一种基于 Ajax 技术的分页方法

A Paging Technology Based on Ajax

廖新飞 （温州职业技术学院 计算机系 浙江 温州 325035）

摘 要： Ajax 将浏览器客户端和服务器端的传统同步交互通信方式改变为异步交互通信方式，使用 Ajax，就算不重载刷新 Web 页面，用户也能顺利快速得到 Web 服务器的数据，可大大改善了传统分页显示方法。本文首先介绍了 Ajax 包含的主要技术及工作原理，然后给出了一个使用 Ajax 技术进行分页显示的实例。

关键词： 分页 Ajax XHTML CSS 异步交互

1 引言

随着网络技术的快速发展，用户需要通过 Web 频繁地从数据库中获取信息。如果获取的数据量较大，就需要分页显示从数据库中读取出来的数据，这样可极大方便用户查询数据。因此，分页显示是 Web 应用程序的一个重要功能。传统的 Web 应用程序采用同步交互过程，分页显示数据时需要刷新整个页面，存在等待时间长，显示位置出现空白等缺陷。而采用 Ajax（Asynchronous Javascript and XML）技术的分页方法，无需刷新整个页面，响应时间短，大大改善了传统分页显示方法。

2 Ajax 简介

2.1 Ajax 主要技术

Ajax 是多种技术的综合，其包含的主要技术如下[1]：

1）XHTML 和 CSS

XHTML 和 CSS 用于实现客户端标准化的界面表示。CSS 提供了从内容中分离应用样式和设计的机制。XHTML 是一种用 XML 重写 HTML 的标记语言。

2）DOM（Document Object Model）

DOM（文档对象模型）是表示文档（比如 HTML 和 XML）和访问、操作构成文档各种元素的应用程序接口（API），JavaScript 通过 DOM 来解析 XML 文档。

3）XML 和 XSLT

XML 作为数据传输的媒介，是服务器与客户端交换数据的一种格式。XSLT 是一种用来转换 XML 文档结构的语言。

4）XMLHttpRequest

XMLHttpRequest 负责将用户信息以异步通信的方式发送到服务器端，并接收服务器返回的响应信息和数据。

5）JavaScript

JavaScript 是一种脚本语言，将 HTML 与 DOM、XMLHttpRequest 等对象联系起来，作为它们之间沟通的渠道。在 Ajax 中 JavaScript 主要被用来传递用户界面上的数据到服务端并返回结果。

2.2 Ajax 工作原理

Ajax 工作原理如图 1 所示，用户在与浏览器的交互过程中，通过 JavaScript 技术能捕获到用户在客户端产生的事件，然后由 JavaScript 创建和配置一个 XMLHttpRequest 对象，并通过该对象异步地把请求发送到服务器端[2]，这时，用户可以继续进行其它的操作，而不必去等待服务器的响应。服务器端在接收到请求后，通过服务器端程序处理请求并把结果返回，返回的结果被 XMLHttpRequest 对象捕获到并返回给 JavaScript，再由 JavaScript 在回调函数中通过 DOM 对页面中的 HTML 元素的操作，实现丰富友好的界面和交互。

传统的 Web 应用程序采用同步交互过程，在交互过程中存在处理—等待—处理—等待的缺陷[3]。与传统的 Web 应用程序相比，Ajax 采用异步交互过程。Ajax 通过调用 XmlHttpRequest 对象实现与服务器的异步通讯，相当于在用户和服务器之间加了一个中间层，使用户操作与服务器响应异步化，并不是所有的用户请求都提交给服务器，像一些数据验证和数据处理等都交给 AJAX 引擎自己来做，只有确定需要从服务器读取新数据时再由 AJAX 引擎代为向服务器提交请求。这样把以前的一些服务器负担的工作转嫁到客户端，

利用客户端闲置的处理能力来处理，减轻服务器和带宽的负担。

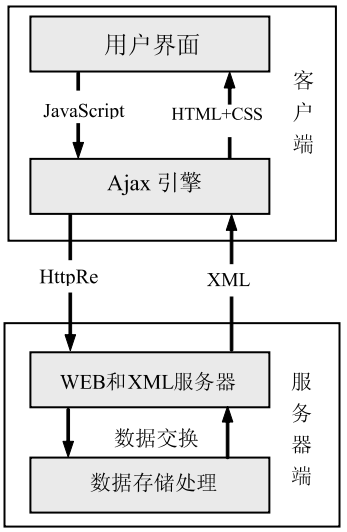

图 1　Ajax 工作原理图

3　基于 Ajax 的分页实现

下面以 JSP 中图书信息的分页显示，来介绍 Ajax 技术在分页中的应用。Web 服务器采用 Tomcat5.0，数据库采用 MySQL4.1，数据库 bookdb 中表 book 的基本结构如表 1 所示。

表 1　book 基本结构表

字段名	数据类型	长度	约束
ID	int	10	Primary Key
图书名称	varchar	50	Not Null
作者	varchar	50	Not Null
出版社	varchar	50	Not Null

基本思想是客户端由 JavaScript 通过 XMLHttpRequest 对象向服务器端发起分页获取数据的请求，服务器端收到请求后，从数据库中获取指定页的数据并将数据封装成 XML 格式返回给客户端，客户端接收到返回的数据后，再由 JavaScript 通过 DOM 将封装在 XML 中的数据解析出来并显示在 HTML 页面中。其具体过程如下：

1）初始化，创建 XMLHttpRequest 对象。

……

```
if(window.XMLHttpRequest){ //Mozilla 浏览器
//创建 XMLHttpRequest 对象
http_request = new XMLHttpRequest();
}
else if(window.ActiveXObject){ // IE 浏览器
//创建 XMLHttpRequest 对象
http_request = new ActiveXObject("Microsoft.XMLHTTP");
}
……
//指定回调函数
http_request.onreadystatechange = populateList;
……
```

说明：①XMLHttpRequest 对象在不同浏览器下的创建方法不同，因此要先获取当前使用的浏览器类型，然后再创建 XMLHttpRequest 对象。② 回调函数指明当服务器返回 http 请求响应信息时客户端的处理方式。

2）客户端通过 XMLHttpRequest 对象向服务器端（getData.jsp）发送请求。

```
……
http_request.open("get","getData.jsp?currentPage=pn",true);//创建一个新的 http 请求
http_request.send(null);//向服务器发送请求
……
```

说明：①JavaScript 本身并不具备向服务器发送请求的能力，要么使用 window.open() 方法重新打开一个页面向服务器提交请求，要么使用 XMLHttpRequest 对象发送请求。不同的是前者是同步交互模式，而后者是异步交互模式。②open 方法：创建一个针对目标 URL、采用相应 http 请求方式的 http 请求。第一个参数：http 请求的方法，GET 或 POST。第二个参数：目标 URL。第三个参数：指定 http 请求是否异步执行。即在等待服务器返回信息的时间内是继续执行下面的代码（异步方式）还是等待服务器返回响应数据再执行下面的代码（同步方式）。true 为异步，false 为同步。默认为 true。send 方法：将 http 请求发送给服务器。其参数为发送的数据，此处为 null。③XMLHttpRequest 是 Ajax 技术体系中最为核心的技术。Ajax 通过使用 XMLHttpRequest 对象回调服务器以调用 Web 服务而无需刷新整个页面。这样就可以象桌面应用程序只同服务器进行数据层面的交换，而不用每次都刷新界面也不用每次将数据处理的工作提交给服务器来做，这样即减轻了服务器的负担又加快了响应速度、缩短了用户

等候时间。

3)服务器端(getData.jsp)接受并处理请求,将获取的指定页的数据封装成 XML 格式后返回给客户端。

```
Statement stmt; //创建一个 Statement 对象
ResultSet rs; //创建一个 ResultSet 对象
int ipage=6; //一页显示的记录数
int RowCount=0; //记录总数
int allpage; //总页数
int cpage; //当前页
String strPage,sql;
//加载 mysql 数据驱动程序
try{Class.forName("com.mysql.jdbc.Driver").newInstance();
//127.0.0.1/bookdb?user=root&
conn=java.sql.DriverManager.getConnection("jdbc:mysql:
//创建 mysql 连接
useUnicode=true&characterEncoding=GB2312");
stmt=conn.createStatement();
//取得待显示页码
strPage=request.getParameter("currentPage");
if(strPage==null){
//表明在 QueryString 中没有 page 这一个参数,此时显示第一页数据
    cpage=1;}
else{cpage=Integer.parseInt(strPage);
if(cpage<1) cpage=1; //页码值小于时1,使其值为1}
sql="SELECT COUNT(*) FROM book";
//为获取记录总数,执行 SQL 查询
rs=stmt.executeQuery(sql);
//获取记录总数
while(rs.next()) RowCount=rs.getInt(1);
allpage=(RowCount+ipage-1)/ipage; //记算总页数
if(cpage>allpage) cpage=allpage; //调整待显示的页码
sql="select ID,图书名称,作者,出版社 from book limit "+(cpage-1)*ipage+","+ipage;
rs=stmt.executeQuery(sql); //获取指定页的数据
//生成 XML 文档
String xmlstr="<?xml version="1.0" encoding="gb2312"?>";
xmlstr=xmlstr+"<list>"+"<currentPage>"+cpage+"</currentPage>";
xmlstr=xmlstr+"<totalPage>"+allpage+"</totalPage>";
xmlstr=xmlstr+"<booklist>";
String xmlstr2="";
//将结果集拼装成 XML
while(rs.next()) //还有记录时继续
{
xmlstr2=xmlstr2+"<books id=""+rs.getInt(1)+"">"; xmlstr2=xmlstr2+"<bookname>"+rs.getString(2)+"</bookname>"; xmlstr2=xmlstr2+"<author>"+rs.getString(3)+"</author>";
xmlstr2=xmlstr2+"<publisher>"+rs.getString(4)+"</publisher>";
xmlstr2=xmlstr2+"</books>";}
xmlstr=xmlstr+xmlstr2;
xmlstr=xmlstr+"</booklist>";
xmlstr=xmlstr+"</list>";
//输出 XML 文档,以便 index.jsp 中的回调函数 populateList 使用
response.getWriter().write(xmlstr);
rs.close();//关闭结果集
stmt.close(); //关闭 SQL 语句对象
conn.close(); //关闭数据库
}catch(Exception e){out.print(e);}
```

4)客户端接收从服务器端返回的数据,在回调函数 populateList 中通过 DOM 将 XML 格式的数据解析并显示出来。

```
function populateList(){//此函数为回调函数
if(http_request.readyState==4){ // 判断对象状态
// 信息已经成功返回,开始处理信息
if(http_request.status==200){
//获取服务器返回的 XML 文档对象
var doc=http_request.responseXML;
//下行获取 XML 中封装的页码值
var currentPage=
doc.getElementsByTagName("currentPage")[0].
```

```
firstChild.data;
//下行获取 XML 中封装的总页数值
var totalPage =
doc.getElementsByTagName("totalPage")[0].firstChild.data;
……
//下行获取 XML 中封装的 books 元素
var bookslist = doc.getElementsByTagName("books");
var innerHTML = "";
if((bookslist!=null)&&(bookslist.length!=0)){
//如果 books 不为空,则继续处理
……
for(var i=0;i<bookslist.length;i++){
var books = bookslist[i];
var id = books.getAttribute("id");
//下行获取图书名称的值
var bookname =
(books.childNodes[0].firstChild==null)?"":books.
childNodes[0].firstChild.data;
//下行获取作者的值
var author =
(books.childNodes[1].firstChild==null)?"":books.
childNodes[1].firstChild.data;
//下行获取作者的值
var publisher =
(books.childNodes[2].firstChild==null)?"":books.
childNodes[2].firstChild.data;
innerHTML += "<tr>";
innerHTML += "<td width=\"10%\" height=\"25\">" +
id + "</td>";
innerHTML += "<td width=\"35%\"
height=\"25\">" + bookname + "</td>";
innerHTML += "<td width=\"30%\"
height=\"25\">" + author + "</td>";
innerHTML += "<td width=\"25%\"
height=\"25\">" + publisher + "</td>";
innerHTML += "</tr>";    }
innerHTML += "</table>\r\n";
}else{
    innerHTML += "暂时没有任何图书信息";}
                ……
}
```

说明:在回调函数中要检查 XMLHttpRequest 对象的 readyState 和 status 值。只有 readyState = =4(代表 http 请求发送成功并且服务器已经传回所有信息)并且 status = =200(代表页面正常执行)时,才可以处理信息并更新页面内容。

最后,分页显示结果如图 2 所示。经测试,分页时无需刷新整个页面,响应时间短,浏览器不会出现空白等待,大大改善了用户体验。

图书信息列表

ID	图书名称	作者	出版社
7	计算机操作系统教程	周爱武,汪海威,李知兵	清华大学出版社
8	分布式操作系统	塔嫩鲍姆,Tanenbaum	机械工业出版社
9	操作系统原理与实训教程	范辉,谢青松	高等教育出版社
10	Linux操作系统教程	刘胤杰,岳浩	机械工业出版社
11	现代操作系统	Tanenbaum,坦尼鲍姆	机械工业出版社
12	微机操作系统与网络实用技术	刘宏忠,成汝震	高等教育出版社

第一页 上一页 下一页 最末页 (第2页,总共4页)

图 2 分页显示结果

4 结束语

Ajax 是 2005 年 2 月才正式提出的一项综合技术,其主要特点是为 Web 开发提供异步的数据传输和交换方式,可以在不重载(Reload)刷新(Refresh)界面的情况下与服务器进行数据交换,从而丰富了浏览器客户端功能,解决了浏览器频繁刷新页面等待数据传输的问题,大大改善了 Web 应用程序的用户体验。

参考文献

1 柯自聪. Ajax 开发精要:概念、案例与框架. 北京:电子工业出版社,2006.

2 沈泽刚,王月. 基于 Ajax 技术的表单数据动态验证. 计算机系统应用,2007,16(6):83-85.

3 张桂元. 征服 Ajax—Web2.0 快速入门与项目实践(Java). 北京:人民邮电出版社,2006.