**Ajax**

一、基础知识

**1.定义解析**

AJAX全称为“Asynchronous JavaScript and XML”（异步JavaScript和XML），是一种创建交互式网页应用的网页开发技术。它使用：

使用XHTML+CSS来标准化呈现；

使用XML和XSLT进行数据交换及相关操作；

使用XMLHttpRequest对象与Web服务器进行异步数据通信；

使用Javascript操作Document Object Model进行动态显示及交互；

使用JavaScript绑定和处理所有数据。

**2.与传统的web应用比较**

传统的Web应用交互由用户触发一个HTTP请求到服务器,服务器对其进行处理后再返回一个新的HTHL页到客户端, 每当服务器处理客户端提交的请求时,客户都只能空闲等待,并且哪怕只是一次很小的交互、只需从服务器端得到很简单的一个数据,都要返回一个完整的HTML页,而用户每次都要浪费时间和带宽去重新读取整个页面。这个做法浪费了许多带宽，由于每次应用的交互都需要向服务器发送请求，应用的响应时间就依赖于服务器的响应时间。这导致了用户界面的响应比本地应用慢得多。与此不同，AJAX应用可以仅向服务器发送并取回必需的数据，它使用SOAP或其它一些基于XML的Web Service接口，并在客户端采用JavaScript处理来自服务器的响应。因为在服务器和浏览器之间交换的数据大量减少，结果我们就能看到响应更快的应用。同时很多的处理工作可以在发出请求的客户端机器上完成，所以Web服务器的处理时间也减少了。Ajax使WEB中的界面与应用分离，而在以前两者是没有清晰的界限的，数据与呈现分离的分离，有利于分工合作、减少非技术人员对页面的修改造成的WEB应用程序错误、提高效率、也更加适用于现在的发布系统。也可以把以前的一些服务器负担的工作转嫁到客户端，利于客户端闲置的处理能力来处理。

## 二、核心技术

**1 AJAX的工作原理以及代码**

Ajax的工作原理相当于在用户和服务器之间加了—个中间层(AJAX引擎),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—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。**Ajax的一个最大的特点是无需刷新页面便可向服务器传输或读写数据(又称无刷新更新页面),这一特点主要得益于XMLHTTP组件XMLHTTPRequest对象。**

**2 XMLHttpRequest 对象方法描述**

**abort()** 停止当前请求

**getAllResponseHeaders()** 把HTTP请求的所有响应首部作为键/值对返回

**getResponseHeader("header")** 返回指定首部的串值

**open("method","URL",[asyncFlag],["userName"],["password"])**  建立对服务器的调用。method参数可以是GET、POST或PUT。url参数可以是相对URL或绝对URL。这个方法还包括3个可选的参数，是否异步，用户名，密码

**send(content)** 向服务器发送请求

**setRequestHeader("header", "value")** 把指定首部设置为所提供的值。在设置任何首部之前必须先调用open()。设置header并和请求一起发送 ('post'方法一定要 )

**onreadystatechange** 状态改变的事件触发器，每个状态改变时都会触发这个事件处理器，通常会调用一个JavaScript函数

**readyState** 请求的状态。有5个可取值：0 = 未初始化，1 = 正在加载，2 = 已加载，3 = 交互中，4 = 完成

responseText服务器的响应，返回数据的文本。

**responseXML** 服务器的响应，返回数据的兼容DOM的XML文档对象 ，这个对象可以解析为一个DOM对象。

**responseBody**  服务器返回的主题（非文本格式）

**responseStream** 服务器返回的数据流

**status** 服务器的HTTP状态码（如：404 = "文件末找到" 、200 ="成功" ，等等）

**statusText** 服务器返回的状态文本信息 ，HTTP状态码的相应文本（OK或Not Found（未找到）等等）

**3 调用ajax方法**

AJAX

( "POST",

"/shared/ete2/submitschool.asp",

"xml="+ encodeURIComponent(xml.xml),

function(msg)

{

var f = window.document.frames["bgPage"];

f.document.write("<body>" + msg.xml.toString() + "</body>");

healthCheckNew();

}

);

function AJAX(type, url, data, callback)

{

var request = false;

try

{

//window对象中有XMLHttpRequest存在就是非IE，包括（IE7，IE8）

request = new XMLHttpRequest();

} catch (trymicrosoft)

{

try

{

request = new ActiveXObject("Msxml2.XMLHTTP");

} catch (othermicrosoft)

{

try

{

request = new ActiveXObject("Microsoft.XMLHTTP");

} catch (failed)

{

request = false;

}

}

}

if (!request)

return;

request.open(type, url, true);

request.setRequestHeader("CONTENT-TYPE","application/x-www-form-urlencoded")

request.setRequestHeader("Content-Length", data.length);

request.onreadystatechange = function()

{

if (request.readyState == 4 && request.status == 200)

{

callback(request.responseXML);

}

};

request.send(data);

}

**4 AJAX的优点**

**<1>.无刷新更新数据。**避免了在网络上发送那些没有改变的信息，减少用户等待时间，带来非常好的用户体验。

**<2>.异步与服务器通信。**不需要打断用户的操作，减少不必要的数据传输、时间及降低网络上数据流量。

**<3>.前端和后端负载平衡。**AJAX可以把以前一些服务器负担的工作转嫁到客户端，提升站点性能。

**<4>.基于标准被广泛支持。**基于标准化的并被广泛支持的技术，不需要下载浏览器插件或者小程序。

<**5>.界面与应用分离。**也可以说是数据与呈现分离

**5 AJAX的缺点**

**<1>.AJAX干掉了Back和History功能，即对浏览器机制的破坏。**

在动态更新页面的情况下，用户无法回到前一个页面状态，因为浏览器仅能记忆历史记录中的静态页面。一个被完整读入的页面与一个已经被动态修改过的页面之间的差别非常微妙；

用户通常会希望单击后退按钮能够取消他们的前一次操作，但是在Ajax应用程序中，这将无法实现。后退按钮是一个标准的web站点的重要功能，但是它没法和js进行很好的合作。

**<2>.AJAX的安全问题。**如同对企业数据建立了一个直接通道。诸如跨站点脚步攻击、SQL注入攻击和基于Credentials的安全漏洞等等。

**<3>.对搜索引擎支持较弱。**如果使用不当，AJAX会增大网络数据的流量，从而降低整个系统的性能。

**<4>.破坏程序的异常处理机制**。像Ajax.dll，Ajaxpro.dll这些Ajax框架是会破坏程序的异常机制的。

**<5>.违背URL和资源定位的初衷。**一个URL地址，如果采用了Ajax技术，也许你在该URL地址下面看到的和我在这个URL地址下看到的内容是不同的。

**<6>.AJAX不能很好支持移动设备。**一些手持设备（如手机、PDA等）现在还不能很好的支持Ajax

**<7>.客户端过肥，**太多客户端代码造成开发上的成本。编写复杂、容易出错 ；

**6 AJAX适用场景**

<1>.表单驱动的交互

<2>.深层次的树的导航

<3>.快速的用户与用户间的交流响应

<4>.类似投票、yes/no等无关痛痒的场景

<5>.对数据进行过滤和操纵相关数据的场景

<6>.普通的文本输入提示和自动完成的场景

**7 AJAX不适用场景**

<1>.部分简单的表单

<2>.搜索

<3>.基本的导航

<4>.替换大量的文本

<5>.对呈现的操纵

**8 跨域问题处理**

这里说的js跨域是指通过js在不同的域之间进行数据传输或通信，比如用ajax向一个不同的域请求数据，或者通过js获取页面中不同域的框架中(iframe)的数据。只要协议、域名、端口有任何一个不同，都被当作是不同的域。

**（1）jsonp跨域**

在js中，我们直接用XMLHttpRequest请求不同域上的数据时，是不可以的。但是，在页面上引入不同域上的js脚本文件却是可以的，jsonp正是利用这个特性来实现的。

<script src="http://www.baidu.com/js/XXX.js?callback=receiveResult"/>

所以通过http://example.com/data.php?callback=dosomething得到的js文件，就是我们之前定义的dosomething函数,并且它的参数就是我们需要的json数据，这样我们就跨域获得了我们需要的数据。这样jsonp的原理就很清楚了，通过script标签引入一个js文件，这个js文件载入成功后会执行我们在url参数中指定的函数，并且会把我们需要的json数据作为参数传入。所以jsonp是需要服务器端的页面进行相应的配合的。知道jsonp跨域的原理后我们就可以用js动态生成script标签来进行跨域操作了，而不用特意的手动的书写那些script标签。如果你的页面使用jquery，那么通过它封装的方法就能很方便的来进行jsonp操作了。

**（2）Nginx反向代理跨域**

nginx.conf 配置一个反向代理路径：

server {

listen 8094;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

root html;

index index.html index.htm;

}

location /apis

{

rewrite ^.+apis/?(.\*)$ /$1 break;

include uwsgi\_params;

proxy\_pass http://localhost:1894;

}

}

配置说明：配置一个/apis 重写到我们真正的api地址http://localhost:1894 形成一个代理的过程。然后更改一下index1.html的api请求地址。

**（3）使用window.name来进行跨域**

窗口载入的所有的页面都是共享一个window.name的，每个页面对window.name都有读写的权限，当然，如果有需要，其中的任何一个页面都可以对window.name的值进行修改。

注意，window.name的值只能是字符串的形式，这个字符串的大小最大能允许2M左右甚至更大的一个容量，具体取决于不同的浏览器，但一般是够用了。上面的例子中，我们用到的页面a.html和b.html是处于同一个域的，但是即使a.html与b.html处于不同的域中，上述结论同样是适用的，这也正是利用window.name进行跨域的原理。

**9 DWR**

DWR(直接web远程访问)，DWR是一个Java库,使服务器上的Java和JavaScript的浏览器进行交互和相互调用尽可能简单。一个可以允许你去创建 AJAX WEB 站点的 Java 开源库。它可以让你在浏览器中的 JavaScript 代码调用 Web 服务器上的 Java 代码，就像在 Java 代码就在浏览器中一样。该技术基本被淘汰，被jquery取代。具体可以看优迈的教程。

**10 常见状态码**

100 Continue

初始的请求已经接受，客户应当继续发送请求的其余部分

101 Switching Protocols

服务器将遵从客户的请求转换到另外一种协议

200 OK

一切正常，对GET和POST请求的应答文档跟在后面

201 Created

服务器已经创建了文档，Location头给出了它的URL。

202 Accepted

已经接受请求，但处理尚未完成。

203 Non-Authoritative Information

文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝

204 No Content

没有新文档，浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而Servlet可以确定用户文档足够新，这个状态代码是很有用的

205 Reset Content

没有新的内容，但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容

206 Partial Content

客户发送了一个带有Range头的GET请求，服务器完成了它

300 Multiple Choices

客户请求的文档可以在多个位置找到，这些位置已经在返回的文档内列出。如果服务器要提出优先选择，则应该在Location应答头指明。

301 Moved Permanently

客户请求的文档在其他地方，新的URL在Location头中给出，浏览器应该自动地访问新的URL。

302 Found

类似于301，但新的URL应该被视为临时性的替代，而不是永久性的。

303 See Other

类似于301/302，不同之处在于，如果原来的请求是POST，Location头指定的重定向目标文档应该通过GET提取

304 Not Modified

客户端有缓冲的文档并发出了一个条件性的请求（一般是提供If-Modified-Since头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。

305 Use Proxy

客户请求的文档应该通过Location头所指明的代理服务器提取

307 Temporary Redirect

和302（Found）相同。许多浏览器会错误地响应302应答进行重定向，即使原来的请求是 POST，即使它实际上只能在POST请求的应答是303时才能重定向。由于这个原因，HTTP 1.1新增了307，以便更加清除地区分几个状态代码： 当出现303应答时，浏览器可以跟随重定向的GET和POST请求；如果是307应答，则浏览器只能跟随对GET请求的重定向。

400 Bad Request

请求出现语法错误。

401 Unauthorized

客户试图未经授权访问受密码保护的页面。应答中会包含一个WWW-Authenticate头，浏览器据此显示用户名字/密码对话框，然后在填写合适的Authorization头后再次发出请求。

403 Forbidden

资源不可用。

404 Not Found

无法找到指定位置的资源

405 Method Not Allowed

请求方法（GET、POST、HEAD、Delete、PUT、TRACE等）对指定的资源不适用。

406 Not Acceptable

指定的资源已经找到，但它的MIME类型和客户在Accpet头中所指定的不兼容

407 Proxy Authentication Required

类似于401，表示客户必须先经过代理服务器的授权。

408 Request Timeout

在服务器许可的等待时间内，客户一直没有发出任何请求。客户可以在以后重复同一请求。

409 Conflict

通常和PUT请求有关。由于请求和资源的当前状态相冲突，因此请求不能成功。

410 Gone

所请求的文档已经不再可用，而且服务器不知道应该重定向到哪一个地址。它和404的不同在于，返回407表示文档永久地离开了指定的位置，而404表示由于未知的原因文档不可用。

411 Length Required

服务器不能处理请求，除非客户发送一个Content-Length头。

412 Precondition Failed

请求头中指定的一些前提条件失败

413 Request Entity Too Large

目标文档的大小超过服务器当前愿意处理的大小。如果服务器认为自己能够稍后再处理该请求，则应该提供一个Retry-After头

414 Request URI Too Long

URI太长

416 Requested Range Not Satisfiable

服务器不能满足客户在请求中指定的Range头

500 Internal Server Error

服务器遇到了意料不到的情况，不能完成客户的请求

501 Not Implemented

服务器不支持实现请求所需要的功能。例如，客户发出了一个服务器不支持的PUT请求

502 Bad Gateway

服务器作为网关或者代理时，为了完成请求访问下一个服务器，但该服务器返回了非法的应答

503 Service Unavailable

服务器由于维护或者负载过重未能应答。例如，Servlet可能在数据库连接池已满的情况下返回503。服务器返回503时可以提供一个Retry-After头

504 Gateway Timeout

由作为代理或网关的服务