要在PHPBookmark应用的某一个已有的显示函数中添加这两个文件的链接。这样就可以像JavaScript函数库中的函数一样，确保样式单中的样式定义是可以使用的。如果回忆第27章内容，控制`head`元素输出的函数名为`do_html_header()`，它位于output_fns.php文件中。

该函数的新版本如程序清单34-4所示。

程序清单34-4　修改后的do_htmp_header()函数包含了指向新样式单和JavaScript函数库的链接

```
function do_html_header($title) {
  // print an HTML header
?>
  <html>
  <head>
    <title><?php echo $title;?></title>
    <style>
      body { font-family: Arial, Helvetica, sans-serif; font-size: 13px; }
      li, td { font-family: Arial, Helvetica, sans-serif; font-size: 13px; }
      hr { color: #3333cc; }
      a { color: #000000; }
      </style>

      <link rel='stylesheet' type='text/css' href='new_ss.css"/>
      <script src="new_ajax.js' type='text/javascript"></script>

    </head>
    <body>
    <img src='bookmark.gif' alt="PHPbookmark logo" border="0"
         align="left' valign='bottom" height='55" width='57" />
    <h1>PHPbookmark</h1>
      <hr />
    <?php
      if($title) {
        do_html_heading($title);
    }
}
```

在上载了新的样式单文件，JavaScript函数库，output_fns.php文件以及在PHPBookmark系统中打开任何页面，这些新文件都应该被正确引入，没有任何错误。接下来，将在这两个文件分别添加新的样式单和脚本，并创建一些Ajax功能。

2. 以Ajax方式添加书签

目前，添加书签的操作出现在用户输入书签URL并点击表单提交按钮。点击表单提交按钮的动作将调用另一个PHP脚本，该脚本将添加书签，返回用户的书签列表，并且显示新添加的书签。换句话说，页面将被重新载入。而Ajax方式是给出一个添加书签的表单，通过后台调用一个JavaScript函数来调用PHP脚本，从而将该书签添加在数据库中并且将响应返回给用户，它不需要重新载入页面的表单提交按钮——所有这些操作都不需要离开已经被载入的页面。这个新函数将请求output_fns.php脚本中的display_add_bm_form()函数。

程序清单34-5所示的是修改后的函数。这里，删除了表单动作，添加了一个id为文本输入框的元素，以及修改了按钮元素的属性。此外，还添加了对getXMLHTTReuquest()函数的调用。

程序清单34-5　修改后的display_add_bm_form()函数

```
function display_add_bm_form() {
  // display the form for people to enter a new bookmark in
?>
<script type='text/javascript'>
var myReq = getXMLHTTPRequest();
</script>
<form>
<table width='250' cellpadding="2" cellspacing="0" bgcolor="#cccccc'>
<tr><td>New BM:</td>
<td><input type="text" name="new_url' name='new_url" value="http://"
     size-"30  maxlength="255'/></td></tr>
<tr><td colspan="2" align='center'>
    <input type="button" value='Add bookmark"
           onClick=" javascript:addNewBookmark();'/></td></tr>
</table>
</form>
<?php
}
```

请仔细查看按钮元素，如下所示：

```
<input type-"button' value='Add bookmark"
         onClick=" javascript:addNewBookmark();'/>
```

当该按钮被点击，onClick事件处理器将调用addNewBookmark()函数。该函数将向服务器发送一个请求，具体的，它将向尝试在数据库里插入一条记录的PHP脚本发送请求。该函数代码如程序清单34-6所示。

程序清单34-6　addNewBookmark()函数

```
function addNewBookmark() {
  var url = "add_bms.php";
  var params = 'new_url=' + encodeURI(document.getElementById('new_url').value);
  myReq.open("POST", url, true);
  myReq.setRequestHeader("Content-type", 'application/x-www-form-urlencoded");
  myReq.setRequestHeader("Content-length", params.length);
  myReq.setRequestHeader("Connection", 'close');
  myReq.onreadystatechange = addBMResponse;
  myReq.send(params);
}
```

以上函数类似于本章前面使用过的getServerTime()函数。其逻辑非常类似：创建变量，向PHP脚本发送数据以及调用函数处理服务器端的响应。

如下所示代码根据表单域的名称和用户输入值创建了一个名称/值对：

```
var params = "new_url=" + encodeURI(document.getElementById('new_url').value);
```

函数的最后一行代码将params参数值发送给后台的PHP脚本，如下所示：

```
myReq.send(params);
```

在发送之前，还将发送3个请求报头，这样服务器端就知道如何处理POST请求中的数据，如下所示：

```
myReq.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
myReq.setRequestHeader("Content-length", params.length);
myReq.setRequestHeader("Connection", "close");
```

接下来就是创建JavaScript函数来处理服务器响应；这里定义为addBMResponse。

```
myReq.onreadystatechange = addBMResponse;
```

同样的，以上代码类似于本章前面介绍过的theHTTPResponse函数。程序清单34-7所示的是该函数代码。

程序清单34-7　addBMResponse()函数代码

```
function addBMResponse() {
  if (myReq.readyState == 4) {
    if(myReq.status == 200) {
      result    myReq.responseText;
      document.getElementById('displayresult').innerHTML = result;
    } else {
      alert('There was a problem with the request.');
    }
  }
}
```

该函数将首先检查对象的状态；如果请求已经完成，将检查来自服务器端的响应码是否是200（OK）。如果响应码不是200，函数将给出一个警告："There was a problem with the request（该请求存在问题）"。任何其他的响应都将来自add_bms.php脚本的执行，并且将现在id值为displayresult的div元素中。这里，displayresult定义在new_ss.css样式单中，如下所示（白色背景）：

```
#displayresult {
   background: #fff;
}
```

在PHP函数display_add_bm_form()的form标记结束前，添加了如下所示的代码，该代码将显示用户将看到的服务器端结果：

```
<div id="displayresult"></div>
```

接下来，将对已有代码add_bms.php进行修改。

3. 对已有代码进行修改

如果希望通过不修改add_bms.php脚本来添加书签，检查用户权限和添加书签的操作就足够了。然而，这样操作所返回的结果可能会不好理解，如图34-4所示。图中的标题和logo以及脚注链接都出现了两次，同时该应用的显示也出现一些问题。

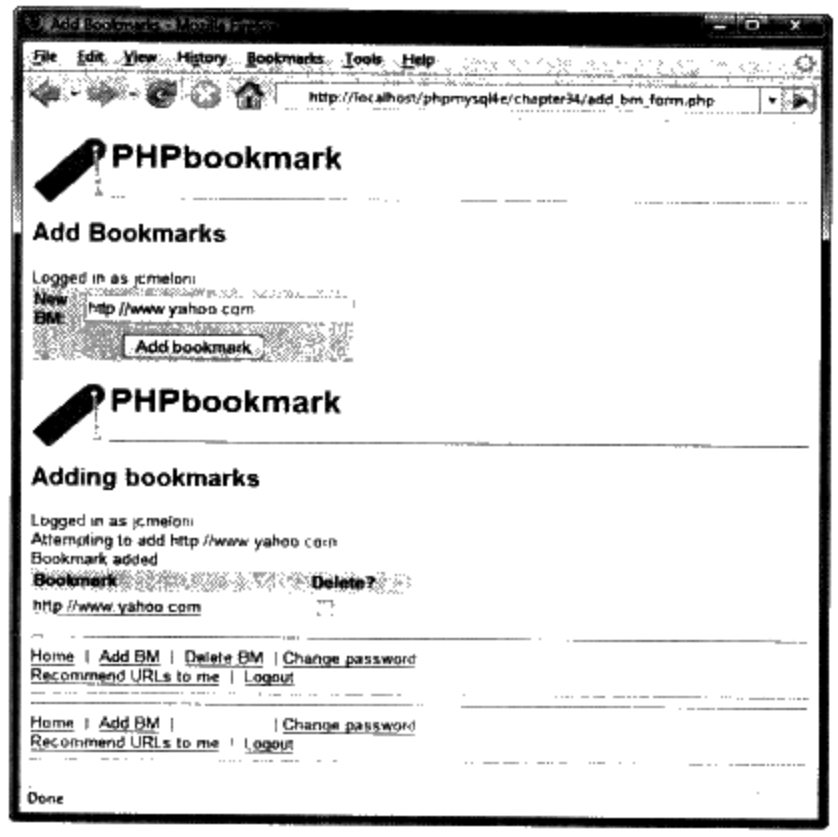

图34-4　在修改add_bms.php脚本前添加书签

在PHPBookmark应用的非Ajax版本中，该表单只是一个页面，提交后的返回结果是另一个页面，而且所有页面元素都将实时的重新载入。然而，在支持Ajax的环境中，添加新书签，获得服务器结果以及是否继续添加新书签都不需要重新载入任何页面元素。这个新功能需要对add_bms.php代码进行一定的修改。最初的代码如程序清单34-8所示。

程序清单34-8　最初的add_bms.php代码

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

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

 do_html_header('Adding bookmarks');

 try {
   check_valid_user();
   if (!filled_out($_POST)) {
     throw new Exception('Form not completely filled out.');
   }
   // check URL format
   if (strstr($new_url, 'http:// ') === false) {
     $new_url = 'http://'.$new_url;
   }

   // check URL is valid
```

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

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

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

该代码的第一行引入了所需的bookmark_fns.php文件。如果查看bookmark_fns.php文件代码，该文件还将调用其他文件：

```
<?php
  // We can include this file in all our files
  // this way, every file will contain all our functions and exceptions
  require_once('data_valid_fns.php');
  require_once('db_fns.php');
  require_once('user_auth_fns.php');
  require_once('output_fns.php');
  require_once('url_fns.php');
?>
```

在支持Ajax的环境中，添加书签的操作可能不需要（也可能需要）这些所有引入文件，但就像该文件头的注释提到——每一个文件都包含了所有注释和异常。在这种情况下，由于应用架构由一系列的动态页面转移到支持Ajax的功能环境，在确认不需要某些代码功能之前，最好还是保留这些额外的元素。请保留add_bms.php的第一行代码。

第二行代码将开始或者继续一个用户会话，同样应该保留；即使是在支持Ajax的环境中，这个动作对安全的完整性是有意义的。同样的，第三行代码也可以保留。它给出了在请求发送中POST参数值，如下所示：

```
$new_url = $_POST['new_url'];
```

最后，可以删除如下所示的代码行：

```
do_html_header('Adding bookmarks');
```

由于已经正在浏览包含HTML报头信息的页面（add_bm_form.php），就不需要重复载入该页面了——不需要转移到其他页面。这种重复将产生两套页面标题和Logo图像，正如图34-4所

示。类似的原因，也可以删除add_bm_form.php最后两行的代码，如下所示：

```
display_user_menu();
do_html_footer();
```

如果删除这些元素，上载文件到服务器以及添加其他书签，其结果都会是期望的，当然还是需要修改一些脚本。图34-5所示的就是对脚本修改后的应用运行的结果。

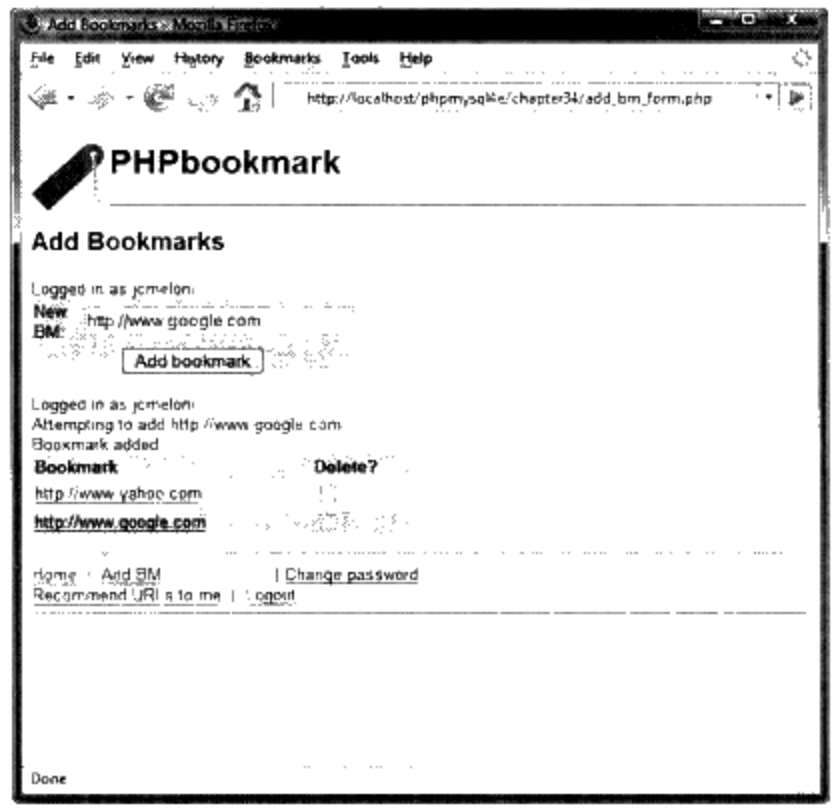

图34-5　修改add_bms.php脚本后添加书签的运行结果

在结果页面上，用户登录状态信息仍然是重复的，但这个问题已经没有以前明显了。接下来要删除这些重复的信息并且修改一些与异常处理相关的功能，这些功能在支持Ajax的环境中才有真正的意义。

要删除用户登录的重复信息，可以删除add_bms.php脚本中的如下代码行：

```
check_valid_user();
```

有效用户的检查已经在add_bms_form.php页面载入时执行；当进入这个已经调用了Ajax函数的页面时，就不用再担心用户是否有效。

接下来删除try语句块和异常处理。删除它们的原因是希望脚本结束运行时能够将已经保存的URL列表显示给用户。这就意味着某些修改和调整可能会引入一些新的错误信息文本。程序清单34-9所示的是修改后的add_bms.php代码。

程序清单34-9　修改后的add_bms.php代码

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

//create short variable name
$new_url = $_POST['new_url'];
//check that form has been completed
if (!filled_out($_POST)) {
  //has not
  echo "<p class=\"warn\">Form not completely filled out.</p>";
} else {
  // has; check and fix URL format if necessary
  if (strstr($new_url, 'http://') === false) {
    $new_url = 'http://'.$new_url;
  }

  // continue on to check URL is valid
  if (!(@fopen($new_url, 'r'))) {
```

```
        echo '<p class=\'warn\">Not a valid URL.</p>";
      } else {
        //it is valid, so continue to add it
        add_bm($new_url);
        echo "<p>Bookmark added.</p>";
      }
    }
    // regardless of the status of the current request
    // get the bookmarks this user has already saved
    if ($url_array = get_user_urls($_SESSION[ valid_user'])) {
       display_user_urls($url_array);
    }
  ?>
```

以上代码仍然延续了可能事件的逻辑顺序，但是能够显示适当的错误信息，不会显示任何重复的页面元素。

以上代码首先将检查表单本身是否已填写。如果没有完全填写，将在书签添加表单和用户已有书签列表之间显示错误信息。如图34-6所示的就是这个响应。

其次，以上代码将检查URL格式是否正确；如果不正确，该URL字符串将被转换成正确的URL格式并继续执行。接下来，将打开一个Socket测试该URL是否有效。如果测试失败，将在书签添加表单和用户已有书签列表之间显示错误信息。然而，如果URL是有效的，它将被添加到用户已有书签列表。如图34-7所示的是添加一个无效URL时，应用给出的响应信息。

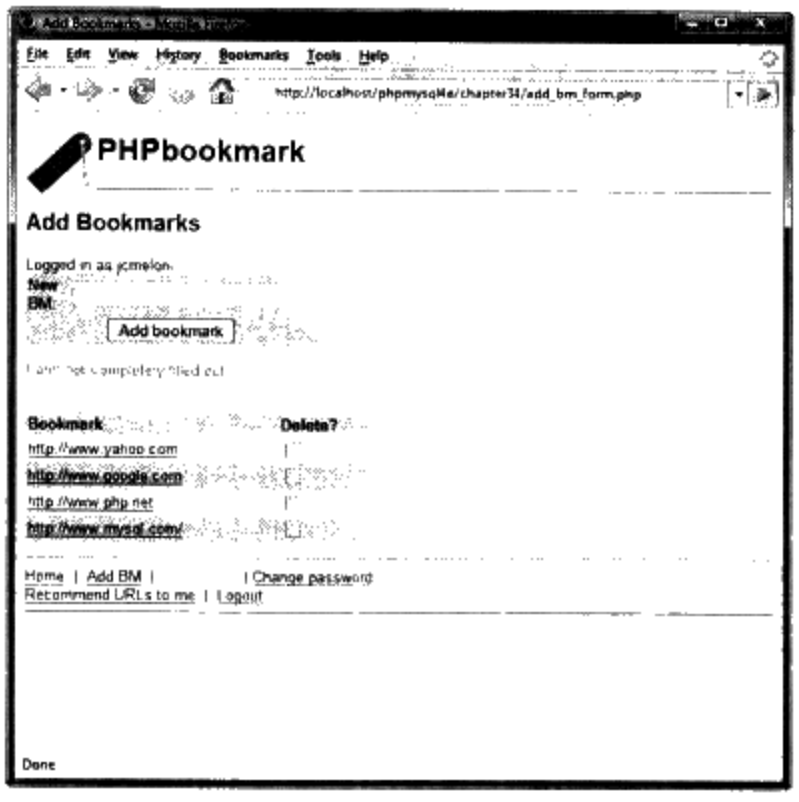

图34-6 尝试添加一个空值

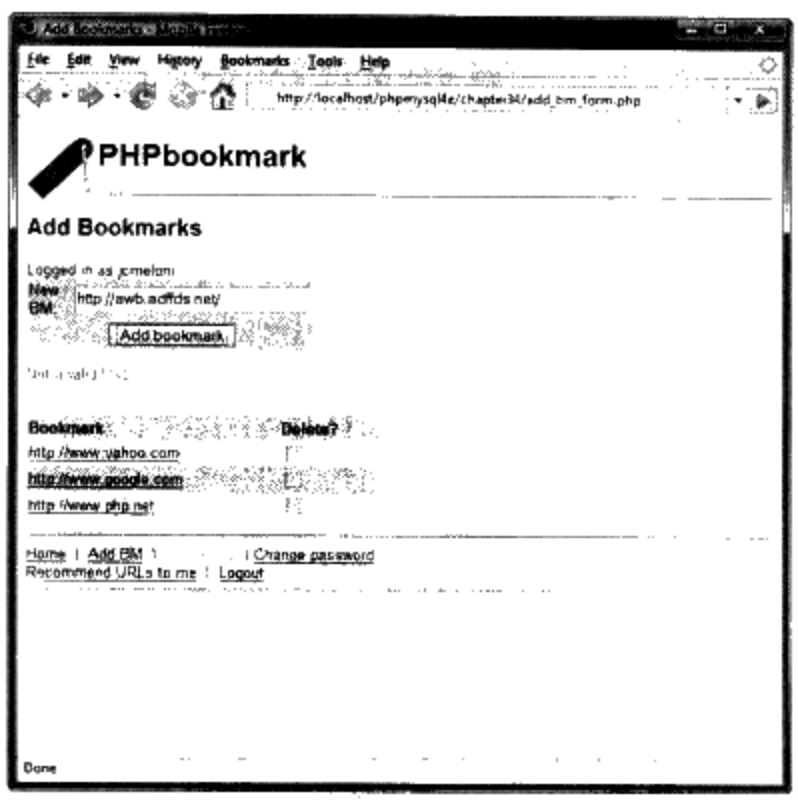

图34-7 尝试添加一个无效URL

最后，无论添加URL至用户已有书签列表是否存在错误，都将显示用户的已有书签列表。如图34-8所示的是该结果。

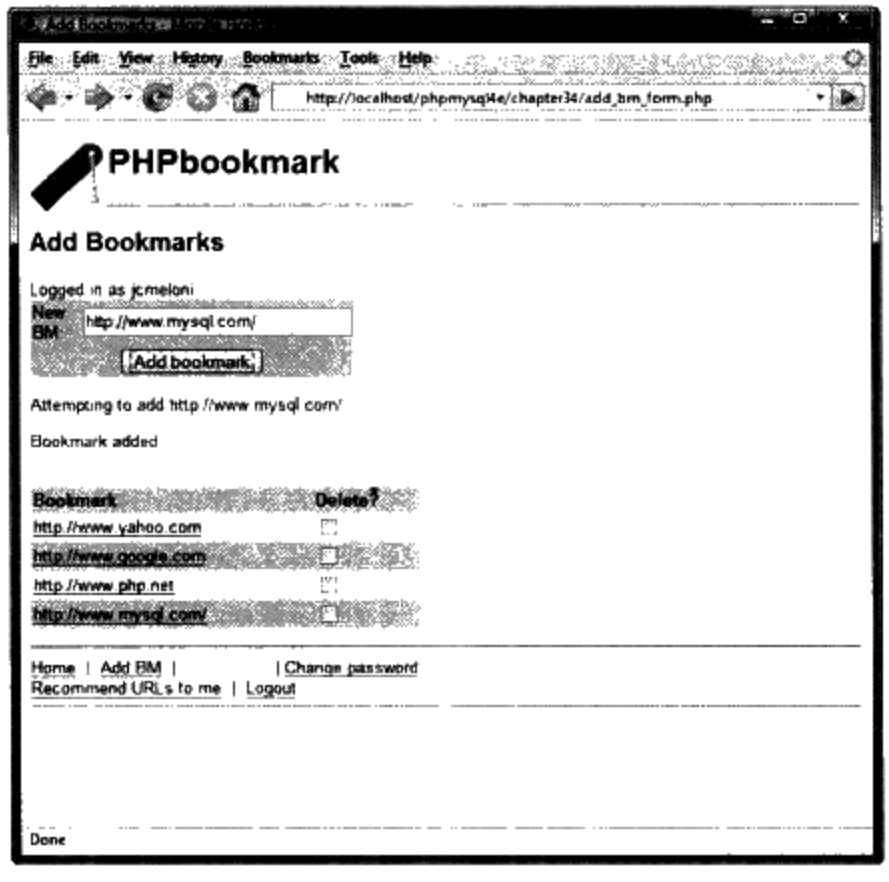

图34-8　成功——添加了一个有效的URL

尽管添加书签这个核心功能已经成功地支持Ajax，还需要对一些元素进行修改。例如，在新系统中，ulr_fns.php脚本的add_bm()函数包含的一些异常处理可能会导致错误信息。程序清单34-10所示的是已有的add_bm()函数。

程序清单34-10　ulr_fns.php脚本中已有的add_bm()函数代码

```
function add_bm($new_url) {
  // Add new bookmark to the database

  echo "Attempting to add ".htmlspecialchars($new_url)."<br />";
  $valid_user = $_SESSION['valid_user'];

  $conn = db_connect();

  // check not a repeat bookmark
  $result = $conn->query("select * from bookmark
                          where username='$valid_user'
                          and bm_URL='".$new_url."'");
  if ($result && ($result->num_rows>0)) {
    throw new Exception('Bookmark already exists.');
  }

  // insert the new bookmark
  if (!$conn->query("insert into bookmark values
     ('".$valid_user."', '".$new_url."')")) {
    throw new Exception('Bookmark could not be inserted.');
```

```
    }

    return true;
}
```

在这里，我们希望能够将异常转换成错误信息，并且继续代码的执行（显示）。这可以通过修改两个if语句块实现：

```
if ($result && ($result->num_rows>0)) {
 echo "<p class=\"warn\">Bookmark already exists.</p>";
} else {
 //attempt to add
 if (!$conn->query("insert into bookmark values
   ('".$valid_user."', '".$new_url."')")) {
   echo "<p class=\'warn\">Bookmark could not be inserted.</p>";
 } else {
   echo "<p>Bookmark added.</p>";
 }
}
```

以上代码仍然延续了可能事件的逻辑顺序，但是能够显示适当的错误信息。在检查要添加的URL是否已经存在于用户已有书签列表后，或者在书签添加表单和用户已有书签列表之间给出一个错误信息，或者脚本尝试添加这个URL。

如果不能添加这个书签，在书签添加表单和用户已有书签列表之间还将显示一个错误信息。但是，如果成功添加该URL后，将显示“Bookmark Added（已添加书签）”信息。从add_bm.php脚本删除的echo语句被添加到add_bm()函数中，否则用户还将看到一个错误信息，例如，在“Bookmark could not be inserted（不能插入书签）”消息之后还将显示“Bookmark Added（已添加书签）”的成功消息，而事实的输出确实是不正确的。

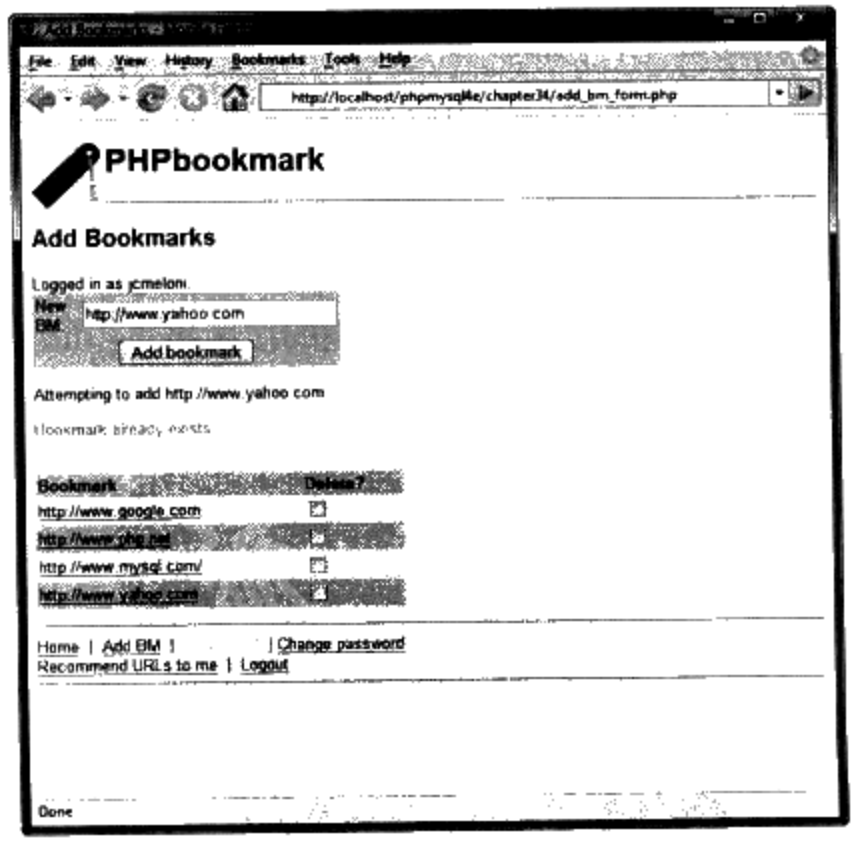

图34-9 尝试添加一个已存在的书签

如图34-9所示的就是这些修改的结果。

4. 对PHPBookmark应用的额外修改

将书签添加功能修改为支持Ajax的用户界面只是要对这个应用进行的许多修改之一。接下来的选择可能就是书签删除功能。书签删除的操作如下所示：

- 从页面脚注删除“Delete BM（删除书签）”链接。
- 当用户选中某个书签左边的“Delete?（是否删除）”复选框时，调用一个JavaScript函数。

- 修改delete_bm.php脚本，这样新的JavaScript函数就可以调用这个脚本完成删除操作并向用户返回信息。
- 对功能进行必要的修改，确保操作能够合理完成，并且在新的用户界面显示正确的信息。

有了修改的结构，就可以根据本章所介绍的内容实现这些修改。接下来的内容将提供一些资源链接，这些链接包含了关于创建支持Ajax站点的更多更详细信息。

请记住，Ajax是一系列技术的组合，用于创建更流畅的用户体验；在了解了Ajax这些组成部分及其功能后，这也让重新思考一个应用的架构成为必要。

34.4 进一步学习

本章关于创建支持Ajax的应用所涉及的内容只是非常肤浅的。有一本新书《Sams Teach Yourself Ajax, JavaScript and PHP All-in-One》详尽的讨论了本章所涉及的所有内容，它可以作为学习了本章内容后进一步提高的工具。当然，也有许多网站专门或者部分地介绍构建Ajax应用的技术，以及一些可以加速开发进程的第三方代码库，这样就能避免重复工作。

34.4.1 进一步了解文档对象模型（DOM）

尽管本书介绍了使用PHP进行服务器端编程以及使用MySQL作为支撑动态应用的关系数据库，它并没有涵盖任何关于客户端的内容，例如，XHTML、CSS、JavaScript以及文档对象模型（Document Object Modal，DOM）。如果还不熟悉DOM，可能需要丰富这方面的内容，因为要创建一个功能全面的Ajax应用，这方面的内容是必不可少的。

许多（如果不是所有）Ajax应用都会使用JavaScript来操作DOM。无论是使用显示元素，浏览器历史记录或者窗口位置，透彻地理解DOM对象及其属性对于创建一个流畅的用户界面来说是至关重要的，这也是Ajax应用的目标。

如下站点包含了大量学习DOM的信息：

- W3C关于文档对象模型的技术报告：http://www.w3.org/DOM/DOMTR
- DOM脚本编程组织的主页：http://domscripting.webstandards.org/
- Mozilla项目关于DOM的开发人员文档：http://developer.mozilla.org/en/docs/DOM（以及关于JavaScript文档的链接：http://developer.mozilla.org/en/docs/JavaScript）

34.4.2 Ajax应用可用的JavaScript函数库

Ajax应用大概出现在2005年，是Jesse James Garret在他的文章提出了这个名词，因为在向其客户描述这个技术时，“他需要一个更短的名词来表述‘异步的JavaScript+CSS+DOM+XMLHtttpRequest’”。从那以后，有很多第三方开发人员提供了许多可供普通开发人员在创建Ajax应用时使用的JavaScript函数库。

注意 请参阅Garret的文章“Ajax: A New Approach to Web Applications（Ajax：Web应用的新方法）”。

尽管花些时间浏览任何Ajax开发站点可以很快找到更多信息，这里还是给出一些流行的函

数库。选择其中一个都将减少开发时间，正如前面提到的，不需要开发人员进行一些重复工作。

- Prototype JavaScript框架可以简化DOM对象的操作。在创建复杂的Ajax应用时，可以使用XMLHTTPRequest对象。更多信息，请参阅：http://www.prototypejs.org/。
- Dojo是一个开放源代码的工具包，它包含了许多基本的JavaScript函数，一个创建框架以及有效的打包并向终端用户分发代码的机制。更多信息，请参阅：http://dojotoolkit.org/。
- MochiKit是一个轻量级的代码库，它包含了处理DOM以及格式化客户端输出的函数。MochiKit的口号有点“不友好”但是很真实：“MochiKit使得JavaScript不那么糟糕”。MochiKit提供的函数和解决方案，可供开发人员使用的文档以及使用MochiKit创建的示例项目都值得一看。更多信息，请参阅：http://mochikit.com/。

34.4.3 Ajax开发人员网站

最后，学习Ajax开发的最佳途径就是实践。找到一些代码，确定如何集成到已有应用，以及向已经使用该技术的开发人员学习。如下所示的网站将有助于更好的开始Ajax应用的开发：

- Ajaxian是一个开发人员门户网站，它为新的或者有经验的开发人员提供了新闻、文章、教程以及示例代码。更多信息，请参阅：http://ajaxian.com/。
- Ajax Matters包含一些深度介绍Ajax开发的文章。更多信息，请参阅：http://www.ajaxmatters.com/。
- Ajax Lines是另一个开发人员门户网站，它提供了关于Ajax所有内容的新闻和文章。更多信息，请参阅：http://www.ajaxlines.com/。