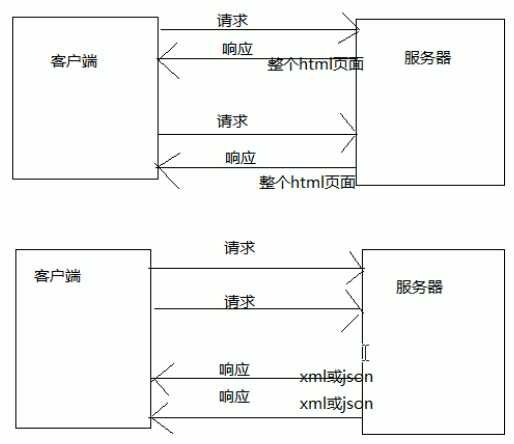
# Ajax

## Ajax是啥？

异步的 js 和 xml .

* 能使用 js 访问服务器，而且是异步访问；
* 服务器给客户端响应的一般是整个页面，一个 html 完整的页面！但是在 ajax 中因为是局部刷新，那么服务器就不用在响应整个页面！而只是数据！
* 数据交换格式：
* Text:纯文本；
* Xml:
* Json: js 提供的数据交互格式，在 ajax 中最受欢迎！
* 示意图



* 应用场景
* 百度的搜索框 下拉提示
* 用户注册时，实时校验用户名是否已被注册。
* 优点
* 异步交互，增强了用户体验；
* 性能：因为服务器无需在响应整个页面，只需要响应部分内容，所以服务器的压力减轻了；
* 缺点
* Ajax 不能应用在所有场景；
* Ajax 无端的增多了对服务器的访问次数，又给服务器增加了压力！

### 同步交互和异步交互

* 同步
* 发一个请求，要等待服务器的响应结束，然后才能发送第二个请求，中间的时间处于“卡”的阶段。
* 刷新的是整个页面。
* 异步
* 发送一个请求，无需等待服务器的响应，然后就可以发送第二个请求！
* 可以使用 js 接受服务器的响应，然后使用 js 进行局部刷新。

## Ajax 发送异步请求（四步）

### 第一步（得到 XMLHttpRequest）

* Ajax 只需要学习一个对象：XMLHttpRequest,如果掌握了它，就掌握了 ajax;
* 得到 XMLHttpRequest
* 大多数浏览器都支持 var XMLHttp = new XMLHttpRequest();
* IE6.0: var XMLHttp = new ActiveXObject(“Msxml2.XMLHTTP”);
* IE5.5以及更早版本的IE: var xmlHttp = new ActiveXObject(“Microsoft.XMLHTTP”);
* 编写创建 XMLHttpRequest 对象的函数：

Function createXMLRequest(){

Try{

Return new XMLHttpRequest();

}catch{

Try{

Return new ActiveXObject(“Msxml2.XMLHTTP”);

}catch{

Try{

Return new ActiveXObject(“Microsoft.XMLHTTP”);

}catch{

Alert(“您用的是啥浏览器啊？”)；

Throw e;

}

}

}

}

### 第二步（打开与服务器的连接）

* xmlHttp.open():用来打开与服务器的连接，需要三个参数：
* 请求的方式：GET/POST;
* 请求的URL:指定服务器端的资源，例如：/project/servlet;
* 请求是否为异步：如果为true表示发送异步请求，否则为同步请求；
* xmlHttp.open(“GET”,” /project/servlet”,true);

### 第三步（发送请求）

* xmlHttp.send(null)

如果不给参数可能会造成部分浏览器无法发送；

* 参数：就是请求体的内容！如果是 GET 请求，必须给出 null.

### 第四步（得到响应）

* 在 xmlHttp 对象的一个事件上注册监听器:onreadystatechange
* xmlHttp 对象一共有 5 个状态：
* 0：刚创建，还没有调用 open() 方法；
* 1：请求开始，调用了 open() 方法，但是好没有调用 send() 方法；
* 2：调用完 send() 方法了；
* 3：服务器已经开始响应了，但不表示响应已经结束了；
* 4：服务器响应结束！（通常我们只关心这个状态！）
* 得到 xmlHttp 对象的状态
* Var state = xmlHttp.readyState; // 结果可能为0，1，2，3，4 共五种状态
* 得到服务器响应的状态码
* Var status = xmlHttp.status; //eg:200,404,500
* 得到服务器响应的内容
* Var content = xmlHttp.responseText; // 获取服务器响应的文本格式的内容
* Var content = xmlHttp.responseXML; // 获取服务器响应的xml格式的内容，xml会被转化为 document 对象

### 得到响应的代码

xmlHttp.onreadystatechange = function(){

if(xmlHttp.state == 4 && xmlHttp.status == 200){

//服务器响应成功

//获取服务器的响应内容

Var content = xmlHttp.responseText;

}

};

### 发送 POST 请求

如果发送请求时需要带有参数，一般都用 POST 请求。

其与 GET 请求的区别主要在：

* Open:xmlHttp.open(“open”,…);
* 添加一步：添加 Content-Type 请求头：
* xmlHttp.setRequestHeader(“Content-Type”,”application/x-www-form-urlencoded”);
* 发送请求时要指定请求体：
* xmlHttp.send(“username=xxx&password=123”);

## 注册表单校验用户名是否被注册

* 注册页面
* 给出注册表单页面
* 给用户名文本框添加 onblur 事件的监听
* 获取文本框的内容，通过 ajax 4步发送给服务器，得到响应结果

\*- 如果为1 ：在文本框之后显示“用户名已被注册”；

\*- 为 0 ：在文本框之后显示 “”；

* Servlet
* 获取客户端传递的用户名参数；
* 判断其值是否为 itcast;

\*- 是：返回1 ；

\*- 否：返回0 ；

## 响应内容为 xml 数据

* 服务器端
* 设置响应头 Content-Type,其值为 “text/xml;charset=utf-8”;
* 客户端

获取 document 对象

* Var doc = xmlHttp.responseXML;

解析 doc 并进行输出！

## 省市联动

### 页面

<select name = “province”>

<option>====请选择省份====</option>

</servlet>

<select name = “city”>

<option>====请选择城市====</option>

</servlet>

### ProvinceServlet

显示省份，在页面加载完毕之后马上请求这个 Servlet!

需要服务器将所有省份的名称使用字符串发送给客户端！

### 页面的工作

* 获取此字符串，使用逗号分隔，得到数组
* 循环遍历这个字符串（省份的名称），使用每个字符串创建一个<option>元素添加到

<select name = “province”> 下面

### CityServlet

* 当页面选择某一个省时，发送请求！
* 得到省份的名称，加载xml 文件，查询出该省份对应的元素对象！并将元素转化为 xml 字符串，发送给客户端；

### 页面的工作

* 加载之前，将 <select name = “city”> 除第一个以外的所有子元素删除；
* 获取服务器的响应结果：doc!
* 获取所有的 city 子元素，循环遍历，得到 <city> 的内容；
* 使用每一个 <city> 的内容创建一个 <option> 元素，并添加到

<select name = “city”> 中。

### 形象化

