AJAX 应用的典型设计模式①

周 杨

(军事经济学院 基础部计算机教研室，武汉 430035)

摘 要: AJAX 是 WEB 领域的前沿技术，它提供了新的互联网交互模型，扩展了 WEB 应用的能力。文章对比分析了 AJAX 工作模式与传统 WEB 工作模式的主要区别，提出了动态加载、预见式缓存、内容分块三个典型的 AJAX 应用设计模式，并较为深入地分析了各个模式的设计原理，展示了相关的实例代码。

关键词: AJAX；设计模式；WEB；异步

Typical AJAX Design Patterns

ZHOU Yang

(Computer Teaching and Researching Section, Military Economics Academy, Wuhan 430035, China)

Abstract: Ajax is a front technology in the Web. It Provides a new model of intemet interaction, which greatly expanded the capacity of Web applications. In this paper, we analyze the main differences between the Ajax design pattems and traditional Web applications, then put forward four typical Ajax design pattems which are Dynamic Loading, Cache-predictable Data and Content partitioned. We described the various design patterns on the aspects of the problems, typical scene and its architecture.

Keywords: AJAX; design pattern; WEB; asynchroni

1 AJAX技术及其实现原理

1.1 AJAX 技术简介

AJAX(Asynchronous JavaScript And XML)作为 WEB 领域的前沿技术，是一种创建交互式网页应用的开发技术。作为一种新的 WEB 设计方式，AJAX 本身并不算新技术，它是建立在 JavaScript、XHTML 和 CSS、DOM、XMLHttpRequest、XML 等大量成熟技术基础之上的一项综合技术，并能够对各大浏览器和平台都支持的、具有公开标准的这些现有技术进行重新应用[1]。

在传统网页的工作模式下，数据采用同步交互方式。用户通过点击网页上的按钮或链接，向 WEB 服务器发送请求信息 HttpRequest，WEB 服务器收到用户的请求后，与数据库服务器交换数据，再向发出请求的用户返回一个 HTML 页面显示数据的改变。于是，服务器在处理请求时，用户多数时间处于等待状态，屏幕内容也是一片空白。AJAX 工作模式与传统网页工作模式最大的不同是 AJAX 采用数据异步传输与请求机制，使客户端与服务器之间的数据通信在后台运行。在浏览器与 WEB 服务器之间增加了一个 AJAX 引擎，其实质就是一些复杂的 JavaScript 程序，这些程序通过调用 XMLHttpRequest 对象的属性和方法与服务器进行数据交互，并通过 DOM 解析处理 XML 文档和部分更新 HTML 页面的内容。因此，用户发出 HTTP 请求后，不必再等待服务器的响应数据来刷新页面，而是继续通过页面和服务器进行其他交互，AJAX 引擎则会自动选择适当的时机向服务器请求数据并将返回的数据显示在客户端。这样，服务器负担的部分工作被转交给客户端处理，节约了 ISP 空间并减轻了服务器及带宽的负担，极大地提高了用户界面的友好程度[2]。图 1 描述了 AJAX 工作模式与传统 WEB 工作模式的主要区别。

① 收稿时间:2010-05-14;收到修改稿时间:2010-06-11

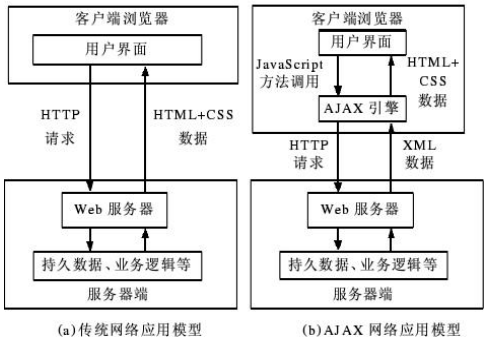

图 1 传统 WEB 服务与应用 AJAX 的 WEB 服务

1.2 AJAX 技术的实现过程

通常情况下，AJAX 只需使用 XMLHttpRequest、DOM、XML、JavaScript 以及 CSS 就可以完成异步交互。其中，利用 XMLHttpRequest 对象与服务器交换数据是技术实现的关键。AJAX 采用 Request/Server 模式形成发送与接收 XMLHt-tpRequest 请求的程序框架，该框架的基本流程是：对象初始化，发送请求，服务器接收，服务器返回，客户端接收，修改客户端页面内容。

首先，初始化对象并发出 XMLHttpRequest 请求，即利用 JavaScript 向服务器发送 HTTP 请求，需要注意的是应根据浏览器的不同初始化对象：

```
Function makeRequest (url) {
If (window.XMLHttpRequest)
xhr= new XMLHttpRequest ();  //对于 Mozila 等浏览器，使用 XMLHttpRequest 方法进行初始化
else {If (window.ActiveXObject)
try{
xhr= new ActiveXObject ();  //对于 IE6 浏览器，使用 ActiveXObjec 方法进行初始化
}
catch (e) {}  //抛出异常并进行自动处理
}
}
```

接着，指定响应处理函数，即指定当服务器返回信息时客户端的处理方式。xhr. onreadystatechange = function ();其中，function ()是用 JavaScript 定义进行客户端处理的函数。然后，通过调用 XMLHttpRequest 对象的 open 和 send 方法向服务器发出 HTTP 请求，其中，true 表示请求是异步，而 url 则代表请求的目标地址：

```
xhr.open(“GET”,url ,true);
xhr.send ( null);
```

最后，使用 function()函数处理服务器返回的信息。通过 xhr.readyState 判断信息是否已经返回，并执行相关操作。

```
Function() {
if(xhr.readyState==4) { //信息已经返回，可以开始处理
if(xhr.status==200) { //页面正常，可以开始处理信息
outMsg=xhr.response.Text;}
Else {outMsg=”Error! Xhr Status:”+ xhr.status; //页面出错}}}
```

2 AJAX应用的典型设计模式

目前，AJAX 应用已经渗透到诸多业务领域，对于不同的应用目的，应当有不同的设计方案。而在 AJAX 广泛的应用实践中，由于缺乏较为一致的设计思路和规范、清晰的体系结构，设计中不规范、不合理的状况相当严重。通过研究 AJAX 的基本设计理论并综合目前大量的应用实践，这里抽象提炼出几种最具代表性的 AJAX 应用设计模式，将其通称为 AJAX 设计模式。

2.1 动态加载模式

在传统的 WEB 应用中，不存在异步通信的概念，“提交页面——等待响应——全屏刷新”是其固有的交互方式，客户端与服务器之间的每一次数据交换，哪怕只是获取或提交少量的数据，都会促使整个页面的重载和全屏的刷新，用户的操作也会因为屏幕刷新而被频繁中断。对于较为复杂的 WEB 应用，由于页面上存在按钮、菜单等可视化元件，交互过程中这些元件一次次毫无意义地重载，不可避免地造成了冗余数据的批量加载，浪费了大量带宽。

AJAX 引入异步交互方式，可以实现交互过程中的动态加载，能够非常有效地避免加载与用户交互无关的冗余数据，从而加快了交互的速度，节省了带宽，使用户体验得到了有效改善。而且，由于异步交互在后台进行，通过对用户请求的异步提交和响应数据的动态加载，避免了提交整个页面和全屏刷新，整个交

互过程中不会发生用户操作被阻断的现象。因此，可以将动态加载模式定义为：在 WEB 程序中利用 AJAX 技术，采取异步通信的方式，根据需要请求获取或提交所必需的数据，并将服务端响应数据或消息以动态方式加载到当前页面中的一种 AJAX WEB 应用设计方案[3]。动态加载可视为最基本的 AJAX 应用设计模式，该模式对于提高程序响应速度、改善应用体验、节省服务器资源有着重要的意义。

2.1.1 设计原理

实现动态加载的前提是要实现异步通信，为此需要创建一个甚至多个 XMLHttpRequest 对象承担请求任务。动态获取数据时应将请求类型设置为 Get，而动态提交数据时则应将请求类型设置为 Post。Get 请求通常需要初始化一些必要的参数，服务器依据请求参数返回相应的数据。在实现异步通信的基础上，通过对用户事件的捕获以及对服务器响应数据的动态呈现，达到动态加载数据的目的。动态加载模式的程序体系结构遵循 MVC 模式，在客户端需要实现 AJAX 引擎，这也是所有 AJAX 应用的共同之处。客户端实体类用于描述相关的领域模型，业务类用于实现客户端的业务逻辑，而 AjaxHttp 类则负责完成与服务器的异步通信。在.Net 中通常由.ashx 类型的程序来处理异步请求，作为 MVC 模式中的控制器，通过它调用相应的业务和数据模型实施具体操作，并将结果返回至客户端。客户端 AJAX 引擎负责接收响应数据，对响应数据进行解析和组织后，呈现到视图。

2.1.2 实例代码

客户端 AjaxHttp 类作为 AJAX 引擎的核心程序之一，用来封装 XMLHttpRequest 的功能。下面是实现一个 AjaxHttp 类的核心代码：

AjaxHttp=new function()

{this.reqList=[]; //定义一个 XMLHttpRequest 的对象数组

this.getXHR=function();{…} //创建一个兼容不同浏览器的 XMLHttpRequest 对象

this.send=function(url,method,callback,data,urlencoded,call-back 2) {…} //封装 XHR 对象向服务器发送请求的相关操作

this.clearReqlist= function();{…} //清除所有 XHR 数组元素，释放资源

this.sendPost= function(url,data, callback,clear, callback2) {…} //进一步封装 XHR 并以 Post 方式发送请求时的代码

this.sendGet= function(url,callback,args,clear, callback 2) {…} //进一步封装 XHR 并以 Get 方式发送请求时的代码

this.loading();{…} //在向服务器发送请求的过程中显示加载数据的提示

this.afterloading();{…}//服务器处理完毕时，显示相关提示

}

2.2 预见式缓存模式

尽管浏览器本身具有一定的数据缓存能力，但只能针对静态内容，对于时刻变化的动态数据，这种缓存模式会受到一定的限制。因此，需要在数据访问中引入比较智能的策略。预见式缓存模式可以定义为：在涉及庞大数据访问的 WEB 程序中，利用 AJAX 来实现一种机制，这种机制通过监视用户的客户端行为，按照预先制定的判断逻辑，对用户下一步可能发出的数据请求进行预载，并将预载请求所得数据进行本地缓存或直接以动态增量的方式呈现到客户端视图界面中。预见式缓存模式重叠使用了动态加载模式，使得用户在浏览服务端较庞大数据时，可以获得非常迅速敏捷的响应，甚至能带来“数据持续不断”的用户体验，能够有效改善传统模式下庞大数据访问“卡壳”的状况[4]。

2.2.1 设计原理

预见式缓存模式需要在客户端实现一个请求代理，由此代理来负责处理客户端的数据请求。如果请求的数据已经存在于本地，就直接利用缓存中的数据来响应，否则向服务器提交异步请求，等待服务器端做出响应并对响应数据进行呈现后，将响应数据缓存于本地。请求代理最为重要的任务是实现预见式缓存数据的功能、设置触发数据预见式加载的条件、制定具体的加载方式及数据本地缓存的策略等。请求代理

内部实现了类似监视器的功能，该监视器负责触发预见式缓存并跟踪客户端用户的行为，根据获得的相关信息判断是否应当立即启动一个异步请求来预载数据。AJAX 引擎中请求代理程序的主要功是在监视器程序内实现预见式缓存的触发机制，包括触发条件、触发方式、数据预载的方法及大小等，从而实现对数据的本地缓存。

2.2.2 实例代码

在不同的应用场景中，预见式缓存的触发机制与本地缓存策略均有所差异，这里结合实例代码对预见式缓存的触发及进行本地缓存的机制做进一步阐述。

通常，在电子商务中需要实现大量商品数据的编辑、修改等管理操作，采用预见式缓存模式对商品数据采用列表方式进行展示，通过该列表实现对商品的检索和浏览。这里是客户端代理程序的主要代码，由 Productlist 类实现：

```
Productlist=new function()
{this.url=”GetProduct.ashx”;      //服务器端控制程序
this.pindex=1;      //设置当前已经显示商品的页码
this.isall=false;      //判断当前数据是否已经全部显示
this.pagenum=5;      //设置每页显示商品的数目
…
this.monitor=fuction()      //实现预见式缓存的监视程序
{//获取商品列表对象
Var Productlist=document.getElementById (“Prod uetlist”);
If(Productist)
{if(this.isall) return;      //判断数据是否已经全部显示
//滚动条到页面底部，触发预见式缓存
if(productlist.scrollTop+Productlist.clientHeight)>=Produetlist.scrollHeight-20)
//提前发出异步请求，获取下页商品数据
AJAXHttp.sendGet(this.url+ ” ?pindex= ” +this.pindex+ ” Pagenum= ” +this.pagenum,null,this.recGetproduct,null,null);
}
}
//回调函数负责接收预载数据，并将其添加到商品列表底部
this.recGetproduct=function(){…}…}
```

2.3 内容分块模式

WEB 页面视图的设计大都遵循 CSS 布局原则，即采用层的方式使页面布局更加有序。而传统应用通常以页面为功能单元，页面分块仅对视图进行分割，分块与服务器之间的通信需要提交整个页面。因此，页面能够实现的应用逻辑非常有限，这也成为限制客户端发展的重要因素。

内容分块模式可以定义为：在 WEB 应用中利用 AJAX 对页面进行分块设计，每个页面由多个内容分块组成，各分块的动态加载及数据的引用均保持相对独立的运行逻辑的一种 AJAX WEB 应用设计方案。AJAX 的页面内容分块模式能够克服传统应用的缺陷。利用 AJAX 分块模式可以实现页面结构的动态调整，使视图组织更加动态灵活，页面各部分的独立实现，降低了耦合，各内容分块可以单独与服务器进行通信，有效避免了冗余数据的加载。而且，综合各分块可以实现更为复杂的逻辑功能，从而提高程序性能。

2.3.1 设计原理

内容分块模式的设计原理是使页面各分块与服务器单独通信，并支持内容分块之间的互相通信，从而减小各分块之间的耦合程度；同时，该模式赋予分块组合结构一定的灵活性，程序可以根据需要动态调整页面组织结构、增删分块、调整布局等。在对页面进行内容分块设计时，需要明确各分块的功能，理清分块之间的关系，创建分块之间及其与服务器之间的通信接口，明确分块与服务器数据通信的方式。

在 AJAX 引擎中，客户端主处理程序负责控制各分块之间及其与服务器之间的通信，处理用户在分块内操作所产生的事件，实施视图结构的动态调整。界面视图由不同的分块层组成，各分块在独立实现的同

时也存在着紧密的联系。分块功能的确定、页面分块的划分以及各分块的内部实现需要视具体的应用情形而定[5]。

2.3.2 实例代码

通常情况下，可以动态注入到分块的内容主要有：HT-ML 代码段、图片 JavaScript 分块。下面分别展示相关代码：

(1) HTML 代码段

WEB 页面上的所有可视元素均通过 HTML 代码进行表现和组织，HTML 代码可以直接嵌入到分块中，产生分块的视图。需要时可以将服务器动态创建的直接注入到分块中，生成所需分块视图。这里用函数 RenderUserControl()实现了将用户控件转变成字符串形式 HTML 代码的过程，返回的 H-TML 代码可直接注入到分块内，产生指定的用户控件视图：

```
String RenderUserControl(String UserControlName)
{ //定义变量
StringBuilder sb=new StringBuilder();
System.IO.StringWriter sw=new System.IO.StringWriter(sb):  //创建系统输出流对象
HtmlTextWriter writer=new HtmlTextWriter(sw); //输出动态创建的 HTML 代码
UserControl uc=new UserControl();  //初始化一个用户控件
Control d=uc.LoadControl(UserControlName);  //加载选择的用户控件
d.RenderControl(Writer);  //将用户控件绘制出来
return.sb.Tostring();  //返回用户控件的 HTML 代码字符串
}
```

(2) 注入图片

使用 XMLHttpRequest 对象对图片进行二进制格式的传输是相当复杂的，因为 XMLHttpRequest 的属性 responseText 和 responseXML 所期待的分别是文本和 XML，而传输过程中数据已经被转换成了乱码。动态注入图片的实际做法并不需要传递图片的二进制数据，而是传递图片的引用。比如，分块进行 HTML 解析时，若传递“”这样一个字符串，便可获得对图片 abc.jpg 的引用，从而加载图片。

(3) 注入 JavaScript 分块

从服务器端发送 JavaScript 代码非常有效，因为客户端不用解析数据，只需通过程序语句执行代码即可。通常情况下，动态注入一个服务器端所创建的 JavaScript 分块的目的是操作 DOM 对象或在分块内增加新功能，客户端可以引用服务器端创建好的一段通用代码，只需执行这段代码而不必知道其实现细节。使用这种 JavaScript 分块的注入方式，代码片断将由服务器来管理，从而可以为客户端添加其在设计阶段无法具有的功能。

3 结语

文章在阐述 AJAX 技术原理及实现方式的基础上，提出了 AJAX 应用的典型设计模式，为 AJAX 应用设计提供了较为通用的设计思路和更加规范、合理的体系结构，在一定程度上简化了 AJAX 应用开发的难度，有效改善了程序的性能，为设计更加复杂的综合性 AJAX 应用程序奠定了基础。

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,2006:116－122.
2 吴吉义,平玲娣.WEB2.0 主流应用技术——AJAX 性能分析.计算机工程与设计,2008,29(8):1913－1914.
3 阳锋,徐建波. AJAX 技术的性能改进研究.计算机工程与科学,2008,30(6):147－148.
4 王沛,冯曼菲.征服 AjaxWeb2.0 开发技术详解.北京:人民邮电出版社,2006:263－272.
5 张夏天. AjaxWeb 应用的编程模型研究和应用框架实现[硕士学位论文].北京邮电大学,2007.39－41.