第 14 章　PHP 与 AJAX

AJAX 技术是近年流行起来的技术。它是现有的多种技术的综合应用，包括 JavaScript 技术、XML 技术、DOM 技术和 XMLHttpRequest 控件等多种技术。AJAX 技术是现在流行的 Web 2.0 的一个重要标志。本章主要内容如下。

- AJAX 原理。
- AJAX 的核心技术。
- AJAX 的相关应用。

通过对本章的学习，读者将能够理解什么是 AJAX，其工作原理是什么，并且能够在实际的应用中使用 AJAX。

14.1　AJAX 概述

AJAX 技术是当前 Web 技术中一门炙手可热的新兴技术。实质上，它并非一门新技术，而是几种新技术的强大组合，能够为用户提供更为自然的浏览体验。

14.1.1　什么是 AJAX

传统 B/S 模式应用程序的弊端是对用户反应不灵敏，用户总是处于“提交=>等待=>响应”的过程中，也就是说，“用户的动作总是和服务器的思考时间同步”。而 AJAX 提供了与服务器异步通信的能力，从根本上让用户从“请求=>等待=>响应”的循环中解脱出来。

AJAX 是 Asynchronous JavaScript and XML（及 DHTML 等）的缩写，翻译成中文为异步 JavaScript 和 XML。这个短语是 Adaptive Path 的 Jesse James Garrett 发明的，按照 Jesse 的解释，这不是个首字母缩写词，AJAX 技术包含以下内容。

- 基于 CSS 标准的表示。
- 使用 Document Object Model 进行动态显示和交互。
- 使用 XMLHttpRequest 与服务器进行异步通信。
- 使用 JavaScript 绑定一切。

AJAX 是现有多种技术的综合应用。它是通过浏览器页面与服务器后台处理的异步处理来减少网络传输，进而减少用户的等待时间和服务器负担的一种综合技术。它由 HTML、JavaScript 技术、DHTML 和 DOM 组成的，这一方法可以将传统笨拙的 Web 界面转化成交互性的 AJAX 应用程序。

14.1.2 工作原理

AJAX 的核心是 XMLHttpRequest。XMLHttpRequest 提供了与服务器异步通信的能力，从而让用户在请求的时候不受到阻塞。其工作流程如下所述。

首先由用户在客户端浏览器页面触发某个事件，如 onclick 事件。当然，这个事件只能是被 JavaScript 脚本语言捕获到的事件。然后 JavaScript 脚本相应的创建一个 XMLHttpRequest 请求，并通过 XMLHttpRequest 异步地把请求发送到服务器端，同时等待服务器端的响应。服务器端程序在接受客户端所提交的请求后，进行处理并把结果返回。返回的结果被 XMLHttpRequest 捕获到并返回给 JavaScript，再由 JavaScript 调用相应的 DOM 进行显示层的控制。AJAX 工作原理图如图 14-1 所示。

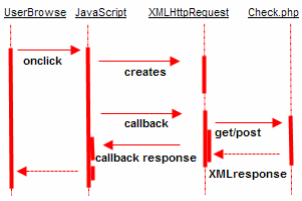

图 14-1 AJAX 工作原理图

要让 AJAX 能够成功运行，需要用户的客户端与服务端两种技术互相配合。客户端采用 JavaScript 引擎向服务器端发送请求，服务端根据客户端提交的请求返回指定内容。

用户客户端的 XMLHttpRequest 对象使 JavaScript 以异步方式访问服务器成为可能，这样当 JavaScript 函数在后台向服务器提交请求时，用户可以在客户端页面继续其他的工作。

14.1.3 优点与缺点

AJAX 技术作为近年来流行的技术，有其技术优点，也有其缺点。优点主要有以下几个方面。

- AJAX 可以提高系统性能，优化用户界面，增强用户浏览体验。
- AJAX 允许在不更新整个页面的前提下维护数据，这使得 Web 应用程序更为迅捷地回应用户的动作，并避免了在网络上发送那些没有改变过的信息。
- AJAX 应用可以仅向服务器发送并取回必需的数据，服务器与客户端之间交换的数据大大减少。
- AJAX 不需要任何浏览器插件，但需要用户允许 JavaScript 在浏览器上执行。
- AJAX 允许将一些服务端的操作移至客户端，减轻了服务器的负担。

其缺点主要有以下几个方面。

- AJAX 可以在不刷新页面的情况下更新页面数据，这可能破坏浏览器【后退】按钮的正常行为。
- AJAX 在客户端执行，因此在开发时需考虑到 AJAX 的兼容性。
- AJAX 对串流媒体的支持没有 FLASH、Java Applet 好。
- 由于 AJAX 的代码存放在页面的 HTML 语言中，因此项目代码可能因此泄漏。
- 使用 AJAX 动态更新页面使得用户难于将某个特定的状态保存到收藏夹中。
- AJAX 的无刷新重载使得页面的变化没有刷新重载整个页面那么明显，所以容易给用户带来困扰。
- 用 JavaScript 作 AJAX 引擎，不得不考虑 JavaScript 的兼容性和 DeBug 等问题。
- 现有的好多手持设备（如 PDA、手机等）不能很好地支持 AJAX。

14.2　使用 AJAX

AJAX 不是指一种单一的技术，而是有机地利用了一系列相关的技术。它由用户的客户端通过 JavaScript 的 XMLHttpRequest 对象向服务器发起请求，再由服务器端根据用户的请求进行处理，完成以后向客户端抛出处理结果，用户浏览器再将服务器处理结果，通过 DOM 结构呈现在 HTML 页面上。本节将根据 AJAX 的请求过程一步步进行讲解。

14.2.1　创建 XMLHttpRequest 对象

AJAX 是通过 XMLHttpRequest 对象向服务器发起请求的。该对象最初是由 Microsoft 的 Internet Explorer 以 ActiveX 对象引入，被称为 XMLHTTP。而后 Mozilla、Netscape、Safari 和其他浏览器也提供了 XMLHttpRequest 类，只不过它们创建 XMLHttpRequest 类的方法不同。

对于 Microsoft 的 Interner Explorer 浏览器，其版本不同，创建 XMLHttpRequest 对象的方法也不一样。通常有以下几种方法。

```
xmlhttp_request = new ActiveXObject("Msxml2.XMLHTTP.3.0");      //IE3.0、IE4.0 或 5.0
xmlhttp_request = new ActiveXObject("Msxml2.XMLHTTP");          //IE6.0
xmlhttp_request = new ActiveXObject("Microsoft.XMLHTTP");       //IE7.0 及以上
```

其中，第 1 行是在 IE 6.0 之前的版本创建该对象时所使用的方法，第 2 行为 IE 6.0 版本创建该对象时所使用的方法，第 3 行为 IE 6.0 版本以上的浏览器创建对象时所使用的方法。为了能够更好的兼容不同版本的 Internet Explorer 浏览器，在实际应用中常常需要根据浏览器版本创建 XMLHttpRequest 类。

对于 Mozilla、Netscape、Safari 等浏览器，创建 XMLHttpRequest 类的方法如下所示。

```
xmlhttp_request = new XMLHttpRequest();
```

如果服务器的响应没有 XML mime-type header，某些版本的 Mozilla 浏览器可能无法正常工作。为了解决这个问题，如果服务器响应的 header 不是 text/xml，可以调用如下所示的其他方法修改该 header。

```
xmlhttp_request = new XMLHttpRequest();
xmlhttp_request.overrideMimeType('text/xml');
```

在实际的应用中，AJAX 编写在静态的 HTML 页面中的 Script 标签以内，也可以编写在独立的 JavaScript 文件中，在需要使用 AJAX 的页面加载该 JavaScript 文件即可。

【示例 14-1】如何在 HTML 页面中创建兼容不同浏览器的 XMLHttpRequest 对象。代码如下所示。

```
<?xml version="1.0" encoding="GB2312" ?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>Insert title here</title>
<script type="text/javascript">
```

```
var xmlhttp_request = false;                          //定义变量
try{
    if( window.ActiveXObject ){                       //判断浏览器是否是 ActiveXObject 对象
        for( var i = 5; i; i-- ){                     //根据 IE 浏览器版本使用不同方法创建
            try{
                if( i == 2 ){
                    xmlhttp_request = new ActiveXObject( "Microsoft.XMLHTTP" );
                } else{
                    xmlhttp_request = new ActiveXObject( "Msxml2.XMLHTTP." + i + ".0" );
                    xmlhttp_request.setRequestHeader("Content-Type","text/xml");
                    xmlhttp_request.setRequestHeader("Charset","gb2312");
                }
                break;
            }catch(e){
                xmlhttp_request = false;
            }
        }
    }else if( window.XMLHttpRequest ){                //判断浏览器是否支持 XMLHttpRequest 对象
        xmlhttp_request = new XMLHttpRequest();
        if (xmlhttp_request.overrideMimeType){
            xmlhttp_request.overrideMimeType('text/xml');
        }
    }
}catch(e){
    xmlhttp_request = false;
}
</script>
</head>
<body>
</body>
</html>
```

分析：在上述程序中，首先定义了一个 JavaScript 的 xmlhttp_reques 变量，再判断用户当前浏览器是否支持 ActiveX 对象，并根据浏览器的版本支持使用不同的方法创建对象。若用户浏览器不支持 ActiveX 对象，则判断浏览器是否支持 XMLHttpRequest 对象，再使用 XMLHttpRequest 组件来创建对象。

一般来讲，IE 浏览器都支持 ActiveX 对象，都是采用 Microsoft XMLHTTP 组件来创建 XMLHttpRequest 对象。而 Mozilla、Netscape、Safari 等非 IE 系列的浏览器几乎都支持 XMLHttpRequest 组件，直接使用该组件来创建 XMLHttpRequest 对象。

14.2.2 向服务器发送请求

XMLHttpRequest 组件的一个特点就是可以不用刷新整个页面，直接与服务器进行交互。XMLHttpRequest 向服务器发送请求的方法如下所示。

```
xmlhttp_request.open(send_method, url, flag);
xmlhttp_request.send(null);
```

其中，xmlhttp_request 为 14.2.1 小节所创建的 XMLHttpRequest 对象；参数 send_method 为 HTTP 请求方式，可以为 GET、POST 或者服务器所支持的调用方式。按照 HTTP 规范，

该参数要求大写，否则可能某些浏览器（如 Firefox）无法处理该请求。

参数 url 为 XMLHttpRequest 对象向服务器发出请求的 URL。为了防止跨站点脚本攻击，XMLHttpRequest 对象所请求的 URL 必须使用与该页面相同的 HTTP 协议、Web 主机地址和端口。虽然有一部分浏览器允许使用任意的 URL，但是大部分的浏览器考虑安全因素不支持跨域请求。如果在实际应用中确实需要进行跨域请求，则必须在服务器端使用 cURL 或其他方法进行相关处理。

参数 flag 为设置请求是否为异步模式，即在等待 XMLHttpRequest 对象所请求的页面响应时是否可以继续执行页面代码。设为 true，可以继续执行页面 JavaScript 脚本而不用等待响应；设为 false，则需等待请求页面响应后，才能执行页面 JavaScript 脚本。

最后使用值 null 调用 XMLHttpRequest 对象的 send( )方法。因为一般已经在所请求的 URL 中添加了要发送给服务器的数据，所以此处不需要发送任何数据。这样就发出了请求，服务器将按照所发送的请求进行响应。

【示例 14-2】在 HTML 页面如何向服务提交请求。代码如下所示。

```
<?xml version="1.0" encoding="GB2312" ?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>Insert title here</title>
<script type="text/javascript">
var xmlhttp_request = false;                              //定义变量
try{
    if( window.ActiveXObject ){                           //判断浏览器是否支持 ActiveXObject 对象
        for( var i = 5; i; i-- ){                         //根据 IE 浏览器版本使用不同方法创建对象
            try{
                if( i == 2 ){
                    xmlhttp_request = new ActiveXObject( "Microsoft.XMLHTTP" );
                } else{
                    xmlhttp_request = new ActiveXObject( "Msxml2.XMLHTTP." + i + ".0" );
                    xmlhttp_request.setRequestHeader("Content-Type","text/xml");
                    xmlhttp_request.setRequestHeader("Charset","gb2312");
                }
                break;
            }catch(e){
                xmlhttp_request = false;
            }
        }
    }else if( window.XMLHttpRequest ){                    //判断浏览器是否支持 XMLHttpRequest 对象
        xmlhttp_request = new XMLHttpRequest();
        if (xmlhttp_request.overrideMimeType){
            xmlhttp_request.overrideMimeType('text/xml');
        }
    }
}catch(e){
    xmlhttp_request = false;
}
function chkuserid(txt){                                  //检测用户名
    var userid = txt.value;
```

```
        if(userid.length > 0){                                  //当用户输入用户名后向服务器发出请求
            var url="/chk.php?userid=" + userid;
            xmlhttp_request.open("GET", url, true);
            xmlhttp_request.send(null);
        }
}
</script>
</head>
<body>
<form action="">
<table>
<thead>注册</thead>
<tr><td> 用 户 名 ： </td><td><input type="text" value="" id="userid" name="userid" onblur=
"javascript:chkuserid(this);"/></td></tr>
<tr><td>密码：</td><td><input type="password" value="" id="password" name="password"/>
<tr><td> 再 次 输 入 密 码 ： </td><td><input type="password" value="" id="password2" name=
"password2"/>
<tr><td></td><td><input type="submit" value="提交" name="submit" id="submit" />
</table>
</form>
</body>
</html>
```

分析：在上述程序中，当用户在注册时输入用户名，从用户名输入框中移出光标后，将触发所定义的 JavaScript 函数 chkuser。该函数将调用浏览器创建的 XMLHttpRequest 对象 xmlhttp_request 向服务器发送请求。

14.2.3 处理服务器响应

在使用 XMLHttpRequest 对象向服务发出请求后，服务器根据用户浏览器发出的请求作出响应。面对服务器所作出的响应，需要告诉 XMLHttpRequest 对象用哪一个 JavaScript 函数来处理。可以将 XMLHttpRequest 对象的 onreadystatechange 属性设置为要处理响应的 JavaScript 的函数名，其语法格式如下所示。

```
xmlhttp_request.onreadystatechange =FunctionName;
```

其中，FunctionName 是用 JavaScript 创建的用于处理服务器响应的函数名。

注意：函数不能写成 FunctionName( )。

除了使用这种方式以外，还可以使用以下形式。

```
xmlhttp_request.onreadystatechange = function(){
// JavaScript 脚本
};
```

其中，直接将用于处理服务器响应的 JavaScript 脚本写在 onreadystatechange 后面的 function 函数体内。在处理时首先要检查请求的状态，只有当一个完整的服务器响应已经收到了，函数才可以处理该响应。XMLHttpRequest 对象提供了 readyState 属性来对服务器响应进行判断，该属性取值如下。

- 0：未初始化。

- 1：正在装载。
- 2：装载完毕。
- 3：交互中。
- 4：完成。

因此，只有当 readyState 属性值为 4 的时候，才收到一个完整的服务器响应，JavaScript 函数才会处理该响应，如下所示。

```
If(xmlhttp_request.readyState==4){
//收到完整服务器响应时处理响应
}else {
//未收到完整服务器响应时进行其他处理或不处理
}
```

当收到一个完整的服务器响应后，还要去检查 HTTP 服务器响应的状态信息，只有当 HTTP 服务器响应的状态值为 200 时才表示正常。关于 HTTP 服务器响应的状态信息，不在本书讨论范围以内，读者可参考相关文档。

对于服务器响应的数据，可以使用以下两种方式得到。

- 以文本字符串的方式返回服务器的响应。
- 以 XMLDocument 对象方式返回响应。

【示例 14-3】在注册时如何处理服务器端的响应。代码如下所示。

```
<?xml version="1.0" encoding="GB2312" ?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>Insert title here</title>
<script type="text/javascript">
var xmlhttp_request = false;                              //定义变量
try{
    if( window.ActiveXObject ){                           //判断浏览器是否支持 ActiveXObject 对象
        for( var i = 5; i; i-- ){                         //根据 IE 浏览器版本使用不同方法创建对象
            try{
                if( i == 2 ){
                    xmlhttp_request = new ActiveXObject( "Microsoft.XMLHTTP" );
                } else{
                    xmlhttp_request = new ActiveXObject( "Msxml2.XMLHTTP." + i + ".0" );
                    xmlhttp_request.setRequestHeader("Content-Type","text/xml");
                    xmlhttp_request.setRequestHeader("Charset","gb2312");
                }
                break;
            }catch(e){
                xmlhttp_request = false;
            }
        }
    }else if( window.XMLHttpRequest ){                    //判断浏览器是否支持 XMLHttpRequest 对象
        xmlhttp_request = new XMLHttpRequest();
        if (xmlhttp_request.overrideMimeType){
            xmlhttp_request.overrideMimeType('text/xml');
        }
    }
```

```
}catch(e){
        xmlhttp_request = false;
}
function chkuserid(txt){                                        //检测用户名
        var userid = txt.value;
        if(userid.length > 0){
                var url="/chk.php?userid=" + userid;
                xmlhttp_request.onreadystatechange=function(){      //处理响应
                        if(xmlhttp_request.readyState==4 && xmlhttp_request.status == 200){
                                alert(xmlhttp_request.responseText);
                        }
                }
                xmlhttp_request.open("GET", url, true);
                xmlhttp_request.send(null);
        }
}
</script>
</head>
<body>
<form action="">
<table>
<thead>注册</thead>
<tr><td> 用 户 名 ： </td><td><input type="text" value="" id="userid" name="userid" onblur=
"javascript:chkuserid(this);"/></td></tr>
<tr><td>密码：</td><td><input type="password" value="" id="password" name="password"/>
<tr><td> 再 次 输 入 密 码 ： </td><td><input type="password" value="" id="password2" name=
"password2"/>
<tr><td></td><td><input type="submit" value="提交" name="submit" id="submit" />
</table>
</form>
</body>
</html>
```

分析：在上述程序中，向服务器请求检测用户所输入的用户名，服务器根据用户浏览器的请求进行响应。客户端通过检测请求状态和 HTTP 服务器响应的状态值来判断服务器是否完成响应。当服务器完成响应时，客户端将在浏览器页面输出服务器响应的信息。

【示例 14-4】服务器端如何响应用户浏览器的请求。代码如下所示。

```
<?php
$userArr = array("simon", "smith", "poly");                     //定义一个用户名数组
$userid = $_GET['userid'];                                      //接收所请求的 userid
if (array_search($userid,$userArr)) {                           //从数组中查找该用户名是否存在
        echo "用户名：$userid err！";
}else {
        echo "用户名：$userid ok！";
}
?>
```

分析：在上述程序中，使用数组存储已存在的用户名，并查找用户所请求的 userid 是否存在，根据查找结果抛出不同的信息。客户端在检测到服务器端响应完成后，将对服务器端的响应进行处理，即显示服务器端返回的信息。在实际的应用中，用户名可能存放在数据库或 XML 中，其判断方法与此类似，读者可自行处理。

14.3　使用 POST 方式的 AJAX

向服务器发送请求所使用的 HTTP 方法主要包括 GET、POST、PUT 或 HEAD 等。在上一节已经讲到采用 GET 方式向服务器发送请求，在这一节中，将对在使用 AJAX 时如何采用 POST 方式向服务器提交请求进行详细地讲解。

14.3.1　POST 方式

使用 AJAX 向服务器提交请求，采用 GET 方式，数据追加到 URL 后面。也就是说，浏览器会将各个表单字段元素及其数据按照 URL 参数值对的格式附加在请求行中的资源路径后面，并且大小一般都限制在 1KB 内，然后再使用 XMLHttpRequeset 对象的 OPEN 方法向服务器发送请求。另外，最重要的一点是，它会被用户客户端的浏览器缓存起来。在用户不手动清除浏览历史的情况下，其他人可以从浏览器的历史记录中，读取到该用户的数据，如用户账号、密码等。因此，在某些情况下，GET 方法会带来严重的安全性问题。

虽然这种方式适用于页面上的任何元素，包括表单等。但在实际的应用中，更多是针对表单的操作。针对表单，通常使用 POST 方式。在 AJAX 中使用 POST 方式向服务器提交请求，跟使用 GET 方式非常类似，只是在执行 AJAX 的时候稍有不同。使用 POST 方式，浏览器将把各表单字段元素及其数据作为 HTTP 消息的实体内容发送给 Web 服务器，而不是像 GET 方式那样将作为 URL 地址的参数进行传递。另外，使用 POST 方式传递的数据量要比使用 GET 方式传送的数据量大的多。

使用 POST 方式，为确保服务器知道实体中有参数变量，常常需要设置 AJAX 请求的头信息，其语法格式如下所示。

```
xmlhttp_request.setRequestHeader("Context-Type","application/x-www-form-urlencoded;")
```

其中，xmlhttp_request 为实例化的 XMLHttpRequest 对象，其后为调用 setRequestHeader 方法设置请求的头信息。

注意：返回的数据默认的字符编码是 utf-8，如果用户客户端浏览器页面是 GB2312 或者其他编码，则可能会显示成乱码。此时服务器和客户浏览器页面均可采用 UTF-8 编码，也可在服务器端输出时指定编码。例如，客户端为 GB2312，服务器在输出时指定编码为 GB2312，如下所示。

```
header('Content-Type:text/html;charset=GB2312');
```

使用 POST 方式时，其参数是参数名和值一一对应的参数对，每一对之间使用“&”符号隔开，同时作为 XMLHttpRequest 的 send 方法的参数传送至服务器。服务器端在接收到该参数时，应使用$_POST[ ]全局数组的方式获取参数值。

14.3.2　POST 实例

在实际的应用中，常常在 AJAX 中采用 POST 方式向服务器提交数据。

【示例 14-5】在应用中如何使用 POST 方法向服务器提交数据。代码如下所示。

```
<?xml version="1.0" encoding="GB2312" ?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>Insert title here</title>
<script type="text/javascript">
var xmlhttp_request = false;                                    //设置变量
try{
    if( window.ActiveXObject ){                                 //判断浏览器是否支持 ActiveXObject 对象
        for( var i = 5; i; i-- ){
            try{
                if( i == 2 ){
                    xmlhttp_request = new ActiveXObject( "Microsoft.XMLHTTP" );
                } else{
                    xmlhttp_request = new ActiveXObject( "Msxml2.XMLHTTP." + i + ".0" );
                    xmlhttp_request.setRequestHeader("Content-Type","text/xml");
                    xmlhttp_request.setRequestHeader("Charset","gb2312");
                }
                break;
            }catch(e){
                xmlhttp_request = false;
            }
        }
    }else if( window.XMLHttpRequest ){                          //判断浏览器是否支持 XMLHttpRequest 对象
        xmlhttp_request = new XMLHttpRequest();
        if (xmlhttp_request.overrideMimeType){
            xmlhttp_request.overrideMimeType('text/xml');
        }
    }
}catch(e){
    xmlhttp_request = false;
}
function $(id){
    return document.getElementById(id);
}
function addcomment(){
    var url="14.6.php";
    var status = document.getElementById("divmsg");
    status.value = "正在提交...";
    var param = "name=" + $("name").value + "&email=" + $("email").value + "&comment=" + $("comment").value;
    xmlhttp_request.onreadystatechange=function(){
        if(xmlhttp_request.readyState==4 && xmlhttp_request.status == 200){//响应完全显示信息
            if(xmlhttp_request.responseText == "1"){
                status.value = "发表成功！";
                $("name").value="";
                $("email").value="";
                $("comment").value="";
            }else{
                status.value = "发表失败，请重新发表！";
            }
        }
    }
```

```
        xmlhttp_request.open("POST", url, true);
        xmlhttp_request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");//设置头信息
        xmlhttp_request.send(param);
}
</script>
</head>
<body>
<form action="" name="form1" mothod="post">
        <table>
                <caption>发表评论</caption>
                <tr><td>姓名：</td><td><input type="text" name="name" id="name" /></td></tr>
                <tr><td>e-mail:</td><td><input type="text" name="email" id="email" /></td></tr>
                <tr><td> 评 论 ： </td><td><textarea name="comment" id="comment" rows="6" cols="50"></textarea></td></tr>
        </table>
        <input type="button" value="发表" name="submit" id="submit" onclick="addcomment();"/>
        <input id="divmsg" name="divmsg" value="" readonly />
</form>
</body>
</html>
```

分析：在上述程序中，采用 POST 方式向服务器提交数据。在提交数据时，需要设置 AJAX 请求的头信息，并且在向服务器发送请求和服务器成功响应客户端的请求时，页面上都给出了相应的提示信息。这主要是因为一些数据量比较大或者网络等原因造成页面没有及时更新，用户不知道当前是什么状态造成的。

【示例 14-6】服务器端获取客户端通过采用 POST 方式提交数据的过程。代码如下所示。

```
<?php
$name = $_POST['name'];
$email = $_POST['email'];
$comment = $_POST['comment'];
$fh = @fopen("14.6.txt","ab");
fwrite($fh, "姓名：".$name, strlen($name));
fwrite($fh, "email:" . $email, strlen($email));
fwrite($fh,"评论：" . $comment, strlen($comment));
fclose($fh);
header("Content-Type:text/html; Charset=gb2312");
echo "1";
?>
```

分析：在上述程序中，将客户端通过 POST 方式提交的数据保存在服务器的一个文本文件中。

14.4　第一个 AJAX 程序——Hello,Ajax!

前面对 AJAX 请求的每一部分进行了详细地讲解。这一节将对第一个 AJAX 程序 Hello,Ajax!进行讲解。该 AJAX 程序可实现当用户浏览页面时，单击相应按钮后页面将弹出“Hello, Ajax!”窗口。

14.4.1 浏览器页面发送请求与处理响应

用户在浏览页面时，浏览器端因某种操作触发预先定义的某个 JavaScript 事件，JavaScript 将调用 XMLHttpRequest 对象向服务器发出请求。

【示例 14-7】HTML 页面如何向服务器端发出请求的过程。代码如下所示。

```
<?xml version="1.0" encoding="GB2312" ?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>Insert title here</title>
<script type="text/javascript">
var xmlhttp_request = false;                                    //定义变量
try{
    if( window.ActiveXObject ){                                 //判断浏览器是否支持 ActiveXObject 对象
        for( var i = 5; i; i-- ){
            try{
                if( i == 2 ){
                    xmlhttp_request = new ActiveXObject( "Microsoft.XMLHTTP" );
                } else{
                    xmlhttp_request = new ActiveXObject( "Msxml2.XMLHTTP." + i + ".0" );
                    xmlhttp_request.setRequestHeader("Content-Type","text/xml");
                    xmlhttp_request.setRequestHeader("Charset","gb2312");
                }
                break;
            }catch(e){
                xmlhttp_request = false;
            }
        }
    }else if( window.XMLHttpRequest ){                          //判断浏览器是否支持 XMLHttpRequest 对象
        xmlhttp_request = new XMLHttpRequest();
        if (xmlhttp_request.overrideMimeType){
            xmlhttp_request.overrideMimeType('text/xml');
        }
    }
}catch(e){
    xmlhttp_request = false;
}
function hello(btn){                                                    //定义事件触发函数
    var url="14.6.php";
    xmlhttp_request.onreadystatechange=function(){                      //处理响应
        if(xmlhttp_request.readyState==4 && xmlhttp_request.status == 200){
            bth.value = xmlhttp_request.responseText;
            alert(xmlhttp_request.responseText);
        }
    }
    xmlhttp_request.open("POST", url, true);                            //发出请求
    xmlhttp_request.send(null);
}
</script>
</head>
```

```
<body>
<input type="buttion" value="第一个 AJAX 程序" name="submit" id="submit" onclick="hello(this);"/>
</body>
</html>
```

分析：在上述程序中，当用户单击页面上的按钮时，触发 JavaScript 函数 hello。该函数调用 XMLHttpRequest 对象的 xmlhttp_request 向服务器发出请求。当接收到服务器完整的响应时，将页面按钮所显示的值更改为服务器所响应的值，同时以窗口的形式显示在页面上。

14.4.2　服务器响应

服务器端在接收到用户浏览器端发出的请求后，对该请求作出响应。

【示例 14-8】在服务器端如何处理用户浏览器的响应。代码如下所示。

```
<?php
echo "Hello, Ajax!";
?>
```

分析：在上述程序中，服务器端响应客户端的请求，输出字符串“Hello, Ajax!”。客户端检测到服务器端完整的响应后，将显示如图 14-2 所示的处理响应页面。

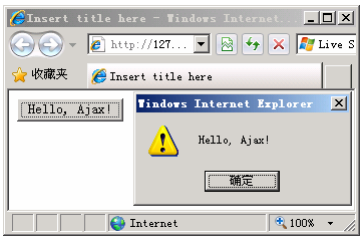

图 14-2　处理响应页面

14.5　本章实例

AJAX 的应用广泛，在实际的应用中也经常会用到。这一节将对实际应用中最常用的实例进行分析。

14.5.1　读取服务器文本文件

在实际应用中，常常需要在用户浏览器读取服务器端的文件，这时可采用 AJAX 的方式读取文件。用户在浏览该页面时不能自动读取文件，只有当用户向服务器请求该文件时，才能将该文件读取至浏览器。

【示例 14-9】如何在用户浏览器端读取服务器端文本文件。代码如下所示。

```
<?xml version="1.0" encoding="GB2312" ?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>Insert title here</title>
<script type="text/javascript">
```

```
var xmlhttp_request = false;                                    //定义变量
try{
    if( window.ActiveXObject ){                                 //判断浏览器是否支持 ActiveXObject 对象
        for( var i = 5; i; i-- ){
            try{
                if( i == 2 ){
                    xmlhttp_request = new ActiveXObject( "Microsoft.XMLHTTP" );
                } else{
                    xmlhttp_request = new ActiveXObject( "Msxml2.XMLHTTP." + i + ".0" );
                    xmlhttp_request.setRequestHeader("Content-Type","text/xml");
                    xmlhttp_request.setRequestHeader("Charset","gb2312");
                }
                break;
            }catch(e){
                xmlhttp_request = false;
            }
        }
    }else if( window.XMLHttpRequest ){                          //判断浏览器是否支持 XMLHttpRequest 对象
        xmlhttp_request = new XMLHttpRequest();
        if (xmlhttp_request.overrideMimeType){
            xmlhttp_request.overrideMimeType('text/xml');
        }
    }
}catch(e){
    xmlhttp_request = false;
}
function read(){                                                                    //读取文件
    var url="14.7.txt";
    var txt = document.getElementById("txt");
    xmlhttp_request.onreadystatechange=function(){                                  //处理响应
        if(xmlhttp_request.readyState==4 && xmlhttp_request.status == 200){
            txt.value = xmlhttp_request.responseText;
        }
    }
    xmlhttp_request.open("POST", url, true);                                        //发送请求
    xmlhttp_request.send(null);
}
</script>
</head>
<body>
<input type="button" value="读取文件" name="submit" id="submit" onclick="read();"/>
<textarea rows="10" cols="30" id="txt" name="txt"></textarea>
</body>
</html>
```

分析：在上述程序中，单击按钮则向服务器提交请求，服务器响应客户端所提交的请求并进行处理。此时直接输出文本文件的内容，用户浏览器端在检测服务器响应完成时，将服务器响应的值放在文本区域内。

14.5.2 三级联动下拉框

在实际应用中，常常让用户选择所在省份、市、县等。当用户选择省份时，市级下拉框将显示为该省份下属的市。若在市级下拉框选择相应市，则在城市下拉框中显示用户所选市

的下属县。这一节将对类似这种的三级联动下拉框进行详细地讲解。

【示例 14-10】如何在 HTML 页面创建 XMLHttpRequest 对象。代码如下所示。

```
<?xml version="1.0" encoding="GB2312" ?>
<!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>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312" />
<title>Insert title here</title>
<script type="text/javascript">
var xmlhttp_request = false;
try{
    if( window.ActiveXObject ){                          //判断浏览器是否支持 ActiveXObject 对象
        for( var i = 5; i; i-- ){                        //根据浏览器版本以不同方式创建该对象
            try{
                if( i == 2 ){
                    xmlhttp_request = new ActiveXObject( "Microsoft.XMLHTTP" );
                } else{
                    xmlhttp_request = new ActiveXObject( "Msxml2.XMLHTTP." + i + ".0" );
                    xmlhttp_request.setRequestHeader("Content-Type","text/xml");
                    xmlhttp_request.setRequestHeader("Charset","gb2312");
                }
                break;
            }catch(e){
                xmlhttp_request = false;
            }
        }
    }else if( window.XMLHttpRequest ){                   //判断浏览器是支持 XMLHttpRequest 对象
        xmlhttp_request = new XMLHttpRequest();
        if (xmlhttp_request.overrideMimeType){
            xmlhttp_request.overrideMimeType('text/xml');
        }
    }
}catch(e){
    xmlhttp_request = false;
}
```

分析：在上述程序中，首先判断用户浏览器是否支持 ActiveXObject 对象，若支持则根据浏览器版本采用不同的方式创建；若用户浏览器不支持 ActiveXObject 对象，则采用直接使用 XMLHttpRequest 对象进行创建。

【示例 14-11】用户在浏览页面选择省份时，浏览器如何向服务器发送请求，并在用户浏览器页面显示服务器的响应过程。代码如下所示。

```
function selprovince(sel){                                    //设置二级元素
    var selpro = sel.options[sel.selectedIndex].value;
    var url="14.9.php?act=getcity&pro=" + selpro;

    var city = document.getElementById("city");
    xmlhttp_request.onreadystatechange=function(){
        if(xmlhttp_request.readyState==4 && xmlhttp_request.status == 200){
            var json;
            json = eval(xmlhttp_request.responseText);
            if (json.length==0){
```

```
                alert("无城市信息!");
                return;
            }
            if (navigator.appName == "Microsoft Internet Explorer"){ //根据浏览器的不同进行处理
                city.options.length=0;
                for(var i=0;i<json.length;i++) {
                    var opt = document.createElement("option");
                    opt.value = json[i]['code'];
                    opt.text = json[i]['title'];
                    city.options.add(opt, i);
                }
            }else{
                st="";
                for(var i=0;i<json.length;i++) {
                    st=st+"<option value='"+json[i]['code']+"'>"+json[i]['title']+"</option>";
                }
                city.innerHTML=st;
            }
        }
    }
    xmlhttp_request.open("POST", url, true);
    xmlhttp_request.send(null);
}
```

分析：在上述程序中，当用户从省份下拉框选择省份时，将触发该 JavaScript 函数。该函数获取页面省份下拉框所选择的值，并向服务器发送请求。浏览器在接收到服务器完整响应时，将在 HTML 页面上显示结果。由于 MicroSoft 的 Internet Explorer 系列浏览器和其他浏览器对于 DOM 的处理不相同，因此在处理服务器响应时要采用不同的方式。

【示例 14-12】用户在选择市级下拉框时，将显示该市下属的县级信息。代码如下所示。

```
function selcity(sel){
    var city = sel.options[sel.selectedIndex].value;
    var url="14.9.php?act=getcounty&city=" + city;
    var county = document.getElementById("county");
    xmlhttp_request.onreadystatechange=function(){
        if(xmlhttp_request.readyState==4 && xmlhttp_request.status == 200){
            var json;
            json = eval(xmlhttp_request.responseText);
            if (json.length==0){
                alert("无县份信息!");
                return;
            }
            if (navigator.appName == "Microsoft Internet Explorer"){ //根据浏览器的不同进行处理
                county.options.length=0;
                for(var i=0;i<json.length;i++) {
                    var opt = document.createElement("option");
                    opt.value = json[i]['code'];
                    opt.text = json[i]['title'];
                    county.options.add(opt, i);
                }
            }else{
                st="";
                for(var i=0;i<json.length;i++) {
                    st=st+"<option value='"+json[i]['code']+"'>"+json[i]['title']+"</option>";
                }
```

```
                        county.innerHTML=st;
                    }
                }
            }
        xmlhttp_request.open("POST", url, true);
        xmlhttp_request.send(null);
}
</script>
</head>
<body>
省份：
<select id="province" name="province" onChange="selprovince(this);">
        <option value="sc">四川省</option>
        <option value="bj">北京市</option>
        <option value="sd">山东省</option>
</select>
市：
<select id="city" name="city" onChange="selcity(this);">
</select>
县：
<select id="county" name="county"></select>
</body>
</html>
```

分析：在上述程序中，用户在选择二级下拉框时，将触发该 JavaScript 函数。该函数获取页面二级下拉框的值，并向服务器发送请求。浏览器在接收到服务器完整响应时，将在 HTML 页面上显示服务器的响应。由于 IE 浏览器和其他的浏览器（如 Firefox）在 DOM 处理上的不同，在处理服务器的响应数据时，采用了不同的方式。

【示例 14-13】在服务器端如何响应用户浏览器的请求。代码如下所示。

```
<?php
$cityArr = array(                                          //定义二级元素
'sc' => array('cd'=>'cd','zg'=>'zg','ya'=>'ya'),
'bj' => array('aa'=>'bb','cc'=>'dd','ee'=>'ff')
);
$countyArr = array(                                        //定义三级元素
'cd'=>array('jj'=>'AA 区','ch'=>'BB 区','wh'=>'CC 区'),
'zg'=>array('fs'=>'DD 县','rx'=>'EE 县','gx'=>'FF'),
'ya'=>array('aa'=>'GG','bb'=>'HH','cc'=>'II'),
'aa'=>array('dd'=>'JJ','ee'=>'KK','ff'=>'LL'),
'bb'=>array('gg'=>'MM','hh'=>'NN','ii'=>'OO'),
'cc'=>array('jj'=>'PP','kk'=>'QQ','ll'=>'RR')
);
header('Content-Type:text/html;charset=GB2312');           //设置返回编码
$act = $_GET['act'];                                       //获取用户端请求
if ($act == "getcity") {
        $pro = $_GET['pro'];
        $json = $cityArr[$pro];
        $js = "[";
        foreach ($json as $k =>$v) {                       //构造 JSON 数据格式
                $js .= "{'code':'$k','title':'$v'},";
        }
        $jss = substr($js,0,strlen($js)-1);
        $jss .= "]";
```

```
        echo $jss;
}else {
        $city = $_GET['city'];
        $json = $countyArr[$city];
        $js = "[";
        foreach ($json as $k =>$v) {                                    //构造 JSON 数据格式
                $js .= "{'code':'$k','title':'$v'},";
        }
        $jss = substr($js,0,strlen($js)-1);
        $jss .= "]";
        echo $jss;
}
?>
```

分析：在上述程序中，通过将二级元素和三级元素放在数组的形式，并根据客户端的请求进行响应。用户浏览器端在检测到服务器端响应后，将服务器的响应信息添加至相应的下拉框中。读者在应用时可将其信息存放在数据库，然后从数据库中读取数据。

14.6 小结

本章主要对 AJAX 作了详细讲解。AJAX 是近年来炒得最热的技术，它是几种老技术的全新组合，主要用于提升用户的浏览体验。AJAX 可以在不刷新页面的情况下向服务器发送请求，并将执行结果显示在客户端页面上。但是在网站上大量地使用 AJAX 将不利于搜索引擎的搜索，因此，适当地在网站上使用 AJAX 技术显得尤为重要。

本章结合当前网站上最为常见的几个 AJAX 实例，对 AJAX 的应用进行讲解。读者应结合本章实例体会 AJAX 的使用，掌握 AJAX 技术。