**培训要求**

**1）回顾JS中核心内容**

**2）了解WEB1.0和WEB2.0时代的技术与特点**

**3）理解AJAX的产生背景、工作原理与特点**

**4）掌握AJAX常用API及应用**

**声明：今天服务端我们使用Servlet技术**

**一）什么是JavaScript【以下简称JS】**

JS是

（1）基于对象

JS本身就有一些现成的对象可供程序员使用，例如：Array，Math，String。。。

JS并不排除你可以自已按一定的规则创建对象

（2）事件驱动

JS代码写好后，需要外界触发后，方可运行，例如：单击事件，定时执行，。。。

（3）解释性

每次运行JS代码时，得需要将原代码一行一行的解释执行

相对编译型语言（例如：Java、C++）执行速度相对较慢

（4）基于浏览器的动态交互网页技术

如果JS嵌入到HTML中，可以不需要服务器支持，直接由浏览器解释执行

如果JS嵌入到JSP或Servlet中，必需要服务器支持，直接由浏览器解释执行

（5）嵌入在HTML标签中

JS必须嵌入到一个名叫<script src="XXX.js"></script>的标签中，方可运行

脚本语言

**二）JS中的三种类型**

（1）基本类型：number，string，boolean

number包含正数，负数，小数

string由''或""定界

boolean由true或false，但js中的boolean也包含更多情况，例如：存在表示true，不存在表示false

|  |
| --- |
| var num = 100;  var str = "哈哈";  var flag = false;  window.alert(num);  window.alert(str);  window.alert(flag); |

（2）特殊类型：null，undefined

null表示一个变量指向null

undefined表示一个变量指向的值不确定

|  |
| --- |
| var array = null;  var student;  alert(array);  alert(student); |

（3）复合类型：函数，对象，数组

对象包含内置对象和自定义的对象

**三）JS中有三种定义函数的方式**

（1）正常方式：function mysum(num1,num2){return num1+num2;}

|  |
| --- |
| function mysum(num1,num2){  return num1 + num2;  }  var myresult = mysum(100,200);  alert("myresult="+myresult); |

（2）构造器方式：new Function("num1","num2","return num1+num2;")

|  |
| --- |
| 每个参数都是字符串类型 最后一个是函数体  var youresult = new Function("num1","num2","return num1+num2");  alert( youresult(1000,2000) ); |

（3）直接量或匿名或无名方式：var mysum = function(num1,num2){return num1+num2;}

|  |
| --- |
| var theyresult = function(num1,num2){  return num1 + num2;  }  alert( theyresult(10000,20000) ); |

**四）JS中有四种对象**

（1）内置对象(本身就有的对象) ：Date，Math，String，Array，。。。

|  |
| --- |
| var str = new Date().toLocaleString();  浏览器的窗口.白色区.显示（）  window.document.write("<font size='44' color='red'>"+str+"</font>"); |

（2）自定义对象：Person，Card，。。。

|  |
| --- |
| function Student(id,name,sal){  //this指向s引用  this.id = id;  this.name = name;  this.sal = sal;  }  var s = new Student(1,"波波",7000);  document.write("编号:" + s.id + "<br/>");  document.write("姓名:" + s.name + "<br/>");  document.write("薪水:" + s.sal + "<br/>"); |

（3）浏览器对象：

window(浏览器)，document(白色区域)，

status(状态栏)，location(地址栏)，history(历史栏 前进 后退)。。。

|  |
| --- |
| function myrefresh(){  window.history.go(0); 0 刷新 -1 后退 1前进  } |

（4）ActiveX对象：ActiveXObject("Microsoft.XMLHTTP")，。。。

A tifu X

**五）演示JS对象的属性，方法和事件的使用**

（1）window.location.href

|  |
| --- |
| var url = "04\_array.html";  window.location.href = url; |

（2）form.submit()

|  |
| --- |
| <form action=*"/js-day01/04\_array.html"* method=*"POST"*>  <input type=*"button"* value=*"提交到服务端"* onclick="doSubmit()"/>  <!-- button 可以对按钮修饰 和 判断 -->  </form>    <script type=*"text/javascript"*>  //定位提交按钮，同时添加点击事件 注意：获取(标签名)方法中  如上面 form中 input标签 可能有多个 默认第一个是0  **var** inputElement= document.getElementByTagName("input")[0]  //为提交按钮添加点击事件  inputElement.conclick = **function**(){  //定位<form>标签 forms表示document对象中所有表单的集合，  通过下标应用不用的表单，0开始  **var** form = document.forms[0];  //表单提交，提交到action属性指定的地方  form.submit();  }  /\* 或者  function doSubmit(){  //表单提交  document.forms[0].submit();  }  \*/  </script> |

（3）inputElement.onblur = 函数

（4）document.createElement(“img”)

（5）imgElement.style.width/height

**六）回顾传统Web应用请求和响应特点【显示当前时间】**

（1）请求：浏览器以HTTP协议的方式提交请求到服务器

（2）响应：服务器以HTTP协议的方式响应内容到浏览器

注意：HTTP是WEB大众化非安全协议HTTPS是WEB安全协议，

是基于HTTP协议的，且加了一些加密等特殊功能，常用于在线支付，

或者是需要安全性较高的网站中，

例如：12306网站

HTTP请求有三个部份组成：请求行，请求头，请求体

HTTP响应有三个部份组成：响应行，响应头，响应体

（3）状态栏：有明显的进度条刷新现象，如果服务器响应较慢的话，进度条刷新

也会变慢，IE9等中高版本浏览器，有明显转圈圈图标

（4）历史栏：会收集原来已访问过的web页面，进行缓存

（5）缺点：不需变化的大量数据，也全部刷新，造成浏览器加载和处理负担

（6）可改进的地方：让不需变化的大量数据，原封不动，不用缓存到历史栏中，

无需全部刷新，只刷新某些需要变化的数据区域，例如：当前时间的区域

|  |
| --- |
| <当前时间:<span>${requestScope.str}</span><br/>//找到就返回中没有事空字符  <input id=*"buttonID"* type=*"button"* value=*"获取当前时间"*/> |

|  |
| --- |
| <script type=*"text/javascript"*>  //定位按钮，同时添加单击事件  //法一：定位第0个input标签  //document.getElementsByTagName("input")[0].onclick = function(){  //法二：定位指定ID标签  document.getElementById("buttonID").onlick = **function**(){ // 当前时间的毫秒数  **var** url = "${pageContext.request.contextPath}/TimeServlet?id="+**new** Date().getTime();  window.location.href = url;  }  </script> |

|  |
| --- |
| **public** **class** TimeServlet **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  //测试  System.***out***.println("GET");  //构造SimpleDateFormat对象  SimpleDateFormat sdf = **new**  SimpleDateFormat("yyyy-MM-dd HH:mm:ss");    //将当期日期按照指定格式输出成字符串  String str = sdf.format(**new** Date());  //将结果绑定到request对象中  request.setAttribute("str",str);    //转发到指定页面  request.getRequestDispatcher(""index.jsp"")  .forward(request,response);  }} |

**七）什么是AJAX【Asynchronous异步的JS和XML】，工作原理与特点**

（1）什么是同步：

请求1->响应1->请求2->响应2->

Web1.0时代

（2）什么是异步：

请求1->请求2->请求3->响应1->响应2->响应3->

请求1->响应1->请求2->请求3->响应2->响应3->

Web2.0时代

项目中：Web1.0为主（整个浏览器刷新），Web2.0为辅（浏览器局部刷新）

（3）什么是AJAX

客户端（特指PC浏览器）与服务器，可以在【不必刷新整个浏览器】的情况下，

与服务器进行异步通讯的技术 即，AJAX是一个【局部刷新】的【异步】通讯技术

AJAX不是全新的语言，是2005年Google公司推出的一种全新【编程模式】，

不是新的编程语言

接收到的请求 ->交给AJAX处理-> servlet ->以输出流的形式返回 -> AJAX执行

（4）不用刷新整个页面便可与服务器通讯的办法有：

（A）Flash/ActionScript

（B）框架Frameset

（C）iFrame（内嵌入框架)

（D）XMLHttpRequest(非IE浏览器)和ActiveXObject(IE浏览器)

背景：早上IE5时，微软就开发出了第一个异步通讯对象，

叫ActiveXObject对象，Firefox等其它浏览器厂商也慢慢引

入异步通讯对象，叫XMLHttpRequest对象，IE的高版本，也

将这个异步对象取名叫XMLHttpRequest对象，但IE有向下

兼容问题，也可以使用ActiveXObject对象。无需第三方jar包，

现代中高版本浏览器中内置了这个异步通讯对象，只需通过

JavaScript就可以创建

注意：所有浏览器中都内置了异步对象，在默认情况下，该异步对象并没有创建出来

|  |
| --- |
| function createAJAX(){  var ajax = null;  try{  ajax = new ActiveXObject("microsoft.xmlhttp");  }catch(e1){  ajax = new XMLHttpRequest();  }  return ajax;  } |

（5）AJAX工作原理

参见<<AJAX工作原理.JPG>>

（6）AJAX包含的技术

参见<<AJAX包含的技术.JPG>>

（7）AJAX开发步骤

步一：创建AJAX异步对象，例如：createAJAX()

步二：准备发送异步请求，例如：ajax.open(method,url)

步三：如果是POST请求的话，一定要设置AJAX请求头，

例如：ajax.setRequestHeader()

如果是GET请求的话，无需设置设置AJAX请求头

步四：真正发送请求体中的数据到服务器，例如：ajax.send()

步五：AJAX不断的监听服务端响应的状态变化，

例如：ajax.onreadystatechange，后面写一个无名处理函数

步六：在无名处理函数中，获取AJAX的数据后，按照DOM规则，用JS语言

来操作Web页面

（8）AJAX适合用在什么地方

AJAX【适合】不用来传递大量数据，而只用来【传递少量数据】，在用户的【体验】上，【更加人性化】

AJAX是一个和服务器无关的技术，即服务器可使用：JavaEE，.NET，PHP，。。。这些技术都可

AJAX只管向服务器发送请求，同时只管接收服务器的HTML或XML或JSON载体响应

服务端不能使用转发或重定向到web页面，因为这样会起浏览器全面刷新

即只能以流的方式响应给浏览器

**八）AJAX应用**

（1）无需刷新整个Web页面显示服务器响应的当前时间

（text/html;charset=UTF-8）

|  |
| --- |
| 当前时间:<span></span><br/>  <input type="button" value="异步方式提交"/> |

|  |
| --- |
| <script type="text/javascript">  //定位button按钮，同时添加单击事件  document.getElementsByTagName("input")[0].onclick = function(){  //NO1)创建AJAX异步对象（每个浏览器内置的，无需第三方jar包）  var ajax = createAJAX();//0  //NO2)AJAX异步对象准备发送请求  var url = "${pageContext.request.contextPath}/TimeServletAjax?id="+new Date().getTime();  var method = "GET";  ajax.open(method,url);//1  //NO3）AJAX异步对象真正发送请求体的数据到服务器，如果请求体无数据的话，用null表示  var content = null;  ajax.send(content);//2    //----------------------------------------等待    //NO4）AJAX异步对象不断监听服务端状态的变化，只有状态码变化了，方可触发函数  //0-1-2-3-4,这些是可以触发函数的  //4-4-4-4-4，这些是不可以触发函数的  //以下这个函数是服务器来触发的，不是程序员触发的，这和onclick是不一样的  ajax.onreadystatechange = function(){  //如果AJAX状态码为4  if(ajax.readyState == 4){  //如果服务器响应码是200  if(ajax.status == 200){  //NO5）从AJAX异步对象中获取服务器响应的结果  var str = ajax.responseText;  //NO6）按照DOM规则，将结果动态添加到web页面指向的标签中  document.getElementsByTagName("span")[0].innerHTML = str;  }  }  }  }  </script> |

|  |
| --- |
| **public** **class** TimeServletAjax **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)**throws** ServletException, IOException {  SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  String str = sdf.format(**new** Date());  //注意：在Web2.0时代，即异步方式下，不能用转发或重定向  //因为：转发或重定向会引起浏览器全部刷新，而不是局部刷新  //所以得用以输出流的方式将服务器的结果输出到浏览器  response.setContentType("text/html;charset=UTF-8");  PrintWriter pw = response.getWriter();  pw.write(str);  pw.flush();  pw.close();  }  } |

（2）基于HTML，以GET或POST方式，检查注册用户名是否在数据库中已存在（text/html;charset=UTF-8）

|  |
| --- |
| <body>  <from>  用户名[GET]:<input id =*"usernameID"* type=*"text"* name=*"username"* maxlength=*"4"/*> <!--不能多多余4个字符-->  光标移出后，立即检测结果  </from>  <hr/>  <!-- 用于定位输出 -->  <span id=*"resID"*> xxx </span>  <script type=*"text/javascript"*>  //创建AJAX异步对象 即 XMLHttpRequest  **function** createAJAX(){  **var** ajax = **null**;  **try**{ // a tie X Obj  ajax = **new** ActiveXObject("microsoft.xmlhttp");  }**catch**(e1){  **try**{  ajax = **new** XMLHttpRequest();  }**catch**(e2){  alter("浏览器不支持，请更换");  }  }  **return** ajax;  }  </script>  <script type=*"text/javascript"*>  //定位上面的文本框，同时提供 焦点失去事件（onblur）:鼠标移开就执行  document.getElementById("usernameID").onblur = **function**(){  //获取文本框中输入的值  **var** username = this.value;  //判断框是否有输入  **if**(username.length == 0 ){  document.getElementById("resID").innerHTML = "用户名必填";// innerHTML 用于取代字符 即取代“resID”中的字符  }**else**{  //后面有汉字出现 要解决编码  username = encodeURI(username); //对指定数据 进行UTF-8编码    //-1-  **var** ajax = createAJAX();  //-2-  **var** method = "GET";  **var** url = "${pageContext.request.contextPath}/UserServlet?time="+**new** Date().getTime()+"&username=" + username;    ajax.open(method,url);    //-3-正式发送  ajax.send(**null**);    //-4-  ajax.onreadystatechange = **function**(){  // 是否为第四步  **if**(ajax.readyState == 4){ //是否为AJAX的第4不  //确保不是 200以外的页面  **if**(ajax.status == 200){  //-5-  **var** tip = ajax.responseText;//responseText :服务端用的 是 html    //-6- 替换内容  document.getElementById("resID").innerHTML = tip;  }  }  }  }  }  </script>  </body> |

|  |
| --- |
| **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)**throws** ServletException, IOException {  //接收jsp中的请求  String username = request.getParameter("username");  /\*\* 判断username中 接收的字符形式 如果是ISO8859-1 特定的转换  byte[] buf = username.getBytes("ISO8859-1");  // 将buf 按 UTF-8 的方式 重新构造一遍  username = new String(buf,"UTF-8");  \*/    System.***out***.println("username=" + username);  String tip = "<font color='green'>可以注册</font>"; //绿  **if**("杰克".equals(username)){  tip = "<font color='red'>该用户已存在</font>";//red 红  }  response.setContentType("text/html;charset=UTF-8");  PrintWriter pw = response.getWriter();  pw.write(tip);  pw.flush();//刷新  pw.close();  } |

（3）基于XML，以POST方式，完成省份-城市二级下拉联动

（text/xml;charset=UTF-8）

Js

|  |
| --- |
| //创建AJAX雨布对象，即 XMLHttpRequest  **function** createAJAX(){  **var** ajax = **null**;  **try**{  ajax = **new** ActiveXObject("microsoft.xmlhttp");  }**catch**(e1){  **try**{  ajax = **new** XMLHttpRequest();  }**catch**(e2){  alate("您的浏览器版本过低，请更换！");  }  }  **return** ajax;  } |

JSP

|  |
| --- |
| <head>  <script type=*"text/javascript"* src=*"js/ajas.js"*></script>  </head>  <body>  <select id=*"provinceID"* style="width:*111px*; ">  <option>请选择省份</option>  <option>广东</option>  <option>湖南</option>  </select>  &nbsp;&nbsp;&nbsp;  <select id=*"cityID"* style="width: *111px*; ">  <option>选择城市</option>  </select>    <script type=*"text/javascript"*> |

Servlet

|  |
| --- |
| **public** **class** ProvinceCityServlet **extends** HttpServlet {  **public** **void** doPost(HttpServletRequest request, HttpServletResponse response  )**throws** ServletException, IOException {  //post提交特有 请求头的测试 并能进行指定编码  request.setCharacterEncoding("UTF-8");  //接收 jsp 传过来的值  String province = request.getParameter("province");    //牢记 通知AJAX异步对象1，服务器响应的数据为xml格式  response.setContentType("text/xml;charset=UTF-8");  //获取字符输出流  PrintWriter pw = response.getWriter();      //实则 返回xml页面 （注意是返回到xml页面）  pw.write("<?xml version='1.0' encoding='UTF-8'?>");  pw.write("<root>");  **if**("广东".equals(province)){  pw.write("<city>广州</city>");  pw.write("<city>深圳</city>");  pw.write("<city>珠海</city>");  }**else** **if**("湖南".equals(province)){  pw.write("<city>长沙</city>");  pw.write("<city>株洲</city>");  pw.write("<city>湘潭</city>");  pw.write("<city>岳阳</city>");  }    pw.write("</root>");  pw.flush();  pw.close();  }  } |

（4）验证码检查 (struts2 应用)

（text/html;charset=UTF-8）

|  |
| --- |
| <head>  <title>验证码检查</title>  <script type=*"text/javascript"* src=*"js/ajax.js"*></script>  </head>  <body>    <form>  <table border=*"0"* align=*"center"*>  <tr>  <th>验证码：</th>  <td><input size=*"2"* type=*"text"* name=*"checkcode"* id=*"checkcodeID"* maxlength=*"4"*/></td>  <td><img src=*"01\_image.jsp"*/> <!-- 超链接的形式 -->  <td id=*"res"*></td>  </tr>  </table>  </form>    <script type=*"text/javascript"*>  //去掉两边的空格  **function** trim(str){" zhaojun "  //格式：/ ^(左侧) \s\*(多个空格) / 用""替代 &(右侧)  str = str.replace(/^\s\*/,"");  str = str.replace(/\s\*$/,"");//"zhaojun"  **return** str;  }  </script>    <script type=*"text/javascript"*> //键盘弹起  document.getElementById("checkcodeID").onkeyup = **function**(){  **var** checkcode = this.value;  checkcode = trim(checkcode);//2525  **if**(checkcode.length == 4){  //NO1)  **var** ajax = createAJAX();  //NO2)  **var** method = "POST"; //checkRequest struts.XML的 映射 name  **var** url = "${pageContext.request.contextPath}/checkRequest?time="+**new** Date().getTime();  ajax.open(method,url);  //NO3)  ajax.setRequestHeader("content-type","application/x-www-form-urlencoded");  //NO4)  **var** content = "checkcode=" + checkcode;  ajax.send(content);    //----------------------------等待    //NO5)  ajax.onreadystatechange = **function**(){  **if**(ajax.readyState == 4){ //输入的长度  **if**(ajax.status == 200){  //NO6)  **var** tip = ajax.responseText;    //NO7)  **var** img = document.createElement("img");  img.src = tip;  img.style.width = "14px";  img.style.height = "14px";  **var** td = document.getElementById("res");  td.innerHTML = "";  td.appendChild(img);  }  }  }  }**else**{ // 当输入正确 但有删除 数字时  //清空图片  **var** td = document.getElementById("res");  td.innerHTML = "";  }  }  </script>  </body> |

|  |
| --- |
| /\*\*  \* 验证码检查  \* **@author** AdminTC  \*/  **public** **class** CheckcodeAction **extends** ActionSupport{  //客户端验证码  **private** String checkcode;//2525  //注入客户端验证码  **public** **void** setCheckcode(String checkcode) {  **this**.checkcode = checkcode;  }  /\*\*  \* 验证  \*/  **public** String check() **throws** Exception {  //图片路径  String tip = "images/MsgError.gif";  //从服务器获取session中的验证码  String checkcodeServer = (String) ActionContext.*getContext*().getSession().get("CHECKNUM");  //将客户端的验证码与服务端的验证码进 行比较  **if**(checkcode.equals(checkcodeServer)){  tip = "images/MsgSent.gif";  }  //以IO流的方式将tip变量的值输出到AJAX异步对象中  HttpServletResponse response = ServletActionContext.*getResponse*();  response.setContentType("text/html;charset=UTF-8");  PrintWriter pw = response.getWriter();  pw.write(tip);  pw.flush();  pw.close();  //以下方式不是最好的，但可以完成AJAX异步交互  **return** **null**;  }  } |

|  |
| --- |
| <struts>  <package name=*"checkcodePackage"* extemds=*"struts-default"* namespace=*"/"*>  <action name=*"checkRequest"*  class=*"sz.echuing.action.CheckcodeAction"*  method=*"check"*>  </action>  </package>  </struts> |

**九）XMLHttpRequest(即：AJAX)对象常用事件，方法和属性**

（1）事件：

ajax.onreadystatechange：表示AJAX异步对象不断监听服务端的响应，

是由服务器程序触发，不是程序员触发

（2）属性：

ajax.readyState==0：表示AJAX异步对象已创建好，但还没有调用open()方法

ajax.readyState==1：表示AJAX异步对象已调用open()方法，但还没有调用send()方法

ajax.readyState==2：表示AJAX异步对象已调用send()方法，但请求还没有到达服务器端

ajax.readyState==3：表示服务端已接收到AJAX异步对象的请求，正在处理响应中。。。

ajax.readyState==【4】：表示AJAX异步对象已经完完全全接收到了服务器的响应信

息，但接收到的数据不一定都正确

上述5种状态不是所有浏览器都相同，但状态4每种浏览器都相同

ajax.status==200：表示AJAX异步对象接收到响应码，如果是200的话，表示一切正常

ajax.responseText：表示从AJAX异步对象中获取HTML载体中的数据

ajax.responseXML：表示从AJAX异步对象中获取XML载体中的数据

（3）方法：

ajax.open(method,url,可选的boolean值)

AJAX异步对象准备发送异步请求

参数一：以什么方式发送，常用的用GET或POST，大小写不敏感

参数二：发送到什么地方去，常用Servlet或Struts2或SpringMVC来接收，

这里只限于JavaEE学科

参数三：默认值为true，表示AJAX对象以【异步】的方式提交到服务端

如果设置为false，表示AJAX对象以【同步】的方式提交到服务端

ajax.setRequestHeader("content-type","application/x-www-form-urlencoded")

表示将请求体中的内容，按照UTF-8的方式进行编码，只针对POST请求有效

ajax.send(content)

AJAX异步对象真正发送异步请求

参数一：表示HTTP【请求体】中的内容

如果是GET方式：content = null，如果强行传值到服务器，服务端收不到，返回NULL

如果是POST方式：content != null，例如：username=jack&password=123&role=admin

**十）数据载体**

（1）HTML

（A）优点：服务端响应的是普通html字符串，无需JS解析，配合innerHTML属性效率高

（B）缺点：如果需要更新WEB页面中的很多处地方，HTML不太方便，

同时innerHTML属性不是DOM的标准，不能操作XML

注意：innerHTML在xml中不能使用，用firstChild.nodeValue替代

（C）适合：小量数据载体，且只更新在web页面中的一个地方

（2）XML

（A）优点：是种通用的普通字符串格式，任何技术都能解析，标签名可以任意，

使用DOM标准规则能够解析XML的任何部分

（B）缺点：XML文件格式相当严格，出错后，responseXML属性返回NULL，

如果XML文件过长，导致解析效率低下

（C）适合：大量具有层次数据载体

（3）JSON （下次课讲）+ struts2框架

兼备HTML和XML的特点

作业：服务端采用Servlet完成

1）检查注册用户名是否在数据库中存在

2）省份-城市二级联动

3）验证码识别