第 1 章 Ajax 概述

1

因特网技术的发展一直非常迅速。虽然最早出现的时候显示的是文本信息（因为那时存储空间和内存都很有限），但近几年它已迅速发展成为图形化的、功能很强的媒体。随着其发展，就需要有相关的工具来开发、制造和维护。随着迅速发展的技术，开发人员不断拓展所能实现的功能范围，也开始要求开发工具更加可靠。

为了满足这种需求，出现了大量“Web开发人员”能够使用的工具。诸如HTML、PHP、ASP和JavaScript等语言，可以帮助开发人员在因特网上创建和部署自己的作品。每种语言都已经有多年的发展历史，并给现在的Web开发人员提供了强大的工具集。虽然这些工具的功能日益强大，但在因特网应用程序和更根深蒂固的桌面应用程序之间仍然存在几个主要差异。

在可见的差异中，可能最明显的就是页面请求。在Web应用程序中要执行某些操作，就要向服务器端发送调用请求。为此页面必须刷新，以呈现出从服务器端传给客户端（通常是诸如Firefox或IE等Web浏览器）的最新信息。这不是某个浏览器的缺点，而是所有Web浏览器的HTTP请求/响应协议就是按这种方式工作的（见图1-1）。虽然从理论上讲，这种方式可能工作得很好，但是开发人员开始寻求更简易的方法，以使Web应用程序的响应时间更短。

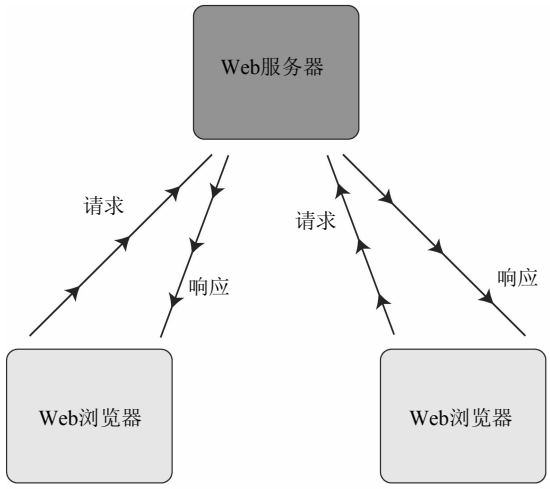

图1-1　因特网当前绝大多数网站所采用的请求/响应方法

1.1　从 CGI 到 Flash 再到 DHTML

开发社区提出需求，商业公司就会给予解决方案。开发工具在设计完成之后，就肯定会有褒贬不一的评价。也许第一种脚本语言就是为了使Web应用设计更加自由，从而避开服务器端CGI（公共网关接口）。

随着CGI的出现，开发人员可以完成一些复杂的操作，如动态图像创建、数据库管理、复杂计算以及动态Web内容创建等。我们现在对Web应用程序的各种认识最早就是源于CGI。不幸的是，CGI虽然解决了很多问题，但其交互和响应的无缝结合仍是个难题。

为了创建真实生动的Web内容，Macromedia公司[1]发布了一个功能强大的Flash套件，在那时这一举动可以称得上是“石破天惊”。直到今天Flash仍然实至名归。它使Web开发人员能够构建出可视化的、令人印象深刻的“影片”，并作为网站、应用程序的一部分功能。这些网站比其他网站更“炫”，因为可以在各种浏览器上实现“运动”式的效果。

在专业设计师手下，嵌入Flash的Web网站在视觉效果方面会给人留下深刻印象。同样，在专业开发人员手中，它们可以表现出强大功能。不过在单个处理过程中同时融合视觉设计和开发技能是很少见的；因此Flash应用程序分为两类，一类是视觉效果好但功能少，另一类则是功能很强但界面效果有待提高。而且这种两难境地又夹杂了兼容性问题：为了实现Flash功能，浏览器必须安装一个插件。

还有一种可视化动态技术——DHTML（动态超文本标记语言），虽然已经出现多年但用户群并不大。DHTML是JavaScript和HTML的紧密结合体，其本质就是将HTML和CSS元素与JavaScript结合，在Web浏览器上实现动态的效果。虽然DHTML在JavaScript专业开发人员手里能够实现一些令人印象深刻的效果，但因为需要掌握一些专家级技术，因此并未进入主流[2]。

如下拉菜单、图像翻转、弹出的工具提示等脚本都很普通，但只是由少数高手开发，大多数开发人员仅仅是应用和部署而已。很少有人自己从头开发这些软件包，因此直到最近也并没有很多人认为JavaScript是一种因特网的强大工具。

1.2　当前 Web 应用程序环境的优缺点

创建基于因特网的Web应用程序有很明显的优缺点。桌面应用程序不断与跨平台兼容问题做斗争，经常要根据完全不同的规则进行编码，而因特网应用程序在不同浏览器中移植则简单得多。加之只有少数主流浏览器拥有巨大的用户基础，因而程序在不同用户之间部署相对比较稳定。

对于在线应用程序而言，还有一个很大的优势是只需要创建和维护一套代码。如果创建的是

1. 已被Adobe公司并购。——编者注
2. DHTML的衰落很大程度上是当时浏览器不兼容造成的。——编者注

桌面应用程序，那么在部署一个修复bug的补丁时，用户要么重新安装整个软件包，要么获取该补丁并安装它。此外，很难判断哪些安装将受到影响。

另一方面，Web应用程序可以只部署在服务器上就能够让所有用户访问。对其功能的任何改变或改进都只需发布到一个中心位置并马上生效。随着开发人员开发能力的增强，他们倾向于开发和维护更高级的产品。

不过任何事情都是有代价的。从维护的角度来看，在中心服务器上部署应用程序是很好的选择，但也将引发新的问题：客户端需要一种访问这个中心服务器的途径。因特网为此提供了一个良好的选择，但速度马上成了新的问题。

例如在使用Word时，单击一个按钮会立即启动某个操作并马上获得响应，而基于因特网的应用程序则需要首先与它连接。虽然高速因特网越来越普及，但很多用户仍然在使用56 Kbps（甚至更低）的调制解调器。因此即使在服务器上能够快速处理信息，发送给最终用户还是需要很长时间。

与该问题相伴的是，对于每个服务器的响应还需要刷新页面，这样就给最终用户带来了很大的挫折感。因此Web应用程序还需要获得桌面应用程序那样的传输速度。如前所述，Flash提供了这样一种手段，并且还提供了强大的ActionScript语言来扩展它，但这需要一个专业人员才能有效地应用它。DHTML通过使用JavaScript也提供了这样的手段，但其代码是很受限制的。

更糟糕的是还经常需要处理浏览器与标准不兼容的问题（甚至完全不遵循标准）。幸运的是，对于这些问题已经有了解决方法——Ajax。这是Jesse James Garrett给“异步JavaScript和XML”所起的名字，它因Google的Gmail等Web应用程序而广泛流行。Ajax基于更流畅的页面载入向服务器端发送请求，并且很少需要刷新整个页面。

1.2.1 走近 Ajax

Ajax在因特网世界引起了大家的重视，不仅仅在于其易于使用且功能强大，还在于它能够吸引几乎所有开发人员的注意。提出Ajax的这两年来，没有出现任何形式的标准（而且彻底基于Ajax来构建其核心的网站还不算太多），但现在Ajax看起来已经像图像翻转一样平常了。

一些一鸣惊人的Web应用程序完全基于Ajax功能。它们不仅巧妙地应用了这一技术，还将Web产业带入一个新的时代，使得标准的Web浏览器能够具备很强的功能，甚至能与桌面应用程序媲美。

例如Flickr（www.flickr.com）或Gmail（www.gmail.com）（参见图1-2），从表面上看它们所提供的服务并没有什么新颖之处（在因特网上有多少网络相册系统和Web邮件服务呀！）。为什么这两个应用程序能够获得如此多（特别是在网络社区中）的关注呢？

我认为这些基于Ajax的新应用程序之所以流行，不是因为它提供了什么新的、令人惊讶的功

能，而是它们以一种有效的、人性化的形式（直到现在，这些在因特网应用程序中还是比较匮乏的）将信息和功能展现给我们。

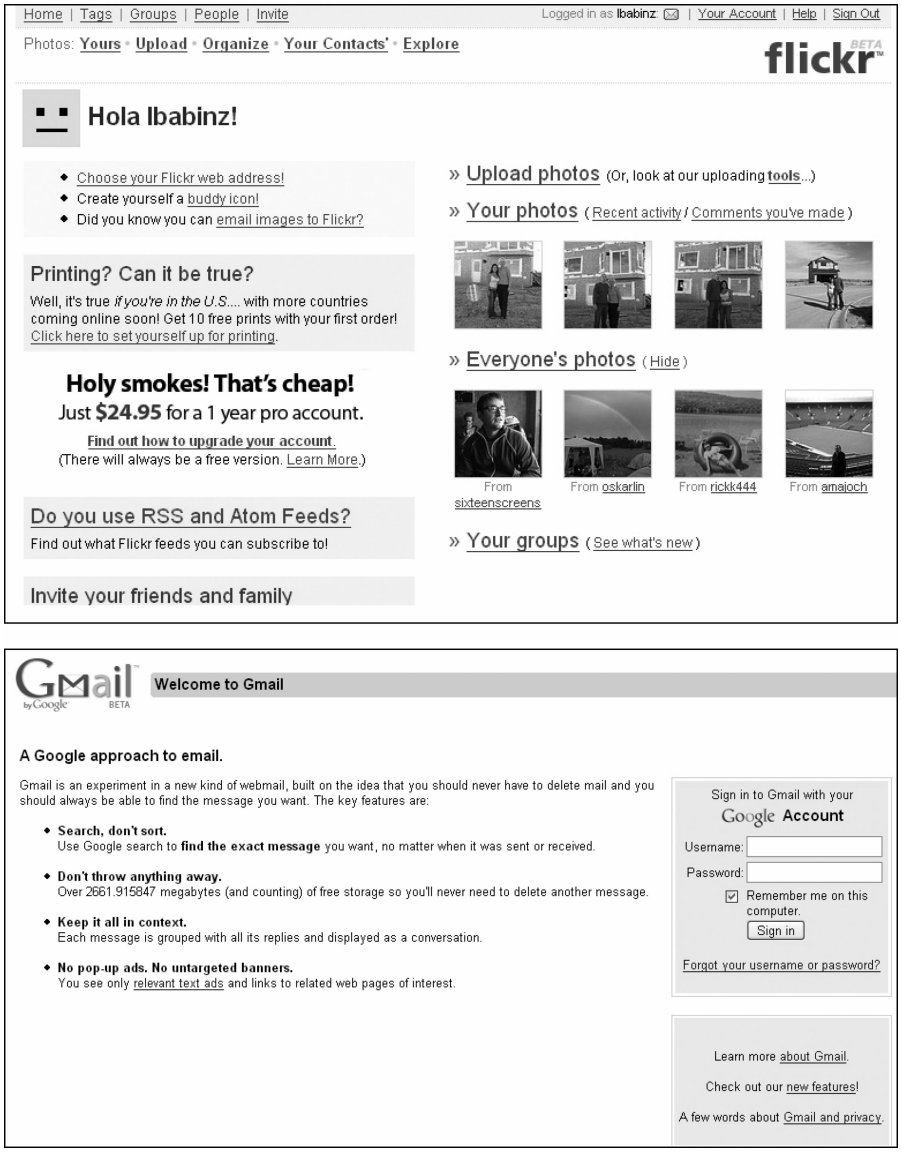

图1-2　诸如Flickr和Gmail之类的网站都是功能强大的Ajax应用

1. Ajax定义

Ajax表示“异步JavaScript和XML”，尽管不是所有人都认为Ajax是一个合适的术语，但那些

批评这个术语的人也不得不承认该技术之所以被广泛接受，这个新名词起到了一定的作用。

本质上，Ajax使用基于JavaScript的XMLHttpRequest对象，向Web服务器发送异步请求或者避免对页面进行刷新（图1-3和图1-4阐释了传统的和基于Ajax的请求/响应模型的区别）。通过使用XMLHttpRequest，Web应用程序能够向服务器获取或发送信息，由服务器完成所需的处理，然后动态地改变Web页面的某个部分，用户甚至不用转到其他页面或将焦点转移到其他地方。你可能认为通过XMLHttpReqeust对象返回的数据格式都是XML，它当然能够返回XML，而且还能够返回在脚本语言中指定的任何数据格式。

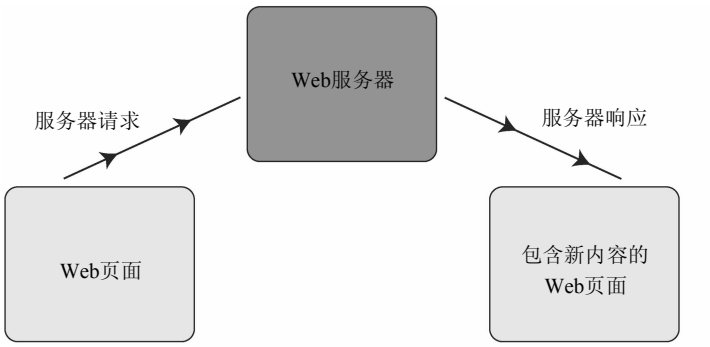

图1-3 现在绝大多数基于Web的应用程序都采用传统的服务器请求/响应模型；每次服务器请求发出后，该页面将刷新以显示新内容

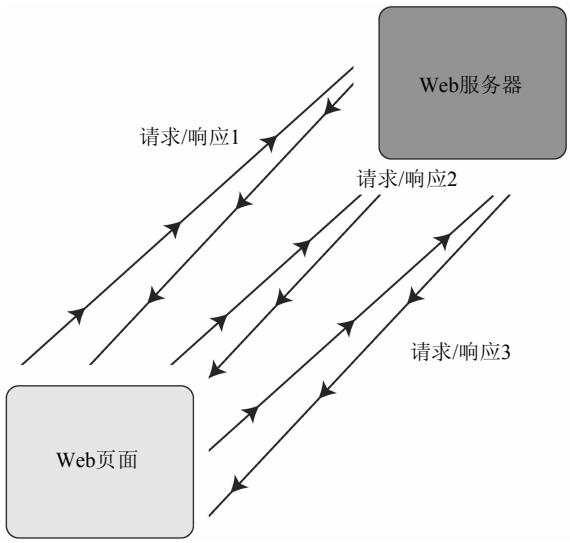

图1-4 使用Ajax异步方法的因特网请求/响应模型；页面可以发送多个服务器请求且无需刷新页面

看一个实例，使用贷款计算器表单来计算将会从银行账户中扣走的钱数，这对于脚本语言非

常简单。通常处理这种应用程序的方法是，填写表单内容，单击提交按钮，然后等待服务器响应。而且你可以重新再来一遍，以尝试新的财务公式。

然而如果使用了基于JavaScript的Ajax解决方案，用户单击提交按钮后仍然会停留在相同的页面中，服务器将完成计算并显示贷款值。然后可以直接修改表单中的值，并能马上看到不同的结果。

接着，修改功能具备了。甚至不用提交就可以通过Ajax在用户每次填完字段后自动向服务器发送一个调用，并马上更新结果。像这样以人为本的友好功能已经成为了主流。

2. Ajax是新技术吗

在资深的Web开发人员面前说Ajax是新技术一定会遭到大声驳斥。Ajax不是新技术，事实上Ajax根本不算一项技术。Ajax描述的只是使用基于JavaScript的`XMLHttpRequest`对象以动态（异步）的形式从服务器获取信息的过程。

使用`XMLHttpRequest`的方法早在1998就开始流行了，如IE 4这样的浏览器就已经拥有实现Ajax的能力（尽管存在一些配置问题）。在当前浏览器开发出来之前很久，从客户端使用JavaScript来完成即时服务器端请求的处理方法就开始使用了。

不过如果从概念的角度（而非技术）来讨论Ajax的广泛使用，那么它对因特网社区绝对是一种新的启示。Web开发人员最终都会认识到，并不是所有向服务器发送的请求都要以相同的方式来处理。在某些方面，Ajax开放了被传统模式约束的上百万Web开发人员的思路，让他们看到冲破这些禁锢的可能。不过，请不要把我看作先锋，我只是这百万开发人员中的一员。

3. 现在Ajax为什么这么流行

该技术已经存在了这么久，为什么直到现在才开始流行呢？它是从哪里开始流行起来的，是谁推动了这个流行趋势，这都很难说清楚。也许是Gmail的应用广泛引发了许多开发人员的讨论，也许是Jesse James Garrett创造了这个术语使得这个概念有了统一的称呼；但我认为Ajax的真正意义，对于开发人员的要比对于使用者的更大。

我们来分析一下诸如会计这样的行业。多年来会计师都在使用纸制数据表及过时的数学公式去组织高度复杂的财务信息。随着计算机的出现，这一切发生了改变，出现了提供此类服务的新方法。的确，源于原始方法的标准直到现在仍然是正确的，不过其内容已经大大丰富了，而且运行业务的新方法也已经出现。

开发者们应用Ajax创建的因特网软件与此类似。传统的约定在有些场合仍然有效，不过现在有了一种实现功能和表现信息的新方法。它是一种新工具，可以应用并促进我们的业务。现在正在开发的一些新方法，直到近段时间才进入开发人员的视野。我对使用Ajax概念来构建应用程序感到十分兴奋，迫不及待想知道有什么创新的因特网机制可以使用。

1.2.2 Ajax 的必要条件

因为Ajax是基于JavaScript技术的，因此用户的浏览器必须启用JavaScript才能够正常工作。也就是说，大多数用户可以选择浏览器是否使用JavaScript，这倒不会引起什么安全性问题。但必须注意，用户具有“禁用”Ajax的能力。因此当开发一个Ajax应用程序时，要确保网站中还有实现同样功能的其他方法，或者给用户一个提示告诉他运行该应用程序所需的环境，这点很重要。

Ajax已获得了许多浏览器的支持，其中包括Firefox（所有可用版本）、IE（4.0及更高版本）、Apple Safari（1.2及更高版本）、Konqueror、Netscape（7.1及更高版本）和Opera（7.6及更高版本）。因此绝大多数广泛应用的浏览器都提供了处理Ajax及其相应技术的手段。在第11章中将介绍更多与处理跨浏览器Ajax相关的问题。

要使Ajax的应用更加有效，唯一的要求就是打破多年应用的标准，然后创建一些真正革命性的、具有良好功能的东西。

1.3 小结

现在你应该对Ajax这个技术新宠的来源及其未来的发展方向有了比较清楚的认识。对于那些还没有动手试过Ajax的Web开发人员来说，读到这里时会很想知道它能够实现什么。我们首先引入了一个无需刷新页面的服务器请求执行方式的概念，只用了很短的篇幅阐述了所有现在可以实现的令人惊讶的构想。还讲述了该技术将打破所有传统约束。

做好了解更多知识的准备了吗？让我们进入新的一章，学习如何在工作中应用Ajax和PHP。

第 2 章 Ajax 基础

对于Ajax有一个误解：能够实现这么酷的功能，JavaScript代码肯定是很难实现和维护的。刚开始尝试这一技术时的确并不简单。基于Ajax的服务器请求结构非常易于理解和调用。只需简单地创建一个`XMLHttpRequest`类型的对象，确定其是否创建成功、将放在哪里以及结果将在何处显示，然后发送它。这就是全部内容。

如果就只有这些，那么有什么可大惊小怪的？那是因为Ajax的关键不在代码，而在功能、友好性和用户界面方面。实际上从开发人员视角来看，实现Ajax其实并不难，它更多的属于锦上添花。这样，开发人员便无需过多操心代码是如何工作的，而是将精力集中在何时应用这个概念的构思上就行了。

虽然Ajax也能够用于很简单的目的，诸如载入HTML页面、执行表单验证之类的普通任务，但其威力只有在与强大的服务器端脚本语言协作时才发挥出来。本书中讨论的服务器端脚本语言是PHP。当把Ajax客户端交互概念与PHP服务器端引擎相结合时，就可以开发出令人惊叹的应用程序。这两者结合在一起的空间是有限的，而本书将展示如何通过它们的结合得到非常强大的功能。

要用Ajax和PHP来创建Web应用程序，首先必须对基础知识有深入的理解。注意，Ajax是一种JavaScript工具，因此当尝试理解Ajax类型的应用程序时，对JavaScript基础知识的掌握就尤为重要。让我们从基础知识开始。

2.1 HTTP 请求和响应基础

要深入理解Ajax概念的整体构思，那么理解网站是如何处理请求以及如何从Web服务器接收响应就很重要。现在浏览器从Web服务器获取信息所采用的标准是HTTP（当前版本是HTTP/1.1）。这是Web浏览器用来向网站发送请求，以及从Web服务器（负责返回响应）接收响应的方法。

HTTP请求的工作机制与电子邮件类似。也就是说，发送请求的同时传输一些报头（header），以便让Web服务器知道要提供什么服务以及如何处理该请求。虽然大部分报头都是可选的，但有一个是必需的（如果你要访问的不仅仅是服务器的默认页）——`host`。这个报头至关重要，用来告诉服务器要提供什么服务。

一旦服务器接收到一个请求，就决定返回什么响应。表2-1中列出了常用的响应代码。

表2-1 常用的HTTP响应代码

响应代码	描 述
200 OK	表示请求的文档或文件已找到，并正确返回
304 Not Modified	浏览器指出它拥有一个本地的缓存副本，而服务器端的内容与此相同
401 Unauthorized	请求所需文档需要验证
403 Forbidden	请求者对所请求的文档不具有相应的权限
404 Not Found	请求的文件找不到（例如不存在）
500 Internal Server Error	服务器在处理请求时遇到了问题
503 Service Unavailable	服务器因负载过大而无法处理该请求

另外，表单还存在多种请求方法。其中像GET和POST这样的方法，大家可能都很熟悉了。表2-2中列出了所有可能的请求方法（尽管通常只使用GET和POST方法）。

表2-2 HTTP请求方法

方 法	描 述
GET	发送请求的最常用手段；对服务器特定资源的简单请求
HEAD	类似于GET请求，只不过返回的响应中没有具体内容，用于获取报头
POST	用来发送包含用户提交数据的请求（对于基于Web的表单最合适）
PUT	传送当前请求文档的一个版本
DELETE	发送一个用来删除指定文档的请求
TRACE	发送请求的一个副本，以跟踪其处理进程
OPTIONS	返回所有可用的方法；可检查服务器支持哪些方法
CONNECT	用于SSL隧道的基于代理的请求

至此，你已经对如何从浏览器向服务器发送请求以及如何返回响应有了基本的认识，再来看看XMLHttpRequest对象的工作原理就简单多了。它实际上相当简单，只是在后台完成操作，无需刷新页面。

2.2 XMLHttpRequest 对象

Ajax只是一个概念，用来描述客户端XMLHttpRequest对象与服务器端脚本的交互。要通过Ajax向服务器发送请求，必须创建一个能够用于不同表单的对象。注意，在各种浏览器中XMLHttpRequest对象的实例化和处理方法是不同的。微软公司的IE是以ActiveX控件的形式创建该对象，而Firefox及Safari等浏览器则是使用基本的JavaScript对象。这对于希望在任何浏览器中都能执行Ajax功能的跨浏览器代码是至关重要的。

2.2.1 XMLHttpRequest 方法

XMLHttpRequest对象的实例创建之后，用户就能够使用它的一系列方法，如表2-3所示。根据使用对象的方式不同，重要的方法也不同。

表2-3　XMLHttpRequest对象的方法

方　法	描　述
abort()	取消当前的请求
getAllResponseHeaders()	以String类型的变量返回所有HTTP报头
getResponseHeader()	返回方法中指定的HTTP报头的值
open()	指定与服务器连接所需的各种属性，可以选择使用GET或POST等方法，确定连接模式是否为异步，以及指定要连接的URL
setRequestHeader()	发送请求时，在报头中添加一个标签/值对
send()	发送当前请求

虽然表2-3中列出的方法可能有些复杂，但并非如此，下面将更详细地分析。

1. abort()

abort方法非常简单，就是停止当前正发送的请求。通过该函数就能够很简单地控制连接的时间长度。如果希望明确地规定请求发送的时间，可以通过abort方法来提前结束请求。

2. getAllResponseHeaders()

可以通过该方法来获取传送的所有HTTP报头。以下就是一个报头集的示例：

```
Date: Sun, 13 Nov 2005 22:53:06 GMT
Server: Apache/2.0.53 (Win32) PHP/5.0.3
X-Powered-By: PHP/5.0.3
Content-Length: 527
Keep-Alive: timeout=15, max=98
Connection: Keep-Alive
Content-Type: text/html
```

3. getResponseHeader("headername")

可以通过该方法来获取特定报头的内容。这对于要查询部分报头信息时很有用，因为所有报头信息将是一个很大的字符串。例如，要查询所请求的文档大小，只需调用getResponseHeader ("Content-Length")。

4. open ("method","URL","async","username","pswd")

现在才接触到XMLHttpRequest对象的最基本的部分。我们将使用该方法来创建与服务器上特定文件的连接。在这里可以设置打开一个文件的方法（GET还是POST），同时还将定义该文件将如何打开。注意，函数中每个参数并非都必须提供，可以根据实际情况来定。

5. setRequestHeader("label","value")

通过该方法可以将label参数指定的报头的值设置为value的值。有一点需要强调，该方法的调用只能够在open()方法调用之后且send函数调用之前。

6. send("content")

这个方法完成向服务器发送请求。如果该请求是异步发送的，那么立刻返回响应；如果不是异步模式则要等到响应收到后才返回。也可以指定一个输出字符串作为参数，它对于表单的处理是很有帮助的，它能够用来传递表单元素的值。

2.2.2 XMLHttpRequest 属性

任何对象都有完整的属性集，用来实现所需的功能。表2-4列出了XMLHttpRequest对象的所有属性。解释这些属性是很必要的，因为使用该对象的高级功能时会用到这些属性。

表2-4 XMLHttpRequest对象的属性

属 性	描 述
onreadystatechange	用作根据状态变化而触发的事件处理
readyState	该对象的当前状态（0：未初始化，1：载入中，2：已载入，3：交互中，4：完成）
responseText	以字符串格式返回响应
responseXML	以XML格式返回响应
status	以数字格式返回请求的状态（将返回标准的页面错误，如表示未找到文件的404）
statusText	以字符串格式返回请求的状态（例如404错误将返回字符串Not Found）

1. onreadystatechange

onreadystatechange属性是一个事件处理，当状态（表示处理过程处于什么特定的阶段）改变时触发特定的代码段或函数。例如，有些函数是在表单初始化时调用的，还有一些功能是在状态改变成complete时启动的。

2. readyState

readyState属性详细描述了当前请求处于过程的哪个阶段。这个属性对于异常处理是十分有用的，并且在决定何时执行特定操作时也很重要。可以使用该属性来创建基于请求处理进程的操作。例如，可以使一段代码在readyState为载入中时开始执行，直到值为完成时结束。

3. responseText

当请求执行后，就将返回responseText属性。如果是向某种脚本发送一个请求，那么该脚本的输出就将通过该属性返回。通常会将该属性的值复制给某个元素的innerHTML属性，从而异步地在页面元素中载入一个脚本或文档。

4. responseXML

工作机制与responseText类似，不过返回的响应是XML格式，如果打算使用浏览器内建的XML处理功能，它是个理想的选择。

5. status

该属性指明了请求返回的响应代码（常见的响应代码见表2-1）。例如，如果请求的文件找不到，则该属性就会设置为404以表示文件未找到。

6. statusText

该属性是status属性的文本表述。如果status属性的值是404，那么statusText的值就将是Not Found。同时使用status和statusText属性，能够让用户更深入地了解发生了什么。毕竟并不是所有用户都知道数字404表示的是什么。

2.2.3　跨浏览器用法

尽管目前IE仍占据着霸主地位，但Firefox等竞争者也取得了很大发展。因此，确保Ajax应用程序保持跨浏览器兼容是很重要的。Ajax功能最为重要的一个方面是在不同浏览器中都能顺利部署，只需做少量的工作就能让其功能在绝大多数浏览器（老版本除外）中应用。下面的代码创建了一个XMLHttpRequest对象实例，它能够在所有支持XMLHttpRequest的浏览器中运行。图2-1展现了在IE和非IE浏览器中输出的区别。

```
//创建一个布尔型变量，用来检查是否为合法的IE实例。
var xmlhttp = false;

//检查是否使用的是IE。
try {
  //如果JavaScript的版本大于5。
  xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  alert ("You are using Microsoft Internet Explorer.");
} catch (e) {

  //如果不是，则使用老版本的ActiveX对象。
  try {
    //如果使用的是IE浏览器。
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    alert ("You are using Microsoft Internet Explorer");
  } catch (E) {
    //否则肯定使用的是非IE浏览器。
    xmlhttp = false;
  }
}
```

```
//如果使用的是非IE浏览器，则创建一个该对象的JavaScript实例。
if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
  xmlhttp = new XMLHttpRequest();
  alert ("You are not using Microsoft Internet Explorer");
}
```

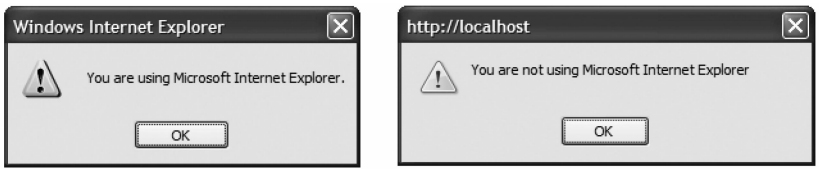

图2-1 该脚本使你知道当前是在哪种浏览器上执行Ajax请求

如你所见，XMLHttpRequest对象的创建过程可能不同，不过最终的结果是一样的。在这方面微软的IE要比绝大多数其他浏览器更复杂些，你还必须检查用户当前使用的IE是什么版本（然后还要检查使用的是什么版本的JavaScript）。这个代码示例的流程是相当简单的。主要是检查用户是否使用的是IE的新版本（通过尝试创建ActiveX对象来检查）；如果不是则尝试创建默认的老版本ActiveX对象。如果也无法创建，则说明用户肯定使用的是非IE的浏览器，那么就创建标准的XMLHttpRequest对象作为JavaScript对象。

另外，这里所展示的方法并不是创建XMLHttpRequest对象的唯一方法。下述代码片断的功能大致相同，但更简单：

```
var xmlhttp;

//如果ActiveX对象可用，则使用的肯定是IE。
if (window.ActiveXObject){
  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
} else {
  //否则将使用JavaScript处理方法。
  xmlhttp = new XMLHttpRequest();
}
```

这种调用XMLHttpRequest对象的方法代码更少。不幸的是，它虽然能够完成任务，但我感觉它并不彻底，如果打算使用一些面向对象技术，那么最好还是使用前一个示例。在使用Ajax时需要尽量考虑到各种可能。

2.2.4 向服务器发送请求

既然已经准备好了新XMLHttpRequest对象，接下来自然就是用其向服务器提交请求。这可以通过许多方法实现，不过最关键的是必须检查响应是否正确，并决定使用GET方法还是POST方法。如果使用Ajax从服务器查询数据，那么GET方法更适合。如果要向服务器发送信息，则POST更胜任。本书后续部分将更深入地探讨这个内容，而现在只要知道GET不适用于信息的发送，主

要是因为其传输容量的内在限制。

要向服务器发送请求，必须先确认几个与功能有关的基本问题。首先，必须决定要连接哪个页面（或脚本），然后是在哪里载入这个页面或脚本。请看下面的函数，它的参数包括将载入的页面（或脚本）以及用来载入该页面的div元素（或其他对象）。

```
function makerequest(serverPage, objID) {

  var obj = document.getElementById(objID);
  xmlhttp.open("GET", serverPage);
  xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
     obj.innerHTML = xmlhttp.responseText;
    }
  }
  xmlhttp.send(null);
}
```

该代码主要的参数是HTML元素ID和服务器页面。它将尝试用XMLHttpRequest对象的open()方法建立一个到该服务器页面的连接。如果readyState属性返回的代码是4，而且status属性返回的是200（OK），那么将把请求的页面（或脚本）返回的响应载入到传递对象的innerHTML属性中。

这里主要实现的是创建一个新的XMLHttpRequest对象，然后通过它向脚本或页面发送一个请求，并将返回结果载入到页面的相应元素中。现在可以思考一下关于使用这个极其简单的概念的新颖奇妙的方法。

2.2.5　一个简单的 Ajax 示例

随着Ajax的应用日益广泛，最常见的用途就是导航。通过Ajax方法可以在页面中动态载入内容。由于Ajax能够在你要求的地方载入内容，而不用刷新整个页面，因此重点在于准确地指出要载入内容的地方。

我们或许已经习惯于单击某个链接时页面从头开始载入的过程，并且可能已经习惯了此类概念下的某些功能。而使用Ajax，当页面已经滚动到下一屏并由Ajax动态载入新内容时，不会又从页面顶部开始显示。页面的位置不会变化，而且内容载入也将悄悄地完成。

对于Ajax而言有一个共性的问题，那就是用户可能不知道页面已经发生变化了。因此，如果用Ajax来实现导航工具，并非页面上的所有内容都会改变，这很重要。在我的经验中，页面通常会把导航菜单放在屏幕的顶端（而不会放在底部、中间或边上），然后下面是载入的内容，这样看起来最好，而且对于用户而言也更显眼。

下面的例子展示了一个普通的Web页面，它将通过Ajax载入内容，根据单击的链接显示不同的内容。

```
<!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">
<head>
<title>Sample 2_1</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<script type="text/javascript">
<!--
//创建一个布尔型变量，用来检查是否为有效的IE实例。
var xmlhttp = false;

//检查是否使用的是IE。
try {
  //如果JavaScript的版本大于5。
  xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  alert ("You are using Microsoft Internet Explorer.");
} catch (e) {
  //如果不是，则使用老版本的Active X对象。
  try {
    //如果使用的是IE浏览器。
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    alert ("You are using Microsoft Internet Explorer");
  } catch (E) {
    //否则肯定使用的是非IE浏览器。
    xmlhttp = false;
  }
}

//如果使用的是非IE浏览器，则创建该对象的JavaScript实例。
if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
  xmlhttp = new XMLHttpRequest();
  alert ("You are not using Microsoft Internet Explorer");
}

function makerequest(serverPage, objID) {

  var obj = document.getElementById(objID);
  xmlhttp.open("GET", serverPage);
  xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
    obj.innerHTML = xmlhttp.responseText;
    }
```

```
    }
    xmlhttp.send(null);
}

//-->
</script>
<body onload="makerequest ('content1.html','hw')">
  <div align="center">

  <h1>My Webpage</h1>
  <a href="content1.html" onclick="makerequest('content1.html','hw'); ➥
return false;">Page 1</a> | <a href="content2.html"➥
onclick="makerequest('content2.html','hw'); ➥
return false;">Page 2</a> | <a href="content3.html" onclick=➥
"makerequest('content3.html','hw'); return false;">Page 3</a> | ➥
<a href="content4.html" onclick="makerequest('content4.html','hw'); return false;">
➥
Page 4</a>
    <div id="hw"></div>
  </div>
</body>
</html>

<!-- content1.html -->
<div style="width: 770px; text-align: left;">
  <h1>Page 1</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod➥
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, ➥
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
➥
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu ➥
fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in➥
culpa qui officia deserunt mollit anim id est laborum.</p>
</div>

<!-- content2.html -->
<div style="width: 770px; text-align: left;">
  <h1>Page 2</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod ➥
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, ➥
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
➥
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu ➥
fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in ➥
culpa qui officia deserunt mollit anim id est laborum.</p>
```

```
</div>

<!-- content3.html -->
<div style="width: 770px; text-align: left;">
  <h1>Page 3</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod➥
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,➥
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
➥
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu➥
fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in➥
culpa qui officia deserunt mollit anim id est laborum.</p>
</div>

<!-- content4.html -->
<div style="width: 770px; text-align: left;">
  <h1>Page 4</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod ➥
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, ➥
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
➥
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu ➥
fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in ➥
culpa qui officia deserunt mollit anim id est laborum.</p>
</div>
```

如图2-2所示，只需几分钟就能够创建一个功能完善的Ajax导航驱动的网站。处理这些链接所需的JavaScript代码都放在<script>标签中，当需要向Web服务器发送一个服务器端请求而且不刷新页面时，就可以使用makerequest()函数。可以在任何事件（这里使用的是onclick()事件）中调用makerequest()函数，它将把返回的内容载入到传给该函数的对象中。

图2-2 一个基于Ajax的应用程序。注意，地址栏能够显示出在导航时是否刷新了页面

使用这种方法来完成导航功能，是一种能实现内容与设计分离的好方法，同时也是创建一个快速载入网站的好方法。因为设计封装器只需要创建一次（而内容可以即时载入），用户在访问这样的网站时遇到的延迟更少，页面更流畅。即使这些用户没有快速的因特网连接，在使用传统链接方法时需要等待网站载入，但用了Ajax就不再需要等待了。使用这种Ajax方法可以载入从服务器获取的内容，而不会对用户正在阅读的Web页面产生任何影响。

2.3　小结

总而言之，当需要获取或维护外部脚本及内部内容时，Ajax能够实时地向服务器发送请求。其配置十分简单，易于维护，可以跨平台进行移植。加上适量的异常处理，就可以确保网站绝大多数用户能够如你所想地体验你的网站或应用程序。

下一章要开始介绍将Ajax概念与健壮的PHP应用程序整合的方法。第3章将结合这两种语言，构建出强大的基于Web的应用程序。

第 3 章

PHP 和 Ajax

3

虽然Ajax的概念中包含一组易于实时创建操作的功能，但如果没有与服务器连接，那么就只能够使用基本的JavaScript。倒不是存在什么问题，只是其真正的威力在于用Ajax概念来连接客户端的JavaScript函数和服务器端PHP语言的处理过程。

本章将通过一些示例展现如何整合PHP和Ajax，以设计出一些对于因特网应用程序而言是新的，但对于桌面应用程序而言已经司空见惯的基本工具。例如，向服务器发送无需刷新页面的请求，如果控制得当就是很强大的工具。在强大的PHP服务器端语言的帮助下，可以创建出一些能容易地移植到任何Web项目中的小应用程序。

3.1 为什么选择 PHP 和 Ajax

那么在各种可用的服务器端语言中（ASP、ASP.NET、ColdFusion，等等），为什么本书只选择PHP语言呢？难道是因为PHP与Ajax技术结合得最好？实际上，上述的任何一种语言都能够很好地应用Ajax，但从功能、代码布局及意识形态上，PHP都与用来控制Ajax的JavaScript语言最类似。

PHP或许是最开放的一种技术，而且这种趋势还在延续。虽然用PHP编写的代码对于Web用户而言是隐藏的，但有大量开发者社区都会共享代码。在因特网中可以寻找到大量的示例，从最基本的到最高深的应有尽有。当把PHP的在线社区与诸如ASP.NET语言的在线社区相比较，就不难看出其中的区别。

JavaScript一直以来也是一种开放的技术，很大程度上是因为它本身对于Web用户而言是不隐藏的。它是一种客户端技术，因此用JavaScript编写的代码总是能够看到。或许正是因为JavaScript所采用的这种处理方式，它也一直拥有一个很开放的社区。通过结合JavaScript社区和PHP社区，总能找到所需的例子。

要想知道PHP和Ajax为什么能够这么好地协作，还应从其功能方面来看。PHP是一种很健壮的、面向对象的语言。JavaScript本身也是一种健壮的语言，也引入了面向对象模型。因此，当你整合这两种成熟的语言时，会感觉它们是最好的，的确可以用它们来实现新奇的功能。

3.2　客户端驱动通信，服务器端完成处理

正如前两章所述，Web页面包含两个方面。一方面是客户端通信，这些功能将在客户端的浏览器中实时发生；另一方面是服务器端处理，这里的脚本通常更复杂，包括数据库交互、复杂的公式、条件语句，等等。

本书使用JavaScript语言来处理客户端交互，并将其无缝地与负责服务器端操作的PHP语言相结合。这两种技术的结合方式是有限的，但只要创造力足够、努力足够，任何梦想都会实现。

3.3　简单示例

为了能更快地实现更为复杂、棘手的功能，我们首先将展示一些基于Ajax思想构建的常用小型Web应用。这些例子你可能已经在许多Web应用程序中见过，因而它们为展示如何使用Ajax提供了一个良好的基础。

这些应用已经非常流行，本章将指导你如何使这些功能符合Ajax概念。并不是每个Ajax应用程序都是好创意，重要的是这些例子为什么能够很好地应用Ajax概念，它们是怎样让用户获得更好的体验。同样的应用如果需要刷新页面将会怎样？没有Ajax是否可能实现这样的功能，应用Ajax能够减少多少工作？

3.3.1　内容缩放

通过单击链接（悬停或按键）来隐藏或显示特定内容，这是一个很吸引人的Ajax类型功能。这类功能可以使你在页面中展现大量内容，而不会产生混乱。通过隐藏可伸缩的菜单链接，就可以在很小的空间中添加大量的信息。

下面的例子使用Ajax根据链接来控制日历的展开与收缩。通过Ajax实现信息的隐藏与展现，用PHP根据当前月份动态生成日历，这样一个能添加到任何应用程序中的可隐藏的日历就实现了，而且对这些应用程序没有任何影响。

首先创建一个嵌入该日历的有效Web页面，如下代码所示：

```
<!-- sample3_1.html -->
<!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">
<head>
<title>Sample 3_1</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<script type="text/javascript" src="functions.js"></script>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

```
<body>
  <div id="createtask" class="formclass"></div>
  <div id="autocompletediv" class="autocomp"></div>
  <div id="taskbox" class="taskboxclass"></div>
  <p><a href="javascript://" onclick="showHideCalendar()">➥
<img id="opencloseimg" src="images/plus.gif" alt="" title="" ➥
style="border: none; width: 9px; height: 9px;" /></a>➥
  <a href="javascript://" onclick="showHideCalendar()">My Calendar</a></p>
    <div id="calendar" style="width: 105px; text-align: left;"></div>
</body>
</html>


  //functions.js

  //创建一个布尔型变量，用来检查是否为有效的IE实例。
  var xmlhttp = false;
//检查是否使用的是IE。
try {
  //如果Javascript的版本大于5。
  xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
  //如果不是，则使用老版本的Active X对象。
  try {
    //如果使用的是IE浏览器。
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  } catch (E) {
    //否则肯定使用的是非IE浏览器。
    xmlhttp = false;
  }
}

//如果使用的是非IE浏览器，则创建一个该对象的JavaScript实例。
if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
  xmlhttp = new XMLHttpRequest();
}

//一个用来确定日历是处于打开还是关闭状态的变量。
var showCalendar = true;

function showHideCalendar() {

  //要载入该页面的位置。
  var objID = "calendar";
```

```
    //将当前图像改为加号或减号。
    if (showCalendar == true){
      //显示日历。
      document.getElementById("opencloseimg").src = "images/mins.gif";
      //载入页面。
      var serverPage = "calendar.php";
      //设置跟踪开、关的变量。
      showCalendar = false;

      var obj = document.getElementById(objID);
      xmlhttp.open("GET", serverPage);
      xmlhttp.onreadystatechange = function() {

        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
          obj.innerHTML = xmlhttp.responseText;
        }
      }
      xmlhttp.send(null);
    } else {
      //隐藏日历。
      document.getElementById("opencloseimg").src = "images/plus.gif";
      showCalendar = true;

      document.getElementById(objID).innerHTML = "";
    }


}
```

这看起来很简单，对吧？你需要注意的是functions.js文件中所包含的JavaScript代码。其中创建了一个名为showHideCalendar的函数，它将基于showCalendar变量的值决定显示还是隐藏日历模块。如果showCalendar变量的值是true，将向服务器发送一个Ajax调用以获取calendar.php脚本。而该脚本产生的结果将显示在calendar页面元素中。你可以根据需要显式地修改将载入的元素。该脚本同时还将改变加减号图像的显示以正确地体现“展开”或“收缩”功能。

当脚本向服务器发送一个调用时，PHP脚本将使用其服务器端功能来创建一个针对当前月份的日历。看看以下代码：

```
<?php

  //calendar.php

  //检查月份值和年份值是否存在。
```

```
  if ((!$_GET['month']) && (!$_GET['year'])) {
    $month = date ("n");
    $year = date ("Y");
  } else {
    $month = $_GET['month'];
    $year = $_GET['year'];
  }

  //计算所查看的月份。
  $timestamp = mktime (0, 0, 0, $month, 1, $year);
  $monthname = date("F", $timestamp);

  //现在基于适当的月份创建表格。
  ?>
  <table style="width: 105px; border-collapse: collapse;" border="1"➥
cellpadding="3" cellspacing="0" bordercolor="#000000">
  <tr style="background: #FFBC37;">
  <td colspan="7" style="text-align: center;" onmouseover=➥
"this.style.background='#FECE6E'" onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;"><?php echo $monthname➥
. " " . $year; ?></span>
  </td>
  </tr>
  <tr style="background: #FFBC37;">
  <td style="text-align: center; width: 15px;" onmouseover=➥
"this.style.background='#FECE6E'" onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;">Su</span>
  </td>
  <td style="text-align: center; width: 15px;" onmouseover=➥
"this.style.background='#FECE6E'" onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;">M</span>
  </td>
  <td style="text-align: center; width: 15px;" onmouseover=➥
"this.style.background='#FECE6E'" onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;">Tu</span>
  </td>
  <td style="text-align: center; width: 15px;" onmouseover=➥
"this.style.background='#FECE6E'" onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;">W</span>
  </td>
  <td style="text-align: center; width: 15px;" onmouseover=➥
"this.style.background='#FECE6E'" onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;">Th</span>
  </td>
  <td style="text-align: center; width: 15px;" onmouseover=➥
```

```
"this.style.background='#FECE6E'" onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;">F</span>
  </td>
  <td style="text-align: center; width: 15px;" onmouseover=➥
"this.style.background='#FECE6E'" onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;">Sa</span>
  </td>
  </tr>
  <?php
    $monthstart = date("w", $timestamp);
    $lastday = date("d", mktime (0, 0, 0, $month + 1, 0, $year));
    $startdate = -$monthstart;

    //计算出所需的行数。
    $numrows = ceil (((date("t",mktime (0, 0, 0, $month + 1, 0, $year))➥
+ $monthstart) / 7));

    //创建适当的行数……
    for ($k = 1; $k <= $numrows; $k++){
      ?><tr><?php
      //使用7列(表示7天)……
      for ($i = 0; $i < 7; $i++){
        $startdate++;
        if (($startdate <= 0) || ($startdate > $lastday)){
          //如果在日历中有空白的格子。
          ?><td style="background: #FFFFFF;"> </td><?php
        } else {
          if ($startdate == date("j") && $month == date("n") &&➥
$year == date("Y")){
            ?><td style="text-align: center; background: #FFBC37;" ➥
onmouseover="this.style.background='#FECE6E'"➥
onmouseout="this.style.background='#FFBC37'">➥
<?php echo date ("j"); ?></td><?php
        } else {
          ?><td style="text-align: center; background: #A2BAFA;" ➥
onmouseover="this.style.background='#CAD7F9'"➥
onmouseout="this.style.background='#A2BAFA'">➥
<?php echo $startdate; ?></td><?php
          }
        }
      }
      ?></tr><?php
    }
    ?>
  </table>
```

这段简单的代码用来生成一个显示当前月份信息的日历。这段代码也可以选择年份和月份，可在超级全局变量$_GET中传递。不过现在只关注当前月份的显示。随着该示例的进行，你会看到如何使用Ajax对该模块进行有效的改善，并创建一些很酷的应用程序。

这段代码本身是很容易理解的。它就是使用PHP中的date函数来确定开始日期和结束日期，然后生成相应的日历。这是本书中首个通过结合PHP与Ajax创建的小应用程序（如图3-1所示）。接下来，你将继续自己的应用程序开发工作。

图3-1 在日历应用程序中进行显示/隐藏操作

3.3.2 自动完成

我所知道的第一个被因特网社区广泛认同的功能是Gmail中的自动完成。其主要功能是，当你在收件人地址上输入某个电子邮件地址时，Gmail会自动搜索你的联系人列表（通过Ajax），并自动弹出一个下拉选择框，显示出所有匹配的电子邮件，然后你可以从中选择一个。其整个代码的集成是无缝的，并提供了一个易于使用的功能。

接下来的例子就将实现该功能，尽管它的功能不如Gmail那么完善。我们主要是通过一个数组来实现对象列表的填充（使用数据库的效率更高，但这超出了本例的范围，我们将在本书后续章节中介绍），然后根据用户的输入来显示它们。用户可以单击其名称来填充该字段（因而称之为自动完成）。

在此将对前一个例子进行扩展，假设用户需要在日历中的某一天中输入一个提醒事项。该系

统可以让用户通过单击某一天来输入提醒的名称和任务。最为理想的情况是，当任务输入后系统就自动将其保存到数据库中。不过现在主要还是关注自动完成功能，实际信息的保存将在后续章节中处理。

看看下面的例子，它通过使用Ajax将自动完成功能与一个弹出式表单整合在一起。请注意style.css和functions.js文件，其内容发生了变化。

```
/* style.css */

body {
  font-family: verdana, arial, helvetica;
  font-size: 11px;
  color: #000000;
}

.formclass {
  position: absolute;
  left: 0px;
  top: 0px;
  visibility: hidden;
  height: 0px;
  width: 0px;
  background: #A2BAFA;
  border-style: solid;
  border-width: 1px;
  border-color: #000000;
}

.autocomp {
  position: absolute;
  left: 0px;
  top: 0px;
  visibility: hidden;
  width: 0px;
}

.taskboxclass {
  position: absolute;
  left: 0px;
  top: 0px;
  visibility: hidden;
  width: 0px;
}
```

```
.calendarover {
  text-align: center;
  background: #CAD7F9;
  width: 15px;
}

.calendaroff {
  text-align: center;
  background: #A2BAFA;
  width: 15px;
}

.calendartodayover {
  text-align: center;
  background: #FECE6E;
  width: 15px;
}

.taskchecker {
  width: 150px;
  background-color: #FFBC37;
  border-style: solid;
  border-color: #000000;
  border-width: 1px;
}

.tcpadding {
  padding: 10px;
}

.calendartodayoff {
  text-align: center;
  background: #FFBC37;
  width: 15px;
}


//functions.js

function createform (e){

  theObject = document.getElementById("createtask");

  theObject.style.visibility = "visible";
  theObject.style.height = "200px";
```

```
  theObject.style.width = "200px";

  var posx = 0;
  var posy = 0;

  posx = e.clientX + document.body.scrollLeft;
  posy = e.clientY + document.body.scrollTop;

  theObject.style.left = posx + "px";
  theObject.style.top = posy + "px";

  //要载入该页面的位置。
  var objID = "createtask";
  var serverPage = "theform.php";
  var obj = document.getElementById(objID);
  xmlhttp.open("GET", serverPage);
  xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      obj.innerHTML = xmlhttp.responseText;
    }
  }
  xmlhttp.send(null);

}

function closetask (){

  theObject = document.getElementById("createtask");

  theObject.style.visibility = "hidden";
  theObject.style.height = "0px";
  theObject.style.width = "0px";

  acObject = document.getElementById("autocompletediv");

  acObject.style.visibility = "hidden";
  acObject.style.height = "0px";
  acObject.style.width = "0px";
}

function findPosX(obj){
  var curleft = 0;
  if (obj.offsetParent){
    while (obj.offsetParent){
      curleft += obj.offsetLeft
```

```
      obj = obj.offsetParent;
    }
  } else if (obj.x){
    curleft += obj.x;
  }
  return curleft;
}

function findPosY(obj){
  var curtop = 0;
  if (obj.offsetParent){
    while (obj.offsetParent){
      curtop += obj.offsetTop
      obj = obj.offsetParent;
    }
  } else if (obj.y){
    curtop += obj.y;
  }
  return curtop;
}

function autocomplete (thevalue, e){

  theObject = document.getElementById("autocompletediv");

  theObject.style.visibility = "visible";
  theObject.style.width = "152px";

  var posx = 0;
  var posy = 0;

  posx = (findPosX (document.getElementById("yourname")) + 1);
  posy = (findPosY (document.getElementById("yourname")) + 23);

  theObject.style.left = posx + "px";
  theObject.style.top = posy + "px";

  var theextrachar = e.which;

  if (theextrachar == undefined){
    theextrachar = e.keyCode;
  }

  //要载入该页面的位置。
  var objID = "autocompletediv";
```

```
  //处理退格键。
  if (theextrachar == 8){
    if (thevalue.length == 1){
      var serverPage = "autocomp.php";
    } else {
      var serverPage = "autocomp.php" + "?sstring=" + ➥
thevalue.substr (0, (thevalue.length -1));
    }
  } else {
    var serverPage = "autocomp.php" + "?sstring=" + ➥
thevalue + String.fromCharCode (theextrachar);
  }

  var obj = document.getElementById(objID);
  xmlhttp.open("GET", serverPage);
  xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      obj.innerHTML = xmlhttp.responseText;
    }
  }
  xmlhttp.send(null);
}

function setvalue (thevalue){
  acObject = document.getElementById("autocompletediv");

  acObject.style.visibility = "hidden";
  acObject.style.height = "0px";
  acObject.style.width = "0px";

  document.getElementById("yourname").value = thevalue;
}
```

现在来看看与前面例子比较有什么变化。首先添加了许多函数。第一个是createform，该函数在鼠标当前位置显示了一个隐藏的div元素，然后用Ajax载入一个名为theform.php的文件。该函数主要是基于JavaScript实现的（通过hidden和visible样式），而Ajax则负责载入该文件。theform.php文件（主要就是一个简单的表单）中的代码如下所示：

```
<!-- theform.php -->
<div style="padding: 10px;">
  <div id="messagebox"></div>
  <form action="" method="post">
    Your Name<br />
    <input id="yourname" style="width: 150px; height: 16px;"➥
```

```
  type="text" value="" onkeypress="autocomplete(this.value, event)" /><br />
    Your Task<br />
    <textarea style="height: 80px;"></textarea><br />
    <div align="right"><a href="javascript:closetask()">close</a></div>
  </form>
</div>
```

接下来一组函数大多用于完成一些清理以及请求获取的工作。函数closetask将在用户决定不再输入某个任务时，“关闭”或有效地隐藏任务窗口。而findPosX和findPosY函数则返回想实现自动完成功能的字段的x及y坐标。

之后的两个函数autocomplete和setvalue是实际负责实现自动完成功能的。函数autocomplete主要将检查当前输入的字符串（使用onkeypress事件），然后通过Ajax将该字符串传给一个名为autocomp.php的文件。这里有许多代码用来使该函数尽可能与各种浏览器兼容，涉及浏览器的按钮及事件处理都比较复杂。

最重要的问题是，必须将表示文本框当前值（Your Name字段）的字符串实时地传给一个PHP文件。接下来的一段代码显示了PHP脚本如何处理这些传入的信息。

```
<?php

  //包含所有名称的列表。
  //有些时候应基于数据库来生成这些数据。
  $names = array ("Lee Babin","Joe Smith","John Doe");
  $foundarr = array ();

  //遍历名称数据，并将匹配的添加到foundarr数组中。
  if ($_GET['sstring'] != ""){
    for ($i = 0; $i < count ($names); $i++){
      if (substr_count (strtolower ($names[$i]), ➥
strtolower ($_GET['sstring'])) > 0){
        $foundarr[] = $names[$i];
    }
  }
}
  //如果有匹配的值。
  if (count ($foundarr) > 0){
    //就显示他们。
    ?>
    <div style="background: #CCCCCC; border-style: solid; ➥
border-width: 1px; border-color: #000000;">
      <?php
        for ($i = 0; $i < count ($foundarr); $i++){
          ?><div style="padding: 4px; height: 14px;" onmouseover=➥
```

```
"this.style.background = '#EEEEEE'" onmouseout=➥
"this.style.background = '#CCCCCC'" onclick=➥
"setvalue ('<?php echo $foundarr[$i]; ?>')"><?php echo $foundarr[$i]; ?> ➥
</div><?php
        }
      ?>
    </div>
    <?php
  }

?>
```

autocomp.php文件将获取传入的字符串并尝试寻找匹配的名称。当发现了与查询的字符串匹配的名称，就将其载入到另一个数组中。这样做是为了确保在没有找到匹配名称的情况下，div元素的高度仍然为0。如果找到一个（或一组）匹配的名称，那么自动完成功能将显示这些相应的匹配项。当你单击某个匹配项时，就会把它的名称载入到相应的表单字段中（使用setvalue函数），并且关闭显示这些自动完成项的弹出表单。好了，现在用Ajax技术实现的自动完成功能就实现了（如图3-2所示）。

该功能不仅为用户节省了大量的时间，还会让他感到程序十分智能。使应用程序尽可能简单，以使新的因特网用户也能够很快地学会使用，这点很重要。

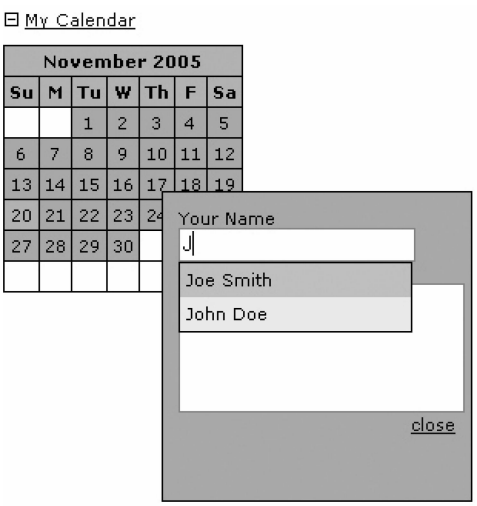

图3-2　自动完成使得数据输入更加流畅高效

3.3.3　表单验证

在第5章（将讲述与表单相关的内容）之前，我不想过于深入到表单验证中。不过展示一个基于Ajax来验证用户是否输入了错误检查难以发现的无效信息的技巧是很有意义的。绝大多数在

客户端验证的字段都是用JavaScript来判断是否为空、是否有无效信息，等等。但只有服务器端的脚本语言才能够判断这些信息对数据库存储是否有效。

不过现在有了Ajax这样的工具，就能够做到两全其美了。以往的工作方式是提交表单，服务器端进行检查，返回当前输入的所有值，并在屏幕返回时重新填充该表单。虽然这种工作方法也很好，但有许多任务需要编码，而且对于诸如“文件上传”这样的字段（它无法预填充）是无法处理的。在接下来的例子中，将使用与前面例子中一样的任务输入代码，不过当提交表单时，首先将检查在Your Name字段中输入的信息是否存在于脚本（存储名称的数组）之中。这类似于一个用户名验证器，该验证器用来避免用户使用一个网站中已注册的用户名。

在此我们就不列出所有的代码了，只展示那些改动的部分。首先，添加了一个名为validateform的新函数，其代码如下所示：

```
//functions.js
function validateform (thevalue){

  serverPage = "validator.php?sstring=" + thevalue;
  objID = "messagebox";

  var obj = document.getElementById(objID);
  xmlhttp.open("GET", serverPage);
  xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      obj.innerHTML = xmlhttp.responseText;
    }
  }
  xmlhttp.send(null);
}
```

该函数在页面的特定部分载入一个PHP脚本。下面的代码包含了对表单的修改：

```
<!-- theform.php -->
<div style="padding: 10px;">
  <div id="messagebox"></div>
    <form method="post">
      Your Name<br />
      <input id="yourname" style="width: 150px; height: 16px;"➥
type="text" value="" onkeypress="autocomplete(this.value, event)" />➥
<br />
     Your Task<br />
     <textarea style="height: 80px;"></textarea><br />
     <input type="button" value="Submit" onclick="validateform➥
  (document.getElementById('yourname').value)" />
  <div align="right"><a href="javascript:closetask()">close</a></div>
```

```
  </form>
</div>
```

如你所见，将添加一个名为messagebox的div元素（用来显示可能遇到的错误信息），以及一个用来调用validateform函数的按钮。当该按钮被单击时，将调用validateform函数，访问validator.php文件中的PHP脚本。该代码如下所示：

```
<?php
  //validator.php

  //保存有效名称的列表。
  //同样，这通常也会用数据库来实现。
  $names = array ("Lee Babin","Joe Smith","John Doe");

  if (!in_array (strtolower ($_GET['sstring']), strtolower ($names))){
    //那么返回一个错误。
    ?><span style="color: #FF0000;">Name not found...</span><?php
  } else {
    //此时将执行处理脚本。
    ?><span style="color: #FF0000;">Form would now submit...</span><?php
  }
?>
```

这个PHP脚本所要处理的内容就是根据传入的Your Name字段检查匹配的有效值。如果找到匹配的信息，那么该脚本将通过JavaScript提交该表单（在这里只是显示一个信息，关于用JavaScript提交一个表单的方法将在后续章节介绍）。如果发现错误，则可以快速地显示出该错误。这个小功能的好处是能够保持表单的内容不变（因为表单还没有提交）。这从编码的角度看能够节省大量时间，并且能够给用户提供流畅、智能的体验（参见图3-3）。

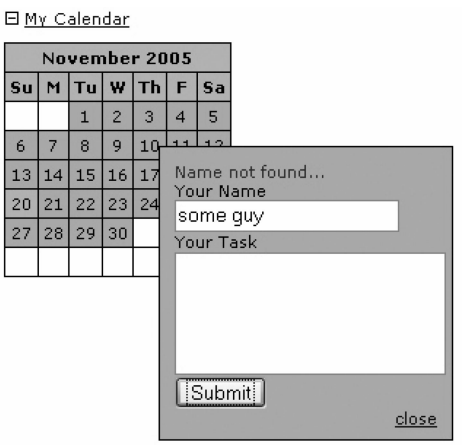

图3-3　如你所见，将能验证输入是否有效

3.3.4 工具提示

在因特网上的“工具提示”是一个很常见的DHTML技巧，即在鼠标的位置上显示一个小文本框，然后填入相关的提示信息。虽然这种方法很好，但在没有使用Ajax时，这个文本框中显示的信息通常是硬编码的，或者是通过一些服务器端的技巧实现的。在接下来的例子中，将使用Ajax动态地载入文本框中的信息。

作为对日历应用程序的有效补充，可以添加一个功能，当用户将鼠标放在某个日期时，将动态显示出该日所安排的任务。而PHP脚本将负责搜索数据库，并返回与该日相关的任务。由于现在还没有涉及数据库，因此现在构建的脚本仍将使用任务数组来提供该信息，而将重点放在工具提示功能上。

首先来看看calendar.php文件为了该变化所需修改的代码：

```
//生成合适的行数……
for ($k = 1; $k <= $numrows; $k++){
  ?><tr><?php
  //使用7列(表示7天)……
  for ($i = 0; $i < 7; $i++){
    $startdate++;
    if (($startdate <= 0) || ($startdate > $lastday)){
      //如果在日历中有空白的格子。
      ?><td style="background: #FFFFFF;"> </td><?php
    } else {
    if ($startdate == date("j") && $month == date("n") && $year == date("Y")){
      <td onclick="createform(event)" class="calendartodayoff"➥
onmouseover="this.className='calendartodayover'; checkfortasks ➥
('<?php echo $year . "-" . $month . "-" . $startdate; ?>',event);"➥
onmouseout="this.className='calendartodayoff'; hidetask();">➥
<?php echo date ("j"); ?></td><?php
      } else {
        <td onclick="createform(event)" class="calendaroff"➥
onmouseover="this.className='calendarover'; checkfortasks➥
('<?php echo $year . "-" . $month . "-" . $startdate; ?>',event);" ➥
onmouseout="this.className='calendaroff'; hidetask();">➥
<?php echo $startdate; ?></td><?php
      }
    }
  }
  ?></tr><?php
}
```

主要的变化是，在onmouseover事件中调用checkfortasks函数以及在onmouseout事件中调用hidetask函数。它主要是将用户鼠标所指向的当前日期传给相应的函数，该函数位于

functions.js文件中（代码如下所示）：

```
//functions.js
function checkfortasks (thedate, e){

  theObject = document.getElementById("taskbox");

  theObject.style.visibility = "visible";

  var posx = 0;
  var posy = 0;

  posx = e.clientX + document.body.scrollLeft;
  posy = e.clientY + document.body.scrollTop;

  theObject.style.left = posx + "px";
  theObject.style.top = posy + "px";

  serverPage = "taskchecker.php?thedate=" + thedate;
  objID = "taskbox";

  var obj = document.getElementById(objID);
  xmlhttp.open("GET", serverPage);
  xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      obj.innerHTML = xmlhttp.responseText;
    }
  }
  xmlhttp.send(null);
}

function hidetask (){
  tObject = document.getElementById("taskbox");

  tObject.style.visibility = "hidden";
  tObject.style.height = "0px";
  tObject.style.width = "0px";
}
```

同样，要实现工具提示也要使用一些DHTML技巧，用来隐藏用于载入任务脚本的div元素。要使其正常运作就需要创建一个新的div，代码如下所示。

```
<body>
  <div id="createtask" class="formclass"></div>
  <div id="autocompletediv" class="autocomp"></div>
```

```
  <div id="taskbox" class="taskboxclass"></div>
  <p><a href="javascript://" onclick="showHideCalendar()"><img id="opencloseimg"➥
src="images/plus.gif" alt="" title"" style="border: none;➥
width: 9px; height: 9px;" /></a> <a href="javascript://" onclick=➥
"showHideCalendar()">My Calendar</a></p>
  <div id="calendar" style="width: 105px; text-align: left;"></div>
</body>
```

checkfortasks函数将获取一个日期，然后（通过Ajax）将其传给一个名为taskchecker.php的新文件。taskchecker.php文件将从数据库中读取数据，然后在动态创建的悬浮式的div元素（与任务输入表单并不一样）中显示相应的任务信息。在本例中，因为还没有涉及与数据库的集成，因此将通过一个数组来实现。这个taskchecker.php文件的代码如下所示：

```
<?php
  //taskchecker.php
  //实际的任务数据。
  //这通常是从数据库中载入的。
  $tasks = array ("2005-11-10" => 'Check mail.',"2005-11-20" => 'Finish Chapter 3');

  $outputarr = array ();

  //遍历数据并检查是否有匹配的值。
  while ($ele = each ($tasks)){
    if ($ele['key'] == $_GET['thedate']){
      $outputarr[] = $ele['value'];
    }
}
  //如果有匹配的项……
  if (count ($outputarr) > 0){
    ?>
    <div class="taskchecker">
      <div class="tcpadding">
      <?php
        for ($i = 0; $i < count ($outputarr); $i++){
          echo $outputarr[$i] . "<br />";
        }
      ?>
      </div>
    </div>
    <?php
  }
?>
```

如你所见，系统将遍历整个数组（该任务实际上通常用数据库查询来实现），并将匹配的信息载入到outputarr数组变量中。如果发现了匹配项，则将它们显示在实时创建的div元素中。

其结果就是一个动态的任务列表，如图3-4所示。

图3-4　通过Ajax工具提示来显示相应的任务

3.4　小结

好了，在稍显复杂但很基础的客户端/服务器上采用的Ajax/PHP的示例还不错。现在应该对如何组合JavaScript、DHTML、Ajax和PHP来创建很酷的小应用程序有了一定的了解。它采用了从头构建的方法，并且将会是你涉及更多高级话题的良好基础。

至此为止，重要的只是了解Ajax概念背后的基础知识。首先，你会发现需要比以前编写更多的JavaScript程序。对我而言，第一次使用Ajax时就发现这是一个复杂的问题，不过相信你的JavaScript编程技能会不断进步。还需要对CSS以及诸如Firefox的JavaScript控制台、DOM解析器之类的工具有良好的了解。其中，JavaScript控制台（如图3-5所示）对于寻找偶尔出现的JavaScript语法错误是十分有效的。

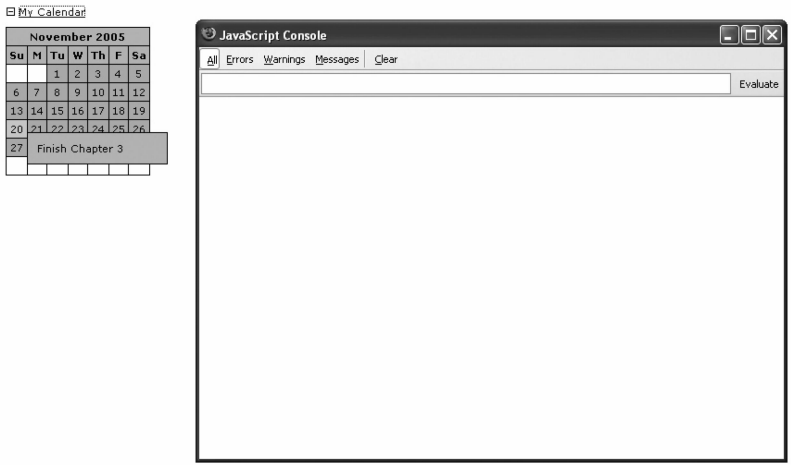

图3-5　Firefox的JavaScript控制台

深入掌握了JavaScript和CSS之后，基于Ajax的应用程序将有无穷的空间。而从合适的PHP脚本中获取合适的信息，并返回/显示出所需要的内容，这是一个现实的问题。在本书后续章节中，

将会基于本章总结的原则构建一些很强大的应用程序。接下来，我们先来看看服务器端重要功能之一：数据库。

第 4 章 数据库驱动的 Ajax

现在你已经知道如何使用PHP和Ajax来实现一些动态效果及功能，那么就可以开始试验PHP中一些更复杂、更强大的功能了。结合使用PHP这种健壮的服务器端语言与Ajax风格的JavaScript的优势在于，可以实现一些仅通过JavaScript不容易（甚至不能）实现的任务。其中数据库存储与检索就是这样的功能。

毫无疑问，MySQL与PHP是开发人员的梦幻组合。它们的功能和文档都很丰富、健壮。虽然MySQL通常是需要授权费的，不过FLOSS（Free/Libre及开源软件）例外，它使你能够在PHP中使用MySQL。FLOSS准许你免费使用MySQL（关于FLOSS的更多信息请参阅`www.mysql.com/company/legal/licensing/foss-exception.html`中的MySQL文档）。从任务处理的角度，PHP和MySQL连接是最易于使用的，而且执行结果也相当好。基于当前的MySQL 5.0，可以实现先前只有在Oracle这样昂贵的数据库解决方案中才能够实现的功能。

MySQL 5.0添加了许多功能，其中比较强大的包括存储过程、触发器、视图等。存储过程让你能够创建和访问仅在MySQL服务器上执行的函数。这样开发人员就能够将更多的工作交给MySQL服务器，并且所需使用的脚本语言也将更少。触发器可以预设一些查询请求，当MySQL服务器中特定事件触发时执行它。与存储过程类似，触发器使MySQL服务器承担了更多的处理任务，从而减轻了脚本语言的压力。视图可以用来创建引用数据库信息的自定义“视图”。称之为“视图（view）”是因为它为“查看（view）”数据库中特定数据提供了一种简单有效的方法。以前这些功能只有在大型数据库系统（如Oracle）中才有，MySQL包含了这些功能也证明其在数据库领域的竞争力越来越强。

结合PHP、MySQL、Ajax的JavaScript将是一种很强大的工具，而且许多开发人员都已经了解了应用方法。实际上，有些用来管理MySQL数据库的整个软件应用程序就是基于Ajax架构构建的。图4-1所示的如TurboDbAdmin（`www.turboajax.com/turbodbadmin.html`）之类的在线应用程序就展现了当PHP、Ajax和MySQL结合时能够完成的大量功能。TubroDbAdmin代表了一类基于Ajax的应用程序。从插入行、维护行、切换标签页、执行查询到创建动态内容等所有事情都是由基于Ajax的方法来处理的。总之，TubroDbAdmin证明了Ajax有能力处理复杂的数据库管理。

虽然TubroDbAdmin能够在MySQL服务器上完成令人称奇的任务，但其安装与实现都很简单，不过与phpMyAdmin之类的基于PHP的MySQL管理系统相比，其健壮性还是会稍差一些。不过，TurboDbAdmin提供了一个有趣的视角，让你了解Ajax能够完成什么。

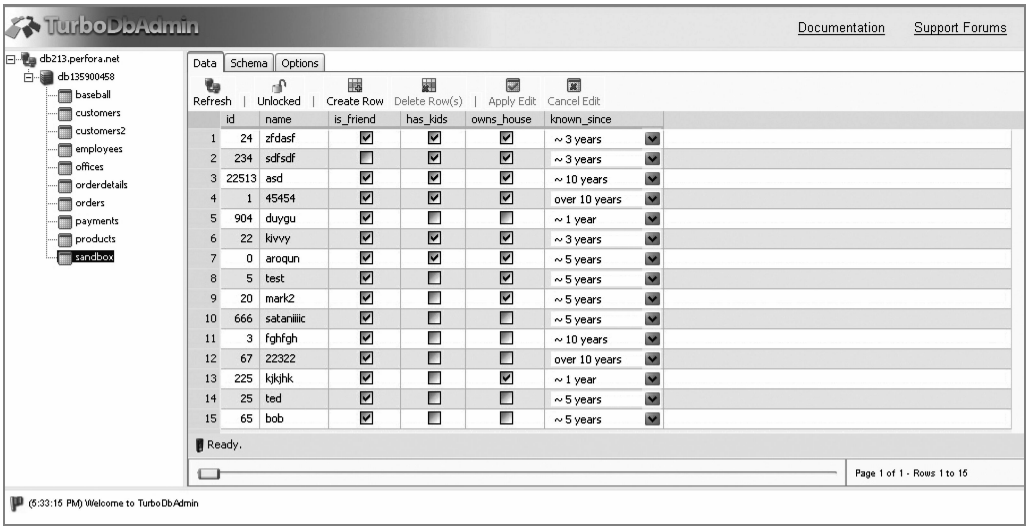

图4-1　如TurboDbAdmin之类的Ajax驱动的应用程序展现了PHP和JavaScript与MySQL结合时能够实现的功能

本章的重点是，如何创建一个易与MySQL服务器连接的、Ajax驱动的在线应用程序。

4.1　MySQL 简介

显然，如果要运行本章的例子，还需要在服务器上运行一些应用程序。为了使该例子尽可能灵活，我们将展示如何用PHP代码来连接MySQL服务器，这些代码需运行在支持PHP 5的服务器上。由于在本书写作时，MySQL 5还很新，许多服务器并未升级，因此本例仍然将展示如何与MySQL 4协作。因此所需的环境就是在Apache（或等价的）服务器上安装了PHP 5及MySQL 4或更高版本（3.0版本也可以正常运行）。

在使用MySQL之前，必须先回顾一些核心的原则。MySQL在执行数据库查询时将使用SQL（结构化查询语言）语言。因此了解SQL是如何工作的以及有哪些查询类型，对于我们要开发的功能有用是很重要的。本书假定你已经了解了数据库创建和执行查询的基础操作。

为了创建实际可用的应用程序，我们将继续构建从第3章开始的应用程序。主要的计划是，将前面的任务管理解决方案和与MySQL数据库相关的面向Ajax的JavaScript及PHP代码连接起来，从而可以动态地在数据库中获取信息、保存数据。当这个任务完成时，就将拥有一个功能完整的任务管理系统，可以应用于任何需要的场景中。

4.2 连接到 MySQL

为了访问和使用MySQL数据库，首先必须创建一个数据库，然后在数据库中创建和维护一系列表。但要连接到数据库还需要创建一个数据库用户，使其有权限访问这个数据库，并为其赋予一个密码。在以下的例子中，将创建一个名为taskdb的数据库，并在其中创建了一个名为apressauth的用户，密码设置为tasks。为了执行此类数据库管理操作，可以转用MySQL提供的命令行界面，或者寻找一个更健壮的解决方案。对于基于Web的，我偏爱phpMyAdmin（www.phpmyadmin.net），而对于基于远程连接访问的，则比较偏爱SQLyog（www.webyog.com/sqlyog）。这些都是免费的解决方案，都可以选用。

要用PHP来连接MySQL数据库，必须使用mysql_connect函数。看一看下列的代码，它们位于dbconnector.php中，用来连接数据库：

```
<?php

  //dbconnector.php

  //定义连接mySQL所需的变量。
  define ("MYSQLHOST", "localhost");
  define ("MYSQLUSER", "apressauth");
  define ("MYSQLPASS", "tasks");
  define ("MYSQLDB", "taskdb");

  function opendatabase(){
    $db = mysql_connect (MYSQLHOST,MYSQLUSER,MYSQLPASS);
    try {
      if (!$db){
        $exceptionstring = "Error connecting to database: <br />";
        $exceptionstring .= mysql_errno() . ": " . mysql_error();
        throw new exception ($exceptionstring);
      } else {
        mysql_select_db (MYSQLDB,$db);
      }
      return $db;
    } catch (exception $e) {
      echo $e->getmessage();
      die();
    }
  }

?>
```

如你所见，与MySQL数据库的连接包括两个部分。首先，mysql_connect函数将尝试连接

数据库，并验证用户名与密码。如果连接成功，将保留这个到服务器的连接。在此，必须指定要连接哪个数据库。因为可能会为每个MySQL用户分配多个数据库，而脚本必须知道使用哪个数据库，这可以通过mysql_select_db函数来实现。如果一切顺序，就能够获得一个与数据库的连接，这时就准备好进入下一节：查询数据库。

4.3　查询 MySQL 数据库

要对数据库表发起一个有效的查询，首先必须有这张数据库表。在此我们将创建一张名为block的数据库表，用来存放一些随机词。以下SQL代码（MySQL用来执行操作的语言）将创建该数据库表：

```
CREATE TABLE block (
  blockid INT AUTO_INCREMENT PRIMARY KEY,
  content TEXT
);
```

这时名为block的数据库表就创建完成了，接着可以再使用SQL来插入一些数据。以下代码将在block数据库表中插入八条随机词：

```
INSERT INTO block (content) VALUES ('frying');
INSERT INTO block (content) VALUES ('awaits');
INSERT INTO block (content) VALUES ('similar');
INSERT INTO block (content) VALUES ('invade');
INSERT INTO block (content) VALUES ('profiles');
INSERT INTO block (content) VALUES ('clothes');
INSERT INTO block (content) VALUES ('riding');
INSERT INTO block (content) VALUES ('postpone');
```

现在不仅配置了一张有效的数据库表，还在该表中保存了一些信息，因此就可以基于Ajax和PHP动态地向数据库发出查询，并且不会刷新页面。Ajax功能可以基于不同的事件来触发。当然最经常用来触发Ajax代码的事件（基本上是可以“捕获”以执行某些代码的操作）是onclick。证明该事件很有用的理由之一就是许多HTML对象都能够触发该事件。通过使用onclick事件可以实现许多很有趣的功能。看看下面的这段代码，它将随机地从随机词数据库中获取一些单词，然后填充到被单击的元素中。当页面首次载入时，sample4_1.html呈现的内容如图4-2所示。

图4-2　随机词生成文本框，在onclick事件触发前

现在先看看下面列出的sample4_1.html代码。你会发现每个方块都注册了一个onclick事件。这就是将触发Ajax功能的操作。

```
<?php /* sample4_1.php */ ?>
```

```
<!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">
<head>
<title>Sample 4_1</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="style.css" />
<script type="text/javascript" src="functions.js"></script>
</head>
<body>
  <?php
    for ($i = 1; $i < 9; $i++){
      ?>
      <div class="dborder" id="dborder<?=$i?>" onclick="grabword (this.id)"></div>
      <?php
    }
  ?>
</body>
</html>
```

当任何方块被单击时，就会启动一个对grabword函数的调用，它将把当前对象的ID作为参数。如果方块中没有内容，该函数将执行一个Ajax请求来填充它，否则就将其内容清空。下面所示的JavaScript函数（包含于funciton.js中）就是用来实现该功能的。

```
//functions.js

//创建一个布尔型变量，用来检查是否为有效的IE实例。
var xmlhttp = false;

//检查是否使用的是IE。
try {
  //如果Javascript的版本大于5。
  xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
  //如果不是，则使用老版本的ActiveX对象。
  try {
    //如果使用的是IE浏览器。
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  } catch (E) {
    //否则肯定使用的是非IE浏览器。
    xmlhttp = false;
  }
}

//如果使用的是非IE浏览器，则创建一个该对象的JavaScript实例。
```

```
if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
  xmlhttp = new XMLHttpRequest();
}
// 用来执行单词收集器脚本的函数。
function grabword (theelement){
  //如果方块中没有内容，就执行Ajax来填充它。
  if (document.getElementById(theelement).innerHTML.length == 0){
    //修改背景色。
    document.getElementById(theelement).style.background = "#CCCCCC";
    serverPage = "wordgrabber.php";
    var obj = document.getElementById(theelement);
    xmlhttp.open("POST", serverPage);
    xmlhttp.onreadystatechange = function() {
      if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        obj.innerHTML = xmlhttp.responseText;
      }
    }
    xmlhttp.send(null);
  } else {
    //修改背景色。
    document.getElementById(theelement).style.background = "#FFFFFF";
    //如果方块已填充有内容则清空它。
    document.getElementById(theelement).innerHTML = "";
  }
}
```

首先将创建一个XMLHttpRequest对象，然后检查方块中是否已经有内容。如果该方块中已填充过内容，则grabword函数只是将该对象的innerHTML属性设置为空。如果不是空，则该函数将发起一个Ajax请求，然后用wordgrabber.php返回的结果来填充这个方块。接下来看看在wordgrabber.php文件中如何执行该查询的：

```
<?php

  //wordgrabber.php

  //完成数据库连接所需的文件。
  require_once ("dbconnector.php");
  //打开该数据库。
  $db = opendatabase();

  //然后执行一个查询，从数据库中获取一个随机词。
  $querystr = "SELECT content FROM block ORDER BY RAND() LIMIT 1";

  if ($myquery = mysql_query ($querystr)){
```

```
    $mydata = mysql_fetch_array ($myquery);
    echo $mydata['content'];
  } else {
    echo mysql_error();
  }

?>
```

该PHP脚本首先引入在前面已经构建的处理数据库连接的脚本（dbconnector.php），然后调用opendatabase函数来连接数据库。这样就只需要构建一个SQL查询来从block表中获得一个随机词，最后将随机词的内容显示出来。在图4-3中显示了方块单击过和未单击过的效果。

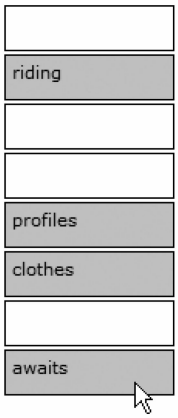

图4-3 单击某个方块时，将通过Ajax控制的PHP数据库访问来填充一个随机词

4.4 MySQL 技巧和警告

在用Ajax连接MySQL时，有几个方面需要谨记。首先，要注意通过Ajax的接口来连接数据库时，如果不小心就会成为数据库的噩梦。想想如果在针对同个用户的同个页面有多个处理，过多的连接可能会让服务器的负载急剧增长。如果一个Web页面中有三个地方将通过Ajax访问数据库，就意味着在该页面中处理该功能时，每个用户都将会打开三个请求。如果是在一个很繁忙的网站中，数据库的负载就可能会过高。不过更多高级的连接处理方法也将出现，以解决Ajax功能所带来的需求增长问题。但不管怎么样关注这个问题是重要的，这样就不会因为没有认识到该问题而产生负载过大的后果。

接下来，要考虑是如何更加人性化地载入MySQL返回的结果。例如在采用页面刷新机制时想输出一个错误信息，那么在页面的某个位置载入这个错误信息是很简单的。但使用了Ajax之后，经常会载入了一些比较小的、更不明显的内容。因此必须考虑如何使用户注意到这些变化。特别是MySQL的错误消息通常都比较大，因此更好的方法是将MySQL错误用电子邮件发送给管理员，

然后向用户输出一个量比较小的警告消息。

安全问题必须警惕。虽然通过Ajax访问的脚本看起来要比完全呈现在页面中的脚本更安全一些，但它们实际上是一个易被攻击的地方，甚至更容易出问题。原因是所有的JavaScript代码对于任何人而言都是公开的，可以通过查看页面的源文件来获得，因此能够找出程序中引用了什么文件。如果这些文件中的脚本没有对直接访问进行验证，那么可能被恶意利用。因为迄今为止本书在Ajax请求中只使用过GET请求，因此还可能存在代码注入攻击风险，特别是SQL注入。

SQL注入是指在通过查询字符串（浏览器中的地址栏）传入恶意代码，会导致该脚本中所包含的动态查询出现问题。正因为如此，在从查询字符串中获取信息来动态创建MySQL查询时，应该注意这个问题。绝大部分数据库软件都提供了去除注入数据（在MySQL中，它是通过一个名为`mysql_real_escape_string`的函数实现的）功能。还有一种减轻SQL注入问题的简单方法，对查询字符串中的所有变量通过`addslashes`函数（针对字符串型变量）或`intval`函数（针对整型变量）进行封装。总而言之，有人可以直接访问你的脚本（特别是动态查询），因而需要做相应的防范。

4.5　将基于 Ajax 的数据库查询应用到工作中

现在已经掌握了执行基于Ajax的数据库请求的基础知识，接下来我们将继续构建前面的日历程序。可以继续使用前面创建的数据库和用户，但需要在数据库中添加一些新信息。在此将创建一个名为`task`的数据库表，其配置方法如下所示：

```
CREATE TABLE task (
  taskid INT AUTO_INCREMENT PRIMARY KEY,
  userid INT,
  thedate DATE,
  description TEXT
);
```

`taskid`字段是标识每个任务的唯一数字ID（可以启用`auto_increment`和`primary key`来进行完整性控制）。`userid`字段将作为一个`foreign key`来关联设置该任务的用户。`thedate`字段将为每个任务设置一个日期值（YYYY-MM-DD），而`description`字段则用来保存实际的任务描述。针对该例子的用途，可以在该表格中填充以下字段信息：

```
INSERT INTO task (userid, thedate, description) VALUES ➥
(1,'2005-12-04','Finish chapter 4');
INSERT INTO task (userid, thedate, description) VALUES ➥
(1,'2005-12-25','Christmas!');
```

接下来将配置一个名为`user`的数据库表，用来保存可以在系统中输入任务信息的用户。

```
CREATE TABLE user (
```

```
  userid INT AUTO_INCREMENT PRIMARY KEY,
  name TINYTEXT
);
```

该表包括两个字段，一个用来保存唯一标识符（userid，它将与task数据库表相关联），另一个是保存用户姓名的name字段。可以在该数据库表中添加一条记录：

```
INSERT INTO user (userid, name) VALUES ('1','Lee Babin');
```

当数据库表创建完之后，接下来就需要配置数据库连接脚本了。要使用PHP MySQL程序库来连接某个数据库，就必须为mysql_connect函数指定连接信息。看看下面的代码块，它将用来连接MySQL数据库：

```
<?php

  //dbconnector.php

  //定义变量mysql connection。
  define ("MYSQLHOST", "localhost");
  define ("MYSQLUSER", "apressauth");
  define ("MYSQLPASS", "tasks");
  define ("MYSQLDB", "taskdb");

  function opendatabase(){
    $db = mysql_connect (MYSQLHOST,MYSQLUSER,MYSQLPASS);
    try {
      if (!$db){
        $exceptionstring = "Error connecting to database: <br />";
        $exceptionstring .= mysql_errno() . ": " . mysql_error();
        throw new exception ($exceptionstring);
      } else {
        mysql_select_db (MYSQLDB,$db);
      }
      return $db;
    } catch (exception $e) {
      echo $e->getmessage();
      die();
    }
  }

?>
```

正如你所见，用来创建数据库连接的dbconnector.php脚本简单而高效。将这些脚本引入到你认为需要的文件中，就可以通过引用$db变量来执行数据库查询。把数据库登录信息存储在一个地方，当数据库连接信息改变时，就能够减少所需的维护量。没有到处散布数据库信息，也能够有效控制安全风险。

4.6　更完善的自动完成功能

有了连接到数据库的手段之后，就可以开始对前几章的例子中预留的占位代码进行替换与升级。和使用静态数组来保存信息相比，用数据库来保存效果更佳，只需引入数据库连接脚本（包括连接到数据库的PHP脚本）并对表user执行一个查询语句，就能够实时从数据库中构建一个最新的姓名列表。autocomp.php和validator.php这两个文件需要进行较大的改动。

```
<?php

  //autocomp.php

  //添加数据库器连接器
  require_once ("dbconnector.php");
  //然后打开一个数据库连接
  $db = opendatabase();

  $foundarr = array ();
  //配置动态的查询字符串。
  $querystr = "SELECT name FROM user WHERE name LIKE ➥
LOWER('%" . mysql_real_escape_string ($_GET['sstring']) . "%') ORDER BY name ASC";

  if ($userquery = mysql_query ($querystr)){
    while ($userdata = mysql_fetch_array ($userquery)){
      if (!get_magic_quotes_gpc()){
        $foundarr[] = stripslashes ($userdata['name']);
    } else {
        $foundarr[] = $userdata['name'];
    }
  }
} else {
  echo mysql_error();
}

  //如果有匹配项，则对其进行遍历并显示它们。
  if (count ($foundarr) > 0){
    ?>
    <div style="background: #CCCCCC; border-style: solid; border-width: 1px;➥
border-color: #000000;">
    <?php
      for ($i = 0; $i < count ($foundarr); $i++){
        ?><div style="padding: 4px; height: 14px;" onmouseover=➥
"this.style.background = '#EEEEEE'" onmouseout=➥
"this.style.background = '#CCCCCC'" onclick=➥
"setvalue ('<?php echo $foundarr[$i]; ?>')"><?php echo $foundarr[$i]; ?></div><?php
```

```
        }
      ?>
    </div>
    <?php
  }

?>
```

注意前面的代码对autocomp.php文件做了什么修改。现在不再使用一个数组来检查匹配的姓名，系统将通过LIKE操作符在数据库中执行匹配操作。该系统可以动态检查数据库中的新名称，因此效果要好得多。

同样，validator.php文件也将采用与autocomp.php文件相同的验证检查。此时采用的方法要比基于姓名数组进行精确匹配的方法更好，系统将基于实际的数据库来对姓名进行精确的匹配。另外该文件的功能先进，现在还提供了一种保存新姓名的方法。从中不难发现代码流是大致相同的，但现在是通过实际的数据存储模型来实现的，并将产生一个验证功能更强的表单（如图4-4所示）。

```
<?php

  //validator.php

  //添加数据库连接器。
  require_once ("dbconnector.php");
  //然后打开一个数据库连接。
  $db = opendatabase();

  //配置动态查询字符串。
  $querystr = "SELECT userid FROM user WHERE name = ➥
LOWER('" . mysql_real_escape_string ( $_GET['sstring']) . "')";

  if ($userquery = mysql_query ($querystr)){
    if (mysql_num_rows ($userquery) == 0){
      //然后返回一个错误。
      ?><span style="color: #FF0000;">Name not found...</span><?php
    } else {
      //在此将转到处理脚本。
      ?><span style="color: #FF0000;">Form would now submit...</span><?php
    }
  } else {
    echo mysql_error();
  }

?>
```

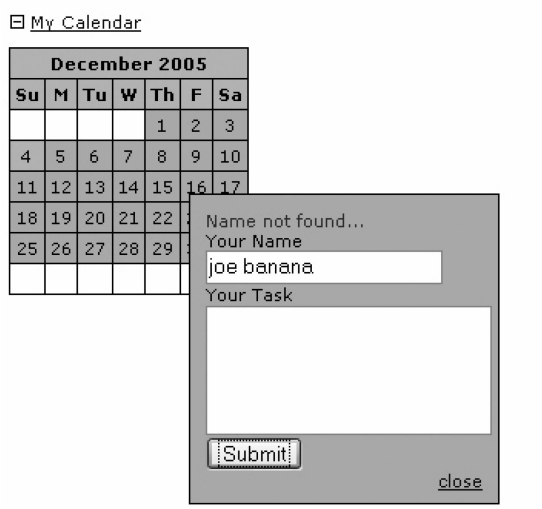

图4-4　带数据库支持的验证功能

4.7　载入日历

对于这个基于Ajax实现的日历而言，接下来要更新的就是日历本身。当完成动态任务列表创建功能之后，日历就能够从数据库中检索信息并载入到每一天的任务列表中。这可以通过在数据库中查询针对某一天的任务记录来实现。看一下taskchecker.php中有什么变化，它的功能是标识出指定某一天的所有任务记录：

```
<?php

  //taskchecker.php

  //添加数据库连接器。
  require_once ("dbconnector.php");
  //打开数据库。
  $db = opendatabase();

  //配置动态查询字符串。
  $querystr = "SELECT description FROM task WHERE thedate=➥
'" . addslashes ($_GET['thedate']) . "'";
  if ($datequery = mysql_query ($querystr)){
    if (mysql_num_rows ($datequery) > 0){
      ?>
        <div style="width: 150px; background: #FFBC37; border-style: solid; ➥
border-color: #000000; border-width: 1px;">
          <div style="padding: 10px;">
```

```
            <?php
              while ($datedata = mysql_fetch_array ($datequery)){
                if (!get_magic_quotes_gpc()){
                  echo stripslashes ($datedata['description']);
                } else {
                  echo $datedata['description'];
                }
              }
            ?>
          </div>
        </div>
      <?php
    }
  } else {
    echo mysql_error();
  }

  //关闭数据库连接。
  mysql_close ($db);

?>
```

如你所见，在此将再次载入数据库连接器脚本，然后调用opendatabase函数。当数据库连接打开后，再创建一个数据库查询，以获得在某一天设定的所有任务。然后使用mysql_num_rows函数来决定特定的一天是否有什么任务，然后通过mysql_fetch_array函数来遍历显示所有任务。后续的清理工作也很重要，通过调用mysql_close函数来关闭与数据库的连接。成功的任务查询的结果如图4-5所示。

图4-5 用Ajax在动态工具提示中显示已指派的任务

4.8 小结

总而言之，在实现Ajax与数据库的协作过程中并没有什么特别困难的操作。不过谨记可移植

性和安全问题是很重要的。数据库是网络攻击中最主要的目标之一，包括SQL注入与破解。编写一段完成连接字符串设置的代码，就能够快速、有效地在一个地方修改这些信息。确保这些信息的安全很重要，将其隐藏在服务器端语言文件（诸如PHP）中也很有效。SQL注入的处理有许多方法，不过最为重要的是对通过查询字符串传入的数据进行完整性验证。

将数据库的强大功能与Ajax的效率结合在一起，这个在线任务管理系统将会变得很完善。下一章将引入表单处理能力（Ajax风格），并往数据库添加实际的任务数据，从而完成这个任务管理系统。

第 5 章 表　单

在上一章中，我们学习了如何从MySQL数据库中检索数据。这只是完成了事情的一半，即从数据库中获取信息以及在不同数据库表中执行动态执行，而另一半则是将信息动态地保存到数据库中。

用户的输入通常是用表单元素来收集的。在Web中有很多不同类型的表单元素，为从用户端获取输入信息提供了多种途径。如果希望表单的处理过程尽可能智能化，最重要的是考虑在用户输入信息时应该有哪种元素。表5-1中展示了开发人员可以访问的表单元素。

表5-1　HTML表单元素

元　素	描　述
button	单击后将通过脚本执行某个操作（通常是基于JavaScript的）
checkbox	提供一系列选择框，让用户做出选择
hidden	用来给表单传递相关信息，但用户不会看到它的值
image	其功能与submit按钮元素类似，不过可以用src属性指定一个图像。添加该功能后，在图像x、y坐标之内的单击都将完成表单提交操作
radio	提供一组选项，但用户只能够从中选择一个。如果整个radio按钮组的名字都相同，那么每次选择其中一个就会取消前一次的选择。它们的工作方式与checkbox类似，区别在于radio只能够返回一种选择（在一组中），而checkbox则可以返回零个或多个
reset	reset按钮提供了一种在表单载入后重新设置表单的方法
select	在一个下拉选择框中提供多种选项。可以将select元素一次最多能够选择的项目数设置成零个或多个（这种形式通常也被称为列表元素）
submit	submit按钮默认用来提交表单。它将自动执行form标签中action字段所指定的脚本。要注意，在表单中可能存在多个submit按钮
text	用来输入信息的基本文本字段
textarea	比text字段功能更强一些，允许输入多行信息并且提供了滚动条
file	提供了一种上传文件的手段。同时还提供了一个Browse按钮，用来在当前计算机中查找文件

多年来，开发人员一直很好（有时也很糟糕）地应用了这些表单元素，创建出了许多有用的基于Web的应用程序。随着时间的推移，编码人员和设计人员都提供了许多针对不同Web功能的良好实现方法。当然也缺少了一个环节，那就是使其能够立即工作（或表面上看起来是这样），而无需页面刷新。最终，通过Ajax也能够达成这个目标。

5.1　引入 Ajax：GET 与 POST

通过常规方法提交一个表单时，必须在form标签中指定用GET还是POST请求类型完成数据的传送。决定使用哪种方法是很重要的。使用GET方法将把所有表单元素的值放在查询字符串中传送。这也意味着浏览器将把所有提交字段的值组装成一个很长的字符串，然后将该字符串传给action属性指定的脚本。使用GET方法存在两个方面的问题。第一个问题与能够传送的数据长度有关，因为GET方法能够在查询字符串中传递的信息很有限。而查询字符串具体的长度限制在不同的浏览器中是不同的；不过对于多数Web应用程序而言还不够长。

GET方法带来的第二个问题会在使用动态数据库查询时出现，因为它是根据从GET请求中获取的信息来执行的。例如当单击某个链接时，将执行一个删除某条记录的数据库脚本。而搜索引擎可能会尝试单击该链接，如果该脚本没有对这种情况进行处理，那么你会很快发现信息已丢失。

因此绝大多数有经验的Web开发人员都会使用POST来实现大部分的表单提交操作（特别是涉及动态数据库查询的表单）。POST方法能够更安全地完成值的传输，并且对用户没有影响。使用这种方法，对处理脚本收到的数据可以按开发人员的构想进行限制。但这并不意味着可以忽略对其的验证，只是你可以对发送和接收的信息给予更多的控制。

无论如何，当使用Ajax方法提交表单时，你仍然可以选择用哪种方法来提交表单；但在本书的示例中将使用POST方法。

5.2　值的传递

当要在一个标准的表单中传递值时，只需创建一个submit或image元素，它会自动将表单的值传给form标签中action属性指定的脚本。如果选择使用submit元素，那么所有的值将捆在一起，然后直接传给相应的脚本，与开发人员之间很少或不需要交互。不过通过Ajax提交表单时，将值传给相应脚本的过程要复杂一些。

要注意的第一件事是，为向服务器发送的异步请求提供要传输的字符串，需要更复杂的构建过程，同时也能够在处理脚本调用之前执行更多JavaScript脚本（诸如表单验证）。虽然这些额外的能力是很好的，但也会带来额外的复杂性代价。

主要过程是，一个XMLHttpRequest实例使用所需的表单值构建查询字符串，然后传给该请求，并为该请求设置相应的报头。通过示例演示要比文字解释更容易，因此将为前一章中的任务管理系统添加相应的表单提交功能。在代码清单5-1和代码清单5-2中可以看到修订后的代码，它通过Ajax功能的JavaScript代码完成“任务创建”表单的提交功能。

首先修改了theform.php文件，使其能够提供实际的提交值。你会发现该表单现在包含四个元素。第一个元素是一个text字段，用来输入用户名。接下来一个元素是textarea字段，

用来输入要提醒的任务描述。第三个字段是一个hidden字段，用来保存从calendar.php文件传过来的日期值（如图5-1所示）。最后一个字段是submit按钮，它用来触发向服务器发送的基于JavaScript的Ajax请求。代码清单5-1和代码清单5-2中列出的脚本展示出对calendar.php和theform.php文件的修改，以使其能够实现日期值的传递。

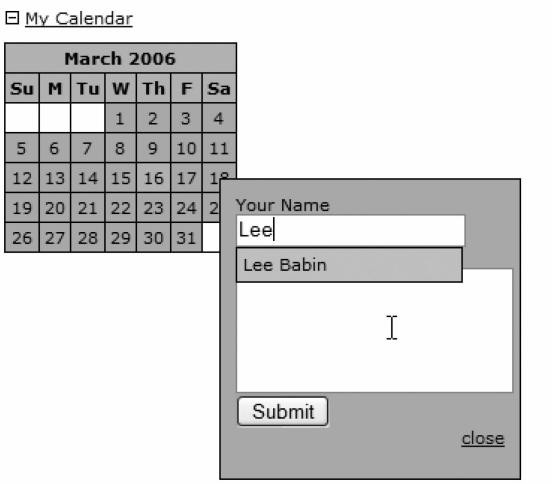

图5-1 运行中的基于Ajax的表单提交

代码清单5-1 动态显示表单的代码（theform.php）

```
<?php
  //theform.php
?>
<div style="padding: 10px;">
  <div id="themessage">
  <?php
    if (isset ($_GET['message'])){
      echo $_GET['message'];
    }
  ?>
</div>
<form action="process_task.php" method="post" id="newtask" name="newtask">
  Your Name<br />
  <input name="yourname" id="yourname" style="width: 150px; height: 16px;"➥
type="text" value="" onkeypress="autocomplete(this.value, event)" /><br />
  Your Task<br />
  <textarea style="height: 80px;" name="yourtask" id="yourtask"></textarea><br />
  <input type="hidden" name="thedate" value="<?php echo $_GET['thedate']; ?>" />
  <input type="button" value="Submit" onclick="submitform➥
(document.getElementById('newtask'),'process_task.php','createtask'); ➥
```

```
return false;" />
    <div align="right"><a href="javascript:closetask()">close</a></div>
  </form>
</div>
```

代码清单5-2　显示日历的代码（calendar.php）

```
<?php

  //calendar.php

  //检查月份和年份值是否存在。
  if (!$_GET['month'] && !$_GET['year']) {
    $month = date ("n");
    $year = date ("Y");
  } else {
    $month = max(1, min(12, $_GET['month']));
    $year = max(1900, min(2050, $_GET['year']));
  }

  //计算将查看的月份信息。
  $timestamp = mktime (0, 0, 0, $month, 1, $year);
  $monthname = date("F", $timestamp);

  //Now let's create the table with the proper month.
  ?>
  <table style="width: 105px; border-collapse: collapse;" border="1"
            cellpadding="3" cellspacing="0" bordercolor="#000000">
    <tr style="background: #FFBC37;">
    <td colspan="7" style="text-align: center;"
        onmouseover="this.style.background=#FECE6E'"
        onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;"><?php echo $monthname." ".$year; ?></span>
  </td>
  </tr>
  <tr style="background: #FFBC37;">
  <td style="text-align: center; width: 15px;"
        onmouseover="this.style.background= '#FECE6E'"
        onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;">Su</span>
  </td>
  <td style="text-align: center; width: 15px;"
        onmouseover="this.style.background='#FECE6E'"
        onmouseout="this.style.background='#FFBC37'">
    <span style="font-weight: bold;">M</span>
```

```
</td>
<td style="text-align: center; width: 15px;"
      onmouseover="this.style.background='FECE6E'"
      onmouseout="this.style.background='#FFBC37'">
  <span style="font-weight: bold;">Tu</span>
</td>
<td style="text-align: center; width: 15px;"
      onmouseover="this.style.background='#FECE6E'"
      onmouseout="this.style.background='#FFBC37'">
  <span style="font-weight: bold;">W</span>
</td>
<td style="text-align: center; width: 15px;"
      onmouseover="this.style.background='#FECE6E'"
      onmouseout="this.style.background='#FFBC37'">
  <span style="font-weight: bold;">Th</span>
</td>
<td style="text-align: center; width: 15px;"
      onmouseover="this.style.background='#FECE6E'"
      onmouseout="this.style.background='#FFBC37'">
  <span style="font-weight: bold;">F</span>
</td>
<td style="text-align: center; width: 15px;"
      onmouseover="this.style.background='#FECE6E'"
      onmouseout="this.style.background='#FFBC37'">
  <span style="font-weight: bold;">Sa</span>
</td>
</tr>
<?php
  $monthstart = date("w", $timestamp);
  $lastday = date("d", mktime (0, 0, 0, $month + 1, 0, $year));
  $startdate = -$monthstart;

  //计算出所需的行数。
  $numrows = ceil (((date("t",mktime (0, 0, 0, $month + 1, 0, $year))
                    + $monthstart) / 7));

//创建适当的行数……
for ($k = 1; $k <= $numrows; $k++){
  ?><tr><?php
  //使用7列(表示7天)……
  for ($i = 0; $i < 7; $i++){
    $startdate++;
      if (($startdate <= 0) || ($startdate > $lastday)){
        //如果日历中有一个空白格。
        ?><td style="background: #FFFFFF;"> </td><?php
```

```
        } else {
          if ($startdate == date("j") && $month == date("n") &&➥
$year == date("Y")){
            ?><td onclick="createform(event,'<?php echo $year . "-" . $month➥
. "-" .
$startdate; ?>')" style="text-align: center;➥
background: #FFBC37;" onmouseover="this.style.background='#FECE6E';➥
checkfortasks ('<?php ➥
echo $year . "-" . $month . "-" . $startdate; ?>',event);"➥
onmouseout="this.style.background='#FFBC37'; hidetask();">➥
<?php echo date ("j"); ?></td><?php
          } else {
            ?><td onclick="createform(event,'<?php echo $year . "-" . $month➥
. "-" . $startdate; ?>')" style="text-align: center;➥
background: #A2BAFA;" onmouseover="this.style.background=➥
'#CAD7F9'; checkfortasks ➥
('<?php echo $year . "-" . $month . "-" . $startdate; ?>',event);" ➥
onmouseout="this.style.background='#A2BAFA'; hidetask();">➥
<?php echo $startdate; ?></td><?php
          }
        }
      }
      ?></tr><?php
    }
    ?>
  </table>
```

这里的示例代码和第4章中的相比，主要的不同是在表格元素中使用onclick事件句柄来调用createform函数。大家还将发现在此传送了一个关联的日期字段，它可以存储在前面theform.php脚本中的hidden字段中。接下来看看业务逻辑，紧接着的程序块所示的函数将添加到functions.js文件中，它将修改createform函数，以使其能够传入这个日期值。另外还创建了一个新的JavaScript文件，名为xmlhttp.js，用来实现基本的Ajax功能。以下就是xmlhttp.js文件及这个新的createform函数（它位于functions.js文件中）的内容。

```
//xmlhttp.js

//创建XMLHttp对象的函数。
function getxmlhttp (){
  //创建一个布尔型变量，用来检查是存在有效的微软ActiveX实例。
  var xmlhttp = false;

  //检查使用的是否为IE。
  try {
    //如果javascript的版本在5.0以上。
```

```
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    //否则使用老版本的ActiveX对象。
  try {
      //如果使用的是IE。
      xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (E) {
      //否则肯定使用的是非IE浏览器。
      xmlhttp = false;
    }
  }

  // 如果不是使用IE，则创建一个
  // 该对象的JavaScript实例。
  if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
    xmlhttp = new XMLHttpRequest();
  }

  return xmlhttp;
}

//用来处理XMLHttpRequest的函数。
function processajax (serverPage, obj, getOrPost, str){
  //获取一个XMLHttpRequest对象。
  xmlhttp = getxmlhttp ();
  if (getOrPost == "get"){
    xmlhttp.open("GET", serverPage);
    xmlhttp.onreadystatechange = function() {
      if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        obj.innerHTML = xmlhttp.responseText;
      }
    }
      xmlhttp.send(null);
    } else {
      xmlhttp.open("POST", serverPage, true);
      xmlhttp.setRequestHeader("Content-Type",➥
"application/x-www-form-urlencoded; charset=UTF-8");
      xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
          obj.innerHTML = xmlhttp.responseText;
        }
      }
      xmlhttp.send(str);
    }
  }
```

```
//functions.js

function createform (e, thedate){

  theObject = document.getElementById("createtask");

  theObject.style.visibility = "visible";
  theObject.style.height = "200px";
  theObject.style.width = "200px";

  var posx = 0;
  var posy = 0;

  posx = e.clientX + document.body.scrollLeft;
  posy = e.clientY + document.body.scrollTop;

  theObject.style.left = posx + "px";
  theObject.style.top = posy + "px";

  //将载入该页面的位置。
  var objID = "createtask";
  var serverPage = "theform.php?thedate=" + thedate;

  var obj = document.getElementById(objID);
  processajax (serverPage, obj, "get", "");

}
```

如你所见，在createform函数中并没有多大变化。注意现在要传送的多了一个新字段，它用来表示要添加到任务中的日期信息。这个日期字段将通过查询字符串传给Ajax请求，然后载入到theform.php文件定义的表单中的hidden字段里。接下来的程序块（也是保存在functions.js文件中）将展示如何通过Ajax来提交表单。

```
//用来提交表单的函数。
function getformvalues (fobj, valfunc){

  var str = "";
  aok = true;
  var val;

  //遍历包含表单中所有对象的列表。
  for(var i = 0; i < fobj.elements.length; i++){
    if(valfunc) {
```

```
        if (aok == true){
          val = valfunc (fobj.elements[i].value,fobj.elements[i].name);
          if (val == false){
            aok = false;
          }
        }
      }
      str += fobj.elements[i].name + "=" + escape(fobj.elements[i].value) + "&";
    }
    //然后返回字符串的值。
    return str;
}

function submitform (theform, serverPage, objID, valfunc){
    var file = serverPage;
    var str = getformvalues(theform,valfunc);
    //如果验证是OK的。
    if (aok == true){
      obj = document.getElementById(objID);
      processajax (serverPage, obj, "post", str);
    }
}
```

这组代码的工作机制为：首先通过theform.php文件中submit按钮所定义的onclick事件句柄来调用submitform函数。submitform函数有四个参数：表单元素本身（theform），表示Ajax请求的发送目的地的serverPage（负责完成该处理的文件），用来载入请求结果的对象（objID），用来验证信息的函数引用（valfunc，可选）。总之，这与前面使用Ajax请求实现的函数区别不大。

不过在submitform函数中，将调用一个名为getformvalues的函数，它将返回一个字符串，包括了提交给表单的字段和值。getformvalues函数只需要传入的表单元素，因此将遍历所有表单元素以寻找已提交的任意字段。为了实现最大的控制（主要是数据验证，后面将讨论这个内容），将创建一个case语句，用来根据不同的字段类型进行相关的处理。通过这种方法来处理该值，能够以不同形式来处理不同的字段类型，这对于表单验证是十分有用的。

当getformvalues函数对表单元素进行遍历时，将收集字段的名称和值。当选择出完整的值和名称信息时，将组织成一个字符串返回给submitform函数以继续处理。

当submitform函数接收到最终的输入字符串后，将调用processajax函数完成最终的服务器请求。processajax函数的功能我们很熟悉，它将创建一个Ajax要使用的XMLHttpRequest对象（如果使用的是IE则是ActiveX对象），然后在open方法中载入表单请求。在open方法中将指定是使用GET还是POST请求，在这里选择的是POST方法。同时会注意到，

为了发送一个表单请求，将通过setRequestHeader方法来生成一个单独的参数，用来设置表单提交的类型。也是在这里，当需要传送文件时，必须在包含文件（将在第6章中详细讨论）中指定setRequestHeader。

最后一步是将str变量传给XMLHttpRequest的send方法。通过传递该字符串并发送请求，其值将提交给process_task.php文件，在此将触发一个服务器端请求。process_task.php文件的内容如代码清单5-3所示。

代码清单5-3　处理表单并在数据库中添加一条新记录（process_task.php）

```
<?php

  //process_task.php

  //创建一个数据库连接。
  require_once ("dbconnector.php");
  opendatabase();

  //准备输入数据库中的数据。
  $yourname = mysql_real_escape_string (strip_tags ($_POST['yourname']));
  $yourtask = mysql_real_escape_string (strip_tags ($_POST['yourtask']));
  $thedate = mysql_real_escape_string (strip_tags ($_POST['thedate']));

  //构建一个动态查询。
  $myquery = "INSERT INTO task (taskid, yourname, thedate, description) VALUES➥
  ('0','$yourname','$thedate','$yourtask')";

  //执行该查询（如果存在问题则发送一个错误消息）。
  if (!mysql_query ($myquery)){
  header ("Location: theform.php?message=There was a problem with the entry.");
  exit;
}

  //如果一切正常，则返回。
  header ("Location: theform.php?message=success");
?>
```

当通过PHP处理脚本向数据库添加信息时，有几个问题需要考虑，其中最重要的是允许哪类信息存放到数据库中。在本例中不将多余的空格或HTML代码添加到数据库中。因此对输入的数据将通过trim、addslashes和htmlspecialchars函数来进行预处理，以创建数据库所需要的数据集。

接下来是创建一个动态的INSERT查询，用来向数据库添加一个新记录。在本例中，将对前一章中创建的数据库表做少许修改，将userid字段改为一个名为yourname的TINYTEXT（数

据类型）字段。这样就可以让任何人都能够很容易地在任务数据库中添加一个任务信息。构建完该查询后，将使用mysql_query函数来执行该查询。如果执行失败，将返回错误信息。如果执行成功则返回到表单中，这时新任务也将添加完成。

由于数据库表结构发生了变化，autocomp.php也将做一些修改，改为从task表读取用户姓名信息，而不是从user表。新的代码如代码清单5-4所示。

代码清单5-4 弹出自动完成列表的代码（autocomp.php）

```
<?php

  //autocomp.php

  //添加数据库连接器。
  require_once ("dbconnector.php");
  //然后打开一个数据库连接。
  $db = opendatabase();

  $myquery = "SELECT DISTINCT(yourname) AS yourname FROM task WHERE➥
yourname LIKE LOWER('%" . mysql_real_escape_string($_GET['sstring']) . "%')➥
ORDER BY yourname ASC";

if ($userquery = mysql_query ($myquery)){
  if (mysql_num_rows ($userquery) > 0){
    ?>
    <div style="background: #CCCCCC; border-style: solid; border-width: 1px;➥
border-color: #000000;">
    <?php
      while ($userdata = mysql_fetch_array ($userquery)){
        ?><div style="padding: 4px; height: 14px;" onmouseover="➥
this.style.background
= '#EEEEEE'" onmouseout="this.style.background = '#CCCCCC'" ➥
onclick="setvalue ('<?php echo $userdata['yourname']; ?>')">➥
<?php echo $userdata['yourname']; ?></div><?php
        }
        ?>
      </div>
      <?php
    }
  } else {
    echo mysql_error();
  }

?>
```

现在autocomp.php将从task表中读取信息，你可以根据需要添加多条任务，该系统使得这个添加过程更加易用。其结果如图5-2所示，首先展现新用户（及新任务）添加之前，然后展现的是在新用户添加之后。

在新任务添加前……　　添加后

图5-2　通过基于Ajax的表单提交向数据库添加记录前后

5.3　表单验证

我认为表单验证（以及验证周期）是区别开发人员是否专业的标准之一。应用程序的运行应像其实现代码预期的一样，这有一部分因素是为了避免潜在的错误以及对这些问题的处理。在开发领域，对错误和非计划操作的处理都称为验证。

对输入信息的验证有两种途径：客户端和服务器。正如你所想，一个是由客户端语言（本例中的JavaScript）处理的，另一个则是由服务器端语言（在本例中就是PHP）实现的。我认为冗余不仅是有用的，而且还是很必要的。要开发出功能完善、不易崩溃的Web应用程序，就需要对用户提交的表单进行验证。如果用户看到bug或崩溃现象，那么就会对你的产品失去信任。如果用户对产品失去信任，就不会再用它。

请看本例。如果用户提交了他的姓名和任务，那么程序将一切正常，但如果未能这样做将会怎样？那么可能会使数据库产生空记录，这将导致系统产生潜在的问题。还记得前面说过JavaScript能够完成一些验证功能吗？现在就要使用这个结构了。首先来看看客户端的验证。

```
//functions.js
function trim (inputString) {
  // 去除传入的字符串头部及尾部的空白。同时还将
  // 删除连续的空白并将其替换为一个空白。如果除了
```

```
  // 传入一个字符串之外还包括其他内容（null值、自定义对象等）则返回输入值。
  if (typeof inputString != "string") { return inputString; }
  var retValue = inputString;
  var ch = retValue.substring(0, 1);
  while (ch == " ") { //检查字符串头部的空白。
    retValue = retValue.substring(1, retValue.length);
    ch = retValue.substring(0, 1);
  }
  ch = retValue.substring(retValue.length-1, retValue.length);
  while (ch == " ") { //检查字符串尾部的空白。
    retValue = retValue.substring(0, retValue.length-1);
    ch = retValue.substring(retValue.length-1, retValue.length);
  }
  while (retValue.indexOf(" ") != -1) {➥
  // 注意在该字符串中有两个空白。
  // 因此将在字符串中查找多个空白。
    retValue = retValue.substring(0, retValue.indexOf(" ")) +➥
  retValue.substring(retValue.indexOf(" ")+1, retValue.length);➥
  // 同样，在每个字符串有两个空白。
  }
  return retValue; //将剪裁过的字符串返回给用户。
} //"trim"函数的结束。
```

首先要介绍的新函数是trim。对该函数没有过多地描述，它本身是相当复杂的。可以说trim函数能够完成服务器端所能完成的事，即在字符串的开头和结尾删除多余的空白字符。PHP提供了自己的函数程序库可以使用，但在JavaScript验证中，所有的代码都必须自己写。该函数的目标是对字符串进行检查，使其不会充斥着空白字符。

```
//用来验证addtask表单的函数。
function validatetask (thevalue, thename){

  var nowcont = true;

  if (thename == "yourname"){
    if (trim (thevalue) == ""){
      document.getElementById("themessage").innerHTML = ➥
"You must enter your name.";
      document.getElementById("newtask").yourname.focus();
      nowcont = false;
    }
  }
  if (nowcont == true){
    if (thename == "yourtask"){
      if (trim (thevalue) == ""){
```

```
            document.getElementById("themessage").innerHTML = ➥
"You must enter a task.";
          document.getElementById("newtask").yourtask.focus();
          nowcont = false;
          }
        }
      }

      return nowcont;
}
```

该函数将被遍历表单元素的getformvalues函数调用。它将对你想验证的字段进行验证（通过thename值），然后将确保这些字段是空的（通过thevalue元素）。如果该字段为空，该函数将返回一个false值，然后将把焦点指向这个空的表单元素。

```
var aok;

//用来提供一个表单的函数。
function getformvalues (fobj, valfunc){

  var str = "";
  aok = true;
  var val;
  //遍历包含表单中所有对象的列表。
  for(var i = 0; i < fobj.elements.length; i++){
    if(valfunc) {
      if (aok == true){
        val = valfunc (fobj.elements[i].value,fobj.elements[i].name);
        if (val == false){
          aok = false;
        }
      }
    }
    str += fobj.elements[i].name + "=" + escape(fobj.elements[i].value) + "&";
  }
  //然后返回字符串的值。
  return str;
}
```

如你所见，getformvalues函数最主要的修改是添加了验证功能。首先向该脚本传递了一个valfunc函数，用来对输入的信息进行验证（本例中将使用验证脚本validatetask）。然后对每种要验证的值类型（在本例中是text和textarea的值）调用该验证函数，然后传入要使用的名称和值。如果对任何一种类型返回false值，那么说明该表单未提交。该系统使用变量aok来确定是否发送了XMLHttpRequest请求。如果其值设置为false，那么就说明存在验证错误，

并且在脚本继续执行之前必须修正这个问题。

```
function submitform (theform, serverPage, objID, valfunc){
  var file = serverPage;
  var str = getformvalues(theform,valfunc);
  //如果验证是ok的。
  if (aok == true){
    obj = document.getElementById(objID);
    processajax (serverPage, obj, "post", str);
  }
}
```

而对于submitform的修改则是不言自明的。现在submitform函数将获取valfunc变量作为参数（由代码清单5-5所示的theform.php文件中的onclick事件句柄传入），然后将把它传给getformvalues函数。而processajax函数现在只有当aok变量设置为true时才会向服务器发送该请求（使验证一直有效，直到表单完成为止）。

代码清单5-5 当日历中的某个日期被单击时将展示的脚本（theform.php）的修订版本

```
<?php
  //theform.php
?>
  <div style="padding: 10px;">
    <div id="themessage">
      <?php
        if (isset ($_GET['message'])){
          echo $_GET['message'];
        }
      ?>
</div>
  <form action="process_task.php" method="post" id="newtask" name="newtask">
    Your Name<br />
    <input name="yourname" id="yourname" style="width: 150px; height: 16px;"➥
type="text" value="" onkeypress="autocomplete(this.value, event)" /><br />
    Your Task<br />
    <textarea style="height: 80px;" name="yourtask" id="yourtask">➥
</textarea><br />
    <input type="hidden" name="thedate" value="<?php echo $_GET['thedate']; ?>" />
    <input type="button" value="Submit" onclick="submitform➥
(document.getElementById('newtask'),'process_task.php','createtask', ➥
validatetask); return false;" />
    <div align="right"><a href="javascript:closetask()">close</a></div>
  </form>
</div>
```

对于theform.php文件而言，唯一的改变就是在调用submitform函数时要传入validatetask函数名。这样就能够通过指定要使用的验证脚本来移植submitform函数。

现在客户端验证已经完成，接下来看看以PHP服务器端脚本的形式实现的冗余验证，其内容如代码清单5-6所示。

代码清单5-6　任务提交脚本（process_task.php）的修订版

```
<?php

  //process_task.php

  //创建一个到数据库的连接。
  require_once ("dbconnector.php");
  opendatabase();

  //验证。
  if (trim ($_POST['yourname']) == ""){
    header ("Location: theform.php?message=Please enter your name.");
    exit;
  }
  if (trim ($_POST['yourtask']) == ""){
    header ("Location: theform.php?message=Please enter a task.");
    exit;
  }

  //准备将填充到数据库的数据。
  $yourname = mysql_real_escape_string (strip_tags ($_POST['yourname']));
  $yourtask = mysql_real_escape_string (strip_tags ($_POST['yourtask']));
  $thedate = mysql_real_escape_string (strip_tags ($_POST['thedate']));

  //构建一个动态的查询。
  $myquery = "INSERT INTO task (taskid, yourname, thedate, description) VALUES➥
('0','$yourname','$thedate','$yourtask')";

  //执行该查询（如果出现问题则发送一个错误消息）。
  if (!mysql_query ($myquery)){
    header ("Location: theform.php?message=There was a problem with the entry.");
    exit;
  }

  //如果一切正常则返回。
  header ("Location: theform.php?message=success");
?>
```

从服务器端视角来看，验证是一件很简单的事，因为PHP这样的编程语言已经有许多函数可供选择（而在JavaScript中则必须引入这些函数）。注意验证语句通常是在进入脚本基本部分之前工作的。你将检查是否为非空字符串（通常由trim函数实现），如果没有提供任何信息则向表单返回一个错误信息。当存在问题时，exit函数将中止脚本的执行，用户可以继续填写该表单。

如你所见，验证会带来更多的工作，但用它完成的脚本很健壮，并且用户也能够最大程度地发挥潜力（如图5-3所示）。

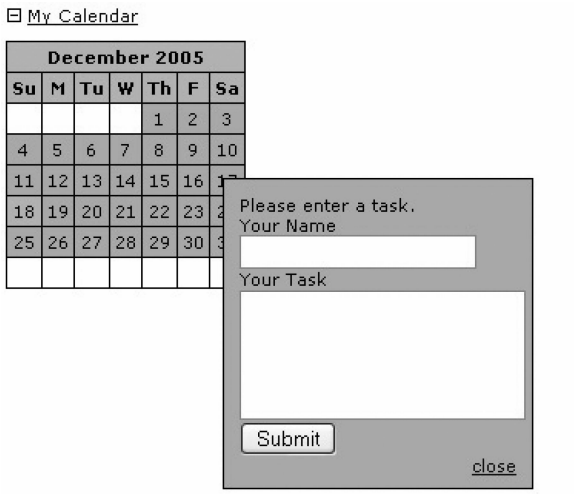

图5-3 验证：真正的开发者之友

5.4 小结

现在，Ajax的另一部分难题已经呈现出来了。随着你继续阅读本书，将逐渐建立核心概念。现在你已经了解了表单提交、动态服务器请求和受限的客户端JavaScript等相关内容，具备了可以用来实现某些有价值函数的大量基础知识了。

向用户提供一种同时与客户端技术和服务器端技术交互的方法，并确认数据已经实现了相互传递，这就打开了一道通往更多有趣、高级的Ajax方法的大门。在开始构建一些吸引人的应用程序之前，还需要讨论一组功能：图像。

第 6 章

图　像

6

当浏览某个网站时速率很低，等待图像载入的漫长过程令你非常恼火。基于文本的网站在任何因特网连接中都能够马上显示（或是看上去），而图像必须下载才能够查看。随着宽带网的出现，渐渐地这些都不再是问题，但图像显示仍然需要时间。图像的载入和查看是必需的，因此开发人员的一个主要任务就是使图像的载入越来越快。

幸运的是，有了诸如Ajax的概念和PHP之类的脚本语言，就拥有一个处理图像的强大工具集。通过Ajax可以动态地载入和显示图像，不必重载页面的其他部分，从而提高了处理速度，同时也能够对用户在屏幕或图像载入时能够看到的内容进行更多的控制。只要让用户知道正在发生什么，通常也能够忍受载入所需的时间。通过Ajax和一些PHP技巧，就能够让用户体验尽可能的流畅和愉快。

本章将介绍使用PHP和Ajax实现图像上传、维护和动态显示的方法。

6.1　图像上传

这里有必要指出一个对于Ajax而言的坏消息，文件上传的处理是无法通过`XMLHttpRequest`对象实现的。其原因是JavaScript无法访问计算机中的文件。虽然这稍稍有点让人失望，但仍然有办法来执行类似Ajax的功能，而无需使用`XMLHttpRequest`对象。聪明的开发人员会发现可以使用隐藏`iframe`来提供表单请求，因此这样实现文件上传也无需对整个页面进行刷新（尽管可能会看到屏幕有些闪动）。

只要把`iframe`的CSS属性`display`设置成`none`，该元素就能够在上传表单中使用，但对于最终用户却是不可见的。通过为`iframe`标签赋予一个`name`属性，就可以使用`form`标签中的`target`属性来将请求传送给这个隐藏`iframe`。当配置完这个`iframe`后，就能够完成任何所需的上传操作，然后再使用Ajax来执行其他的功能。看看下面的例子，它能够将一个图像上传到指定的目录中。其代码如代码清单6-1所示，它可以创建出如图6-1所示的应用程序。

图6-1　使用了Ajax的文件上传系统，它将通过隐藏`iframe`来隐藏上传元素

代码清单6-1 用于创建一个包含隐藏iframe的表单的代码（sample6_1.hmtl）

```
<!-- sample6_1.html -->
<!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">
<head>
<title>Sample 6_1</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="style.css" />
<script type="text/javascript" src="xmlhttp.js"></script>
<script type="text/javascript" src="functions.js"></script>
</head>
<body>
  <div id="showimg"></div>
  <form id="uploadform" action="process_upload.php" method="post"➥
enctype="multipart/form-data" target="uploadframe"➥
onsubmit="uploadimg(this); return false">
    Upload a File:<br />
    <input type="file" id="myfile" name="myfile" />
    <input type="submit" value="Submit" />
    <iframe id="uploadframe" name="uploadframe" src="process_upload.php"➥
  class="noshow"></iframe>
    </form>
</body>
</html>
```

代码清单6-1创建了应用程序的基础和用户界面。在此包括表单（带有file元素）以及负责传送请求的iframe。注意在该HTML文档的head标签中有一个noshow类，这是用来隐藏iframe的。

为了完成实际的上传操作，需要使用一些实现Ajax的JavaScript代码。该JavaScript将执行functions.js文件中名为uploadimg函数所实现的上传操作。在提交按钮被单击时将调用该函数。

```
//functions.js
function uploadimg (theform){
  //提交该表单。
  theform.submit();
}
```

现在该文件只有一个函数（uploading），它只是用来提交表单。不过随着本章不断对该例子进行开发，它将成为构建整个Ajax结构的重要元件。当表单提交时，如下所示的PHP文件（将载入到iframe中）将负责处理实际的文件上传操作。请看代码清单6-2所示的PHP脚本。

代码清单6-2　上传图像所需的PHP代码（process_upload.php）

```
<?php

  //process_upload.php

  //允许的MIME文件类型。
  $allowedtypes = array ("image/jpeg","image/pjpeg","image/png","image/gif");
  //文件想保存的位置。
  $savefolder = "images";

  //如果是有效的文件
  if (isset ($_FILES['myfile'])){
    //然后需要确认是所希望的文件类型。
    if (in_array ($_FILES['myfile']['type'], $allowedtypes)){
      //接着可以执行复制操作。
    if ($_FILES['myfile']['error'] == 0){
      $thefile = $savefolder . "/" . $_FILES['myfile']['name'];
      if (!move_uploaded_file ($_FILES['myfile']['tmp_name'], $thefile)){
        echo "There was an error uploading the file.";
      } else {
        //向文档的父结点发信号，让其载入该图像。
        ?>
        <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"➥
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
          <html xmlns="http://www.w3.org/1999/xhtml">
          <head>
          <script type="text/javascript" src="functions.js"></script>
          </head>
          <body onload="doneloading (parent,'<?=$thefile?>')">
            <img src="<?=$thefile?>" />
          </body>
          </html>
          <?php
        }
      }
    }
  }
?>
```

在这段PHP代码中，首先要创建两个变量，用来确定希望上传的文件类型以及保存文件的位置。$allowedtypes数组中存放的是一个所允许上传的MIME类型列表。一个文件的MIME类型是一个字符串，用来表示文件所包含的数据类型。在本例中，只允许JPEG、GIF和PNG三种图像类型。

要将上传的图像保存到Web服务器的一个文件夹中，也就意味着必须在Web服务器上有一个可写的目录。代码清单6-2将上传目录指定为images（由$savefolder变量指定）。要使Web服务器上的文件夹可写，可以使用FTP客户端，如果有命令行访问的权限还可以使用chmod命令（chmod 777 /path/to/images）。

将上传的文件写到目标文件夹中，要使用move_uploaded_file函数。这个PHP函数将获取该图像并将其移动到指定的位置。另外，它确保该文件确实是由该脚本上传的。如果出现任何错误将返回false值，因此通过代码对其进行监控并采取相应的措施是很重要的。如果一切正常，在你选择的文件夹中就会有一个新上传的漂亮图片，而用户几乎看不到处理过程。通过使用onload事件能够触发一个JavaScript函数来传送已上传到父帧（执行上传操作的那个帧）中的文件名。这个onload事件是很有用的，因为它可以用来确定图像什么时候上传结束。下一节将讲述如何显示已上传的图像。

6.2 图像显示

当图像上传到服务器后，接下来就是显示它。这可以在图像上传完成后通过一个Ajax请求来实现。看看下面这些将添加到xmlhttp.js（代码清单6-3）和functions.js（代码清单6-4）脚本中的函数。

代码清单6-3 执行Ajax请求所需的JavaScript代码（xmlhttp.js）

```
//xmlhttp.js

//用来创建一个XMLHttp对象的函数。
function getxmlhttp (){
  //创建一个布尔型变量，用来检查是否存在有效的ActiveX实例。
  var xmlhttp = false;

  //检查使用的是否是IE浏览器。
  try {
    //如果JavaScript的版本大于5.0。
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    //如果不是，则使用老版本的ActiveX对象。
    try {
      //如果使用的是IE浏览器。
      xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (E) {
      //那么使用的肯定是非IE浏览器。
      xmlhttp = false;
    }
  }
```

```
  //如果使用的不是IE，则创建一个该对象的JavaScript实例。
  if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
    xmlhttp = new XMLHttpRequest();
  }
  return xmlhttp;
}

//处理XMLHttpRequest的函数。
function processajax (obj, serverPage){
  //获取一个XMLHttpRequest对象。
  var theimg;
  xmlhttp = getxmlhttp ();
  xmlhttp.open("GET", serverPage);
  xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      document.getElementById(obj).innerHTML = xmlhttp.responseText;
    }
  }
  xmlhttp.send(null);
}
```

代码清单6-4　载入已上传图像所需的JavaScript代码（functions.js）

```
//functions.js

//用来确定process_upload.php文件何时执行完的函数。
function doneloading(theframe,thefile){
  var theloc = "showimg.php?thefile=" + thefile
  theframe.processajax ("showimg",theloc);
}
```

如你所见，在此可以使用前几章中已经出现的某些功能函数，通过它们可以动态地将刚上传的图像显示到Web页面中，而无需刷新屏幕。uploadimg函数仍然将执行表单提交操作，但现在还将调用一个名为doneuploading的函数，它将在process_upload.php脚本完成图像上传时（根据onload事件判断）启动。doneuploading函数将接受两个参数，一个是隐藏iframe的父帧，另一个是上传图像的文件名。然后使用Ajax动态地将图像载入到父帧中的指定元素中。

代码清单6-5展示了showimg.php文件如何接收文件名并显示该图像。

代码清单6-5　显示指定文件名的图像所需的PHP代码（showimg.php）

```
<?php
    //showimg.php
```

```
    $file = $_GET['thefile'];

    //检查图像是否存在。
    if (!is_file($file) || !file_exists($file))
        exit;
?>
<img src="<?= $file ?>" alt="" />
```

showimg.php文件负责显示已上传的文件。它首先要接收到相应的文件名，该文件是刚通过基于Ajax实现的文件上传功能上传的。functions.js中的doneloading函数将负责把该文件名传给showimg.php文件（通过Ajax）。然后showimg.php将检查传入的文件名是否有效（通过is_file和file_exists函数）。如果找到有效的文件，该脚本将显示它，如图6-2所示。

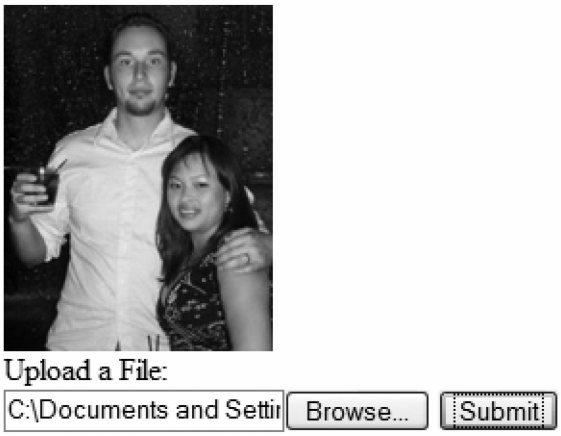

图6-2 呵呵，有了显示功能就更棒了

6.3 图像载入

很不幸，虽然脚本知道图像还在载入中，但用户对正在发生的事却是没有概念的。这时通过Ajax就可以提示用户正在发生什么。虽然第一次看到“Loading...”（载入中）这样的信息是在Google的Gmail中，不过后来在许多其他Ajax驱动的应用程序中也常常看到。通过使用innerHTML属性在showimg.php脚本执行其功能时显示一个载入提示信息，这是很简单的。看看代码清单6-6，它列出了uploadimg函数现在的代码，此时添加了一个对setStaus的调用，该函数用来在你所选择的HTML元素中写入状态信息。

代码清单6-6 对uploadimg函数的修改（functions.js）

```
function uploadimg (theform){
  //提交该表单。
  theform.submit();
```

```
  //然后向用户显示一个载入提示信息。
  setStatus ("Loading...","showimg");
}
//用来设置载入状态的函数。
function setStatus (theStatus, theObj){
  obj = document.getElementById(theObj);
  if (obj){
    obj.innerHTML = "<div class=\"bold\">" + theStatus + "</div>";
  }
}
```

在此创建了一个名为setStatus的函数，它有两个参数：一个是要显示的信息，另一个是要显示该信息的元素。使用该函数就能够为用户提供一个实时的提示信息。编写Ajax应用程序的全部目标就是让用户能知道发生了什么。现在当载入一个图像时，在脚本处理完成之前会看到类似于图6-3所示的载入提示信息。

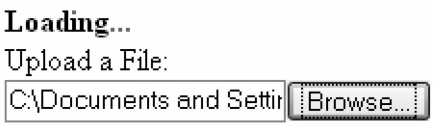

图6-3　载入中，载入中，载入中；这些文件一直在载入中

6.4　略缩图动态生成

自动生成略缩图对于任何网站而言都是很好的功能。当创建诸如内容管理系统、照片分类管理等软件时这将特别有用。对于图像大小调整，PHP提供了许多工具，但载入时间和略缩图生成所需的刷新始终是问题。在接下来的例子中将组合Ajax和PHP创建一个略缩图生成机制，它提供了文件上传功能，并且可以让用户实时调整图像的大小。看看在代码清单6-7中对showimg.php文件做了哪些修改。

代码清单6-7　为略缩图生成而做的修改（showimg.php）

```
<?php
    //showimg.php

    $file = $_GET['thefile'];

    //检查图像是否存在。
    if (!is_file($file) || !file_exists($file))
        exit;
?>
<img src="<?= $file ?>" alt="" />
<p>
```

```
    Change Image Size:
    <a href="thumb.php?img=<?= $file ?>&amp;sml=s"
      onclick="changesize('<?= $file ?>','s'); return false;">Small</a>

    <a href="thumb.php?img=<?= $file ?>&amp;sml=m"
      onclick="changesize('<?= $file ?>','m'); return false;">Medium</a>

    <a href="thumb.php?img=<?= $file ?>&amp;sml=l"
      onclick="changesize('<?= $file ?>','l'); return false;">Large</a>
</p>
```

这段代码在输出的图像下面添加了一个简单的菜单，通过该菜单可以以三种不同的大小来显示图像。每个链接都将调用changesize函数，该函数的参数是图像路径和指定的大小。当链接被单击时将调用changesize函数，然后根据请求中指定的大小创建一个当前图像的略缩图，然后通过Ajax动态载入该图像。changesize函数如代码清单6-8所示。

代码清单6-8 通过Ajax调用略缩图生成脚本的函数

```
function changesize (img, sml){
  //然后向用户显示一个载入提示信息。
  theobj = document.getElementById("showimg");
  if (theobj){
    setStatus ("Loading...","showimg");
    var loc = "thumb.php?img=" + img + "&sml=" + sml;
    processajax ("showimg",loc);
  }
}
```

此时还将应用到前面例子中所使用的功能，让用户知道正在载入一个新图像。当Ajax请求处理完成时，这个载入提示信息就将消失。changesize函数仅是向服务器发送一个Ajax请求，并在名为showimg的div元素中载入thumb.php文件。thumb.php的代码如代码清单6-9所示，它将用来生成略缩图并在屏幕上显示它。

代码清单6-9 根据Ajax传入的图像名称创建相应略缩图的PHP代码（thumb.php）

```
<?php

    //thumb.php

    function setWidthHeight($width, $height, $maxWidth, $maxHeight)
    {
        $ret = array($width, $height);
        $ratio = $width / $height;
        if ($width > $maxWidth || $height > $maxHeight) {
            $ret[0] = $maxWidth;
```

```
            $ret[1] = $ret[0] / $ratio;

            if ($ret[1] > $maxHeight) {
              $ret[1] = $maxHeight;
              $ret[0] = $ret[1] * $ratio;
            }
          }
          return $ret;
    }

//修改图像大小的函数。
function createthumb($img, $size = "s")
{
    //首先检查是否为有效文件。
    if (is_file($img)) {

        //获取当前文件大小。
        if ($cursize = getimagesize ($img)) {

            //然后根据xml变量寻找所需的新大小。
            $sizes = array("s" => 100, "m" => 300, "l" => 600);
            if (!array_key_exists($size, $sizes))
                $size = "s";

                $newsize = setWidthHeight($cursize[0],
                                          $cursize[1],
                                          $sizes[$size],
                                          $sizes[$size]);

                //现在已经有了大小约束，接下来是文件类型。
                $thepath = pathinfo ($img);

                //设置略缩图。
                $dst = imagecreatetruecolor ($newsize[0],$newsize[1]);

                //生成一个文件名。
                $filename = str_replace (".".$thepath['extension'], "", $img);
                $filename = $filename . "_th" . $size . "." . $thepath['extension'];

                $types = array('jpg' => array('imagecreatefromjpeg', 'imagejpeg'),
                            'jpeg' => array('imagecreatefromjpeg', 'imagejpeg'),
                            'gif' => array('imagecreatefromgif', 'imagegif'),
                            'png' => array('imagecreatefrompng', 'imagepng'));

                $func = $types[$thepath['extension']][0];
```

```
                $src = $func($img);

                //创建一个拷贝。
                imagecopyresampled($dst, $src, 0, 0, 0, 0,
                                $newsize[0], $newsize[1],
                                $cursize[0], $cursize[1]);
                //创建略缩图。
                $func = $types[$thepath['extension']][1];
                $func($dst, $filename);
?>
    <img src="<?= $filename ?>" alt="" />
    <p>
        Change Image Size:
        <a href="thumb.php?img=<?=$img?>&amp;sml=s"
          onclick="changesize('<?=$img?>','s'); return false;">Small</a>
        <a href="thumb.php?img=<?=$img?>&amp;sml=m"
          onclick="changesize('<?=$img?>','m'); return false;">Medium</a>
        <a href="thumb.php?img=<?=$img?>&amp;sml=l"
          onclick="changesize('<?=$img?>','l'); return false;">Large</a>
    </p>
<?php
              return;
            }
        }

      echo "No image found.";
  }

  createthumb($_GET['img'], $_GET['sml']);
?>
```

在thumb.php文件中首先要注意的函数是setWidthHeight。该函数唯一的用途是按比例获取一个合适的图像大小。换句话说，它将接受的参数是图像的宽度和高度，或是最大宽度和最大高度，然后基于传入的参数返回按比例缩放的宽度及高度值。

接下来的一个函数是createthumb，它会稍稍复杂些。createthumb函数的参数是图像路径以及大小，它将确定要创建的图像类型。这个函数拥有一个约束集，以基于函数最前面的small、med和large变量来创建相应的略缩图。接着将尝试寻找指定的图像路径。如果找到，则计算出新的大小参数（通过调用setWidthHeight函数），然后根据处理的图像是JPEG、GIF还是PNG来调用相应的图像创建函数。这是通过一个数组来实现的，该数组的内容是各种图像类型以及用来读写此类图像的GD函数。

当略缩图创建成功之后，该脚本将输出这个新创建的略缩图，然后像前面一样显示同样的导

航按钮，使用户可以根据需要创建不同大小的新略缩图。

这些函数可以无缝地整合，从上传新图像到图像大小动态调整等所有的事都能够迅速高效地实现，而且具有很好的人性化、很少的页面刷新。对于桌面应用程序而言，这样的功能已实现多年了，现在可以很高兴地说Web也具有同样优秀的交互性了。看看图6-4所展现的功能。

图6-4　动态图像调整——多好的概念！

6.5　小结

现在，在介绍完图像之后，对应用Ajax和PHP的HTML基本元素的介绍也就告一段落了。在本章中，学习了如何用一种全新的形式来处理图像。通过使用PHP的高级脚本功能和Ajax的新文件上传概念，就能够创建出一些更高级的基于图像的Web应用程序。

使用JavaScript和它的`XMLHttpRequest`对象，可以在需要时通过把服务器调用载入到Web页面中来实现所需的功能。但从用户的角度考虑易用性始终是很重要的，因此有时添加类似“载入中……”这样的消息就能够使用户体验获得很大提升。

现在你已经有了一些基础知识，可以开始研究一些更高级的Ajax和PHP概念。我始终坚信研究实际范例并真正使用它是学习的最佳途径。带着这种观念我们将继续下一章，它讲述了开发基于Ajax和PHP的网络应用程序所需的概念。