目录

[(一) Web应用概述 2](#_Toc394559898)

[(1.1)web服务器 2](#_Toc394559899)

[(1.2)web部署描述文件 2](#_Toc394559900)

[(二) XHTML 3](#_Toc394559901)

[(2.1)表单 3](#_Toc394559902)

[(2.2)表格 4](#_Toc394559903)

[(2.3)超链接和图片 4](#_Toc394559904)

[(2.4)无序列表 4](#_Toc394559905)

[(三) CSS 5](#_Toc394559906)

[(3.1)CSS的盒子模型 5](#_Toc394559907)

[(3.2)样式表的分类 6](#_Toc394559908)

[(3.3)选择器的分类 7](#_Toc394559909)

[(3.4)div+css页面布局 8](#_Toc394559910)

[(四) JavaScript 11](#_Toc394559911)

[(4.1)JavaScript基础语法 11](#_Toc394559912)

[(4.2)DOM文档对象模型 14](#_Toc394559913)

[(4.3)案例练习 17](#_Toc394559914)

[(五) Servlet 18](#_Toc394559915)

[(5.1)Servlet概述 18](#_Toc394559916)

[(5.2)Servlet的生命周期 19](#_Toc394559917)

[(5.3)Servlet开发中的模板 21](#_Toc394559918)

[(5.4)Servlet启动加载 22](#_Toc394559919)

[(5.5)登陆模块 23](#_Toc394559920)

[(5.6)请求转发和重定向 24](#_Toc394559921)

[(5.7)路径问题 24](#_Toc394559922)

[(5.8)get请求和post请求 25](#_Toc394559923)

[(5.9)web组件之间跳转 25](#_Toc394559924)

[(5.10)乱码问题 25](#_Toc394559925)

[(5.11)过滤器 25](#_Toc394559926)

[(六) JSP 26](#_Toc394559927)

[(6.1)基础语法 27](#_Toc394559928)

[(6.2)常用指令和标签 27](#_Toc394559929)

[(6.3)JSP的9个内置对象 28](#_Toc394559930)

[(6.3.1)request对象 28](#_Toc394559931)

[(6.3.2)response对象 28](#_Toc394559932)

[(6.3.3)session对象 28](#_Toc394559933)

[(6.3.4)web组件之间的数据共享 29](#_Toc394559934)

[(6.4)JSTL和EL 29](#_Toc394559935)

[(七) AJAX 30](#_Toc394559936)

[(7.1)get请求实现 30](#_Toc394559937)

[(7.2)post请求实现 32](#_Toc394559938)

[(7.3)servlet响应实现 34](#_Toc394559939)

[(7.4)XMLHttpRequest对象 35](#_Toc394559940)

[(八) jQuery 35](#_Toc394559941)

[(8.1)选择器 36](#_Toc394559942)

[(8.2)DOM 37](#_Toc394559943)

[(8.3)AJAX 37](#_Toc394559944)

# Web应用概述

Web应用程序是一种可以通过Web访问的应用程序。Web应用程序的一个最大好处是用户很容易访问应用程序。用户只需要有[浏览器](http://baike.baidu.com/view/7718.htm)即可，不需要再安装其他软件。

## (1.1)web服务器

|  |
| --- |
| **(1)安装Tomcat6.0服务器**  端口8005、8080、8009  **(2)在myeclipse中配置Tomcat6.0**  设置tomcat可用、tomcat路径、配置JDK  **(3)在myeclipse中创建web应用**  Web root folder：WebRoot是web应用的发布路径、主要存放视图层的资源以及src的字节码文件、直接拷贝到tomcat6.0/webapps下。  Context root URL：是web应用的访问路径、必须以/开头、即在tomcat服务器中的名字、浏览器地址栏中访问的web应用的名字。  **(4)发布web应用**  发布的名字的敏感度、可以直接发布、也可以压缩成war包发布。  **(5)访问web应用**  <http://服务器的IP地址:端口/访问路径/文件路径/>资源全名  注意大小写敏感 |

## (1.2)web部署描述文件

web.xml是web部署描述文件，是Java平台web应用必须要有的配置文件，是整个web应用的核心配置文件。

欢迎列表可以配置多个，从前往后查找，哪个先找到就显示哪个页面。

|  |
| --- |
| <welcome-file-list>  <welcome-file>/page/x.html</welcome-file>  <welcome-file>/page/y.html</welcome-file>  </welcome-file-list> |

# XHTML

**HTML概念：**

超文本[标记语言](http://baike.baidu.com/view/329009.htm)，[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)下的一个应用。

“[超文本](http://baike.baidu.com/view/156868.htm)”就是指页面内可以包含图片、[链接](http://baike.baidu.com/view/147669.htm)，甚至音乐、[程序](http://baike.baidu.com/subview/17674/17674.htm)等非文字元素。

超文本标记语言的结构包括“头”部分（[外语](http://baike.baidu.com/view/127315.htm)：Head）、和“主体”部分（[外语](http://baike.baidu.com/view/127315.htm)：Body），其中“头”部提供关于网页的信息，“主体”部分提供网页的具体内容。

**XHTML概念：**

可扩展[超文本](http://baike.baidu.com/view/156868.htm)标记语言，是一种置标语言，表现方式与[超文本标记语言](http://baike.baidu.com/view/383720.htm)（[HTML](http://baike.baidu.com/view/692.htm)）类似，不过[语法](http://baike.baidu.com/view/135635.htm)上更加严格。符合XML规范。

## (2.1)表单

|  |
| --- |
| <!-- form标签提供用户的录入信息，表单分为表单项和表单按钮，一个页面可以多个表单。 -->  <!-- (1)action属性表示提交路径，即把表单提交给谁。 -->  <!-- (2)method属性表示请求类型。 -->  <!-- get请求是将数据放在地址栏中，然后发给服务器的。 -->  <!-- post请求时将数据放在http的请求头中，然后发给服务器的。 -->  <formaction=*"a.html"*method=*"get"*>  文本框<inputtype=*"text"*name=*"t01"*/><P/>  密码框<inputtype=*"password"*name=*"p01"*/><P/>    <!--单选按钮的name必须相同，value属性表示真正发送的值。 -->  单选按钮<inputtype=*"radio"*name=*"r01"*value=*"1"*checked=*"checked"*/>男  <inputtype=*"radio"*name=*"r01"*value=*"2"*/>女<P/>    <!--复选按钮的name强烈建议相同，value属性表示真正发送的值。 -->  复选按钮<inputtype=*"checkbox"*name=*"c01"*value=*"zuqiu"*/>足球  <inputtype=*"checkbox"*name=*"c01"*value=*"lanqiu"*/>篮球  <inputtype=*"checkbox"*name=*"c01"*value=*"paiqiu"*/>排球<P/>    <!-- name属性表示标签的名字，value属性表示真正发送的值。 -->  下来菜单  <selectname=*"s01"*>  <optionvalue=*"dalian"*>大连</option>  <optionvalue=*"beijing"*selected=*"selected"*>北京</option>  <optionvalue=*"shanghai"*>上海</option>  </select><P/>    <!-- value属性按钮显示的内容。 -->  提交按钮<inputtype=*"submit"*value=*"提交按钮"*/><P/>  重置按钮<inputtype=*"reset"*value=*"重置按钮"*/><P/>    <!-- 文本域 -->  <textareaname=*"t01"*rows=*"5"*cols=*"20"*>文本域</textarea>  <!-- 隐藏域 -->  <inputname=*"h01"*type=*"hidden"*value=*"123456"*/>  <!-- 普通按钮 -->  <inputname=*"b01"*type=*"button"*value=*"按钮"*/>  </form> |

## (2.2)表格

|  |
| --- |
| <!--在html中表格是由行构成的、行是由单元格构成。 -->  <!-- table表格、tr行、th标题单元格、td数据单元格-->  <!-- width宽、height高、border边框、colspan合并单元格 -->  <tablewidth=*"500"*height=*"200"*border=*"1"*>  <caption>表格的标题</caption>  <tr><th>姓名</th><th>年龄</th><th>部门</th></tr>  <tr><td>张三</td><td>20</td><td>市场部</td></tr>  <tr><td>李四</td><tdcolspan=*"2"*>25研发部</td></tr>  </table> |

## (2.3)超链接和图片

|  |
| --- |
| <!-- a表示超链接、href表示跳转路径 -->  <!-- 注意路径问题的讨论： -->  <!-- web应用中的路径分为：  (1)相对路径：不以/开头的路径写法属于相对路径，相对于地址栏的路径。  (2)绝对路径：以/开头的路径写法属于绝对路径，/访问路径/文件路径/文件全称。  -->  <ahref=*"y.html"*>跳转到&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Y页面</a>  <ahref=*"/web03/page/y.html"*>跳转到Y页面</a>  <!-- img图片标签、alt提示信息、src图片路径 -->  <imgalt=*"提示信息"*src=*"/web03/img/tu.jpg"*> |

## (2.4)无序列表

|  |
| --- |
| <!-- 无序列表 -->  <ul>  <li>导航菜单一</li>  <li>导航菜单二</li>  <li>导航菜单三</li>  <li>导航菜单四</li>  </ul>  <!-- 有序列表 -->  <ol>  <li>导航菜单一</li>  <li>导航菜单二</li>  <li>导航菜单三</li>  <li>导航菜单四</li>  </ol> |

# CSS

CSS层叠样式表，用来编写实现页面的样式。

[CSS](http://baike.baidu.com/subview/15916/5236733.htm)目前最新版本为[CSS3](http://baike.baidu.com/view/1713027.htm)，是能够真正做到[网页](http://baike.baidu.com/view/828.htm)表现与内容分离的一种样式设计语言。

## (3.1)CSS的盒子模型

|  |
| --- |
| <bodystyle="border:*2px solid*;">  <!--  CSS的盒子模型由6个属性构成：所有的html标签的样式都具备这样六个属性  内容：content  内边距：padding  边框：border  外边距：margin  背景颜色：background-color  背景图片：background-image  注意：如果背景颜色和背景图片同时存在，则显示背景图片。  -->    <!-- style表示样式的属性， -->  <divstyle="font-size:*50px*;color:*red*;">CSS层叠样式表</div>    <!-- 理解CSS的盒子模型 -->  <pstyle="width:*200px*;height:*200px*;border:*2px solid*;background-image:*url(../img/tu.jpg)*">    <pstyle="width:*100px*;height:*100px*;border:*2px solid*;padding:*10px*;margin-left:*20px*;background-color:*red*;">    内容content    </p>    </p>    </body> |

## (3.2)样式表的分类

|  |
| --- |
| <html>  <head>  <title>CSS样式表的分类：</title>  <!-- (2)嵌入式样式表  <style type="text/css">  div {  font-size:50px;  color:red;  }  </style>  -->  <!-- (3)应用外部样式表。 -->  <linkrel=*"stylesheet"*type=*"text/css"*href=*"/web04/css/main.css"*/>  </head>  <body>  <!--  CSS根据样式定义的位置将样式表分为四类：  (1)内联式样式表：将样式定义在标签里面。缺点：一个页面相同样式不合适!  (2)嵌入式样式表：将样式定义在页面的<style></style>中。缺点：多个页面相同样式不适合!  (3)外部样式表：将样式文件定义在外部，然后页面引用。优点：真正实现内容和样式的分离。  (4)输入样式表：css文件之间可以相互引入(页面也可以引入)。思考！！！  -->    <!-- (1)内联式样式表 -->  <div>CSS层叠样式表</div>  <div>CSS层叠样式表</div>  <div>CSS层叠样式表</div>    <p>CSS层叠样式表</p>  <p>CSS层叠样式表</p>  <p>CSS层叠样式表</p>  </body>  </html> |

## (3.3)选择器的分类

|  |
| --- |
| <html>  <head>  <title>CSS选择器的分类</title>  <styletype=*"text/css"*>  **ollia***:link* {  font-size:*20px*;  color:*green*;  }    **ollia***:visited* {  font-size:*30px*;  color:*red*;  }    **ollia***:hover* {  font-size:*50px*;  color:*yellow*;  }    </style>  </head>  <body>  <!--  CSS选择器的理解：样式表已经定义好，可以选择那些html标签进行渲染。分为六种：  (1)html选择器：使用html标签当做样式的名字，选择所有html标签。  (2)class选择器：使用"[标签].样式名"当做样式的名字，使用class属性引用，有标签会受限。  (3)id选择器：使用"#样式名"当做样式的名字，使用id属性引用。注意：id必须不能重复！  (4)分组选择器：相同的样式应用于多个选择器。用逗号分隔。  (5)包含选择器：选择所有被标签一包含的标签二进行渲染。用空格分隔。  (6)伪选择器：仅用于超链接的样式。  a:link链接样式、a:hover鼠标落在上方的样式、a:visited访问后的样式  -->  <ol>  <li><ahref=*"#"*>菜单导航一</a></li>  <li><ahref=*"#"*>菜单导航二</a></li>  <li><ahref=*"#"*>菜单导航三</a></li>  <li><ahref=*"#"*>菜单导航四</a></li>  </ol>  </body>  </html> |

## (3.4)div+css页面布局

DIV+CSS是WEB设计标准，它是一种网页的布局方法。与传统中通过表格（table）布局定位的方式不同，它可以实现网页页面内容与表现相分离。提起[DIV+CSS](http://baike.baidu.com/view/399288.htm)组合，还要从XHTML说起。[XHTML](http://baike.baidu.com/view/15906.htm)是一种在[HTML](http://baike.baidu.com/view/692.htm)（[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)的子集）基础上优化和改进的新语言，目的是基于XML应用与强大的数据转换能力，适应未来网络应用更多的需求。

“DIV+CSS”其实是错误的叫法，而标准的叫法应是XHTML+CSS。

|  |
| --- |
| <html>  <head>  <title>div+css网页局部</title>  <styletype=*"text/css"*>  **body** {  margin:*0px*;  background-color:*#ffffcc*;  }    *#container* {  width:*800px*;  height:*800px*;  background-color:*yellow*;  margin:*auto*;  }  *#header* {  width:*800px*;  height:*100px*;  background-color:*red*;  margin:*auto*;  }    *#main* {  width:*800px*;  height:*600px*;  background-color:*green*;  margin:*auto*;  }    *#footer* {  width:*800px*;  height:*100px*;  background-color:*blue*;  margin:*auto*;  }    *#logo* {  width:*250px*;  height:*90px*;  background-color:*grey*;  float:*left*;  }    *#banner* {  width:*540px*;  height:*90px*;  background-color:*grey*;  float:*right*;  }    *#menu* {  width:*800px*;  height:*40px*;  background-color:*#ffccff*;  }    *#left* {  width:*180px*;  height:*600px*;  background-color:*#ffff00*;  float:*left*;  }    *#right* {  width:*600px*;  height:*600px*;  background-color:*#ffff00*;  float:*right*;  }  </style>  </head>  <body>  <inputtype=*"text"*id=*"abc"*/>  <inputtype=*"button"*value=*"按钮"*onclick="alert(document.getElementById('abc').value);"/>  <!-- float属性：可以让块级元素浮动到左边或右边、后边的块级元素则会浮动上来。 -->    <!-- 整个容器的区域 -->  <divid=*"container"*>  <!-- 头部区域 -->  <divid=*"header"*>  <divid=*"logo"*></div>  <divid=*"banner"*></div>  </div>    <!-- 菜单区域 -->  <divid=*"menu"*></div>    <!-- 主体区域 -->  <divid=*"main"*>  <divid=*"left"*></div>  <divid=*"right"*></div>  </div>    <!-- 底部区域 -->  <divid=*"footer"*>底部</div>    </div>  </body>  </html> |

# JavaScript

Javascript[1] 是一种由[Netscape](http://baike.baidu.com/view/153922.htm)的[LiveScript](http://baike.baidu.com/view/2373233.htm)发展而来的原型化继承的[基于对象](http://baike.baidu.com/view/3481746.htm)的动态类型的区分大小写的客户端[脚本语言](http://baike.baidu.com/view/76320.htm)，主要目的是为了解决[服务器](http://baike.baidu.com/view/899.htm)端语言，比如[Perl](http://baike.baidu.com/view/46614.htm)，遗留的速度问题，为客户提供更流畅的浏览效果。当时服务端需要对数据进行验证，由于[网络速度](http://baike.baidu.com/view/1065551.htm)相当缓慢，只有28.8kbps，验证步骤浪费的时间太多。于是[Netscape](http://baike.baidu.com/view/153922.htm)的浏览器[Navigator](http://baike.baidu.com/view/887414.htm)加入了Javascript，提供了[数据验证](http://baike.baidu.com/view/3248821.htm)的基本功能，js的作用也主要在实现网页的一些特效，[css](http://baike.baidu.com/subview/15916/5236737.htm)难以实现的特效，通常和[Jquery](http://baike.baidu.com/view/1020297.htm)，[Ajax](http://baike.baidu.com/subview/1641/5762264.htm),联合使用。

HTML超文本标记语言，负责编写页面的内容。

CSS层叠样式表，负责编写页面的样式。

JavaScript脚本语言，负责编写和用户进行动态交互。

C/S架构：Client客户端/Server服务器端架构。桌面应用程序。

B/S架构：Brower浏览器/Server服务器架构。Web应用程序。

HTML、CSS、JavaScript都是由web服务器(比如tomcat)发送到客户端的浏览器中执行。

RIA：富客户端应用。

JavaScript和Java：没有任何关系。（雷锋和雷峰塔的关系）

## (4.1)JavaScript基础语法

|  |
| --- |
| <scripttype=*"text/javascript"*>    // (一)JavaScript的数据类型  // JavaScript语言大小写敏感。  // JavaScript是弱数据类型、但是可以通过typeof函数判断其类型。  // 浏览器执行JS脚本是从上至下逐行执行。  // (1)字符串类型、声明变量统一使用var关键字。  // 如果要输出单引则外面双引，如果要输出双引则外面单引。  **var**v1 = "'字符串类型'";  document.write(v1 + "<p/>");  document.write(**typeof**(v1) + "<p/>");  // alert(v1);  // (2)数字类型  **var**v2 = 100;  **var**v3 = 50.55;  **var**v4 = -200;  **var**v5 = 3e5;  document.write(v2 + "<p/>");  document.write(v3 + "<p/>");  document.write(v4 + "<p/>");  document.write(v5 + "<p/>");  document.write(**typeof**(v2) + "<p/>");  // 判断是否是数字类型的函数isNaN  // 如果不是数字类型返回true、如果是数字类型返回false  alert(isNaN(v1));  alert(isNaN(v2));  // (3)布尔类型  **var**v6 = **true**;  v6 = **false**;  document.write(v6 + "<p/>");  document.write(**typeof**(v6) + "<p/>");  // (4)数组类型  **var**list = **new** Array("大连", "北京", "上海");  document.write("数组长度：" + list.length + "<p/>");  document.write("数组元素：" + list[2] + "<p/>");  document.write(**typeof**(list) + "<p/>");    // (二)JavaScript运算符  document.write(2 < 3);  document.write(2 <= 3);  document.write(2 > 3);  document.write(2 >= 3);  document.write(2 == 3);  document.write(2 != 3);  document.write("abc"<"acc");  document.write("<p/>");  document.write(**true** &&**false**);  document.write(**true** || **false**);  document.write(!**false**);    // (三)JavaScript的流程控制语句  // if分支语句  **var**a = 2;  **var**b = 3;  **var**c = 5;  **if** ((a>b) && (a>c)) {  alert(a);  } **else if** ((b>a) && (b>c)) {  alert(b);  } **else** {  alert(c);  }    // while循环、for循环  **var**s = 0;  **for** (**var**i=1; i<= 10; i++) {  s = s + i;  }  alert(s);      </script> |

|  |
| --- |
| <html>  <head>  <title>JavaScript函数、事件</title>  <scripttype=*"text/javascript"*>  // 定义函数：function关键字函数名 (参数列表)  // 调用函数一定是通过某一个事件源的事件触发完成的。    // 没有返回值的函数  **function** m1(a, b) {  alert(a + b);  }    // 有返回值的函数，用return关键字返回即可，return返回到函数的调用处。  **function** m2() {  **return false**;  }    </script>  </head>  <body>  <!--onclick表示单击事件、单击时候触发。 -->  <inputtype=*"button"*value=*"点击按钮触发事件"*onclick="m1(2,3)"/>  <!--onsubmit表示提交事件、当提交表单时候触发。  return true则表单提交、return false则表单不提交。-->  <formaction=*"/web05/page/js01.html"*method=*"get"*onsubmit="return m2();">  <inputtype=*"submit"*value=*"提交按钮"*/>  </form>  </body>  </html> |

## (4.2)DOM文档对象模型

元素节点、属性节点、文本节点。

浏览器将HTML的所有内容都封装到一个对象中、这个对象就是document对象、简称到DOM对象。所以html中的所有内容DOM中都有。

|  |
| --- |
| <html>  <head>  <title>DOM的常用方法</title>  <!--  获取元素节点的方法：  **(1)getElementById函数：根据id获取一个元素节点的对象。**  **(2)getElementsByTagName函数：根据标签名获取元素节点对象的数组对象，然后通过下标操作。**    **获取属性节点的方法：**  **(3)getAttribute函数：获取属性节点值。**  **(4)setAttribute函数：设置属性节点值。**    **获取文本节点的方法：**  **(5)childNodes[0].nodeValue：获取文本节点的值**  -->  <scripttype=*"text/javascript"*>  **function** m1() {  **var**v1 = document.getElementById("username");  alert(v1.value);  **var**v2 = document.getElementById("password");  alert(v2.value);  **var**v3 = document.getElementById("city");  alert(v3.value);  }  **function** m2() {  // 获取标签名叫input的元素节点的数组  **var**arr = document.getElementsByTagName("input");  alert("长度：" + arr.length);  alert("用户：" + arr[0].value);  alert("密码：" + arr[1].value);  alert((document.getElementsByTagName("select"))[0].value);  }  **function** m3(dir) {  **var**v1 = document.getElementById("tu");  // 获取属性节点的值、参数是属性节点的名字  **var**v2 = v1.getAttribute("src");  // 设置属性节点的值  v1.setAttribute("src", dir);  }  **function** m4() {  **var**v = document.getElementById("a");  // 获取文本节点的值  alert(v.childNodes[0].nodeValue);  }  </script>  </head>  <body>  用户：  <inputid=*"username"*type=*"text"*/>  <p>  密码：<inputid=*"password"*type=*"password"*/>  <p>  城市：<selectid=*"city"*>  <optionvalue=*"1"*>大连</option>  <optionvalue=*"2"*>上海</option>  </select>  <p>  图片：<imgid=*"tu"*alt=*"提示信息"*src=*"/web05/img/tu.jpg"*/>    <p>    <inputtype=*"button"*value=*"按钮"*onclick="m4()"/>  <p>  <aid=*"a"*href=*"#"*>网站的地址</a>    </body>  </html> |

|  |
| --- |
| <html>  <head>  <title>DOM的常用方法</title>  <!-- DOM的常用方法：  **(6)createElement函数：创建元素节点的对象。**  **(7)createTextNode函数：创建文本节点的对象。**  **(8)appendChild函数：添加子节点。**  **(9)innerHTML属性：节点中的html内容，可以获取也可以设置。**  (10)事件：IE浏览器中的事件都是冒泡型的事件。  -->  <scripttype=*"text/javascript"*>  **function** m1() {  // 创建元素节点对象  **var**v1 = document.createElement("div");  // 创建文本节点对象  **var**v2 = document.createTextNode("文本节点的内容!");  // 将v2放入v1中  v1.appendChild(v2);  document.body.appendChild(v1);  }  **function** m2() {  // 通过dom动态创建一个下拉菜单  **var**v1 = document.createElement("select");    **var**v2 = document.createElement("option");  **var**v3 = document.createTextNode("北京");  v2.appendChild(v3);  v1.appendChild(v2);  document.body.appendChild(v1);  }  **function** m3() {  **var**v = document.getElementById("d");  alert(v.innerHTML);  v.innerHTML = "错误信息的提示!";  }  **function** m4(info) {  alert(info);  }  </script>  </head>  <bodyonclick="m4('body');">  <divonclick="m4('div');">    <spanonclick="m4('span');">文本</span>    </div>  </body>  </html> |

## (4.3)案例练习

|  |
| --- |
| <scripttype=*"text/javascript"*>  **function** checkAll() {    **var**v1 = document.getElementById("c1");  **var**list = document.getElementsByTagName("input");  **for** (**var**i=0; i<list.length; i++) {  list[i].checked = v1.checked;  }    }  </script>  <scripttype=*"text/javascript"*>    **function** check() {    **var**v1 = document.getElementById("username");  **var**v11 = document.getElementById("usernameError");  **if** (v1.value.length == 0) {  v11.innerHTML = "<font color='red'>用户不能为空!</font>";  v1.focus();  **return false**;  } **else** {  v11.innerHTML = "";  }    **var**v2 = document.getElementById("age");  **if** (v2.value.length == 0) {  alert("年龄不能为空!");  v2.focus();  **return false**;  }      **return true**;  }    </script> |

# Servlet

servlet是在[服务器](http://baike.baidu.com/view/899.htm)上运行的小[程序](http://baike.baidu.com/view/17674.htm)。

[服务器](http://baike.baidu.com/view/899.htm)上需要一些[程序](http://baike.baidu.com/view/17674.htm)，常常是根据用户输入访问[数据库](http://baike.baidu.com/view/1088.htm)的[程序](http://baike.baidu.com/view/17674.htm)。这些通常是使用[公共网关接口](http://baike.baidu.com/view/649193.htm)（CGI(Common Gateway Interface)）应用[程序](http://baike.baidu.com/view/17674.htm)完成的。然而，在[服务器](http://baike.baidu.com/view/899.htm)上运行Java，这种[程序](http://baike.baidu.com/view/17674.htm)可使用Java编程语言实现。在通信量大的服务器上，[Java](http://baike.baidu.com/view/29.htm)servlet的优点在于它们的执行速度更快于CGI[程序](http://baike.baidu.com/view/17674.htm)。各个用户请求被激活成单个[程序](http://baike.baidu.com/view/17674.htm)中的一个线程，而无需创建单独的进程，这意味着[服务器](http://baike.baidu.com/view/899.htm)端处理请求的[系统开销](http://baike.baidu.com/view/860875.htm)将明显降低。

## (5.1)Servlet概述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **第五层** | **第四层** | **第三层** | **第二层** | **第一层** |
| **View** | **Controller** | **Service** | **DAO** | **EIS** |
| **视图层** | **控制层** | **业务层** | **数据层** | **企业信息系统层** |
| **JSP(HTML+Java)**  **(HTML+CSS+JavaScript)** | **Servet** | **Spring** | **JDBC/iBatis** | **Oracle/Mysql** |

## (5.2)Servlet的生命周期

|  |
| --- |
| /\*\*  \*  \* Servlet的概念：Servlet在web应用中是控制层的技术。Servlet的作用：接受页面请求、处理业务功能、跳转下一个页面。  \* Servlet的实现：直接继承javax.servlet.http.HttpServlet就可以。  \*  \* JavaEE平台的核心的理念：容器+组件+配置文件。容器提供给组件运行环境，容器管理组件的声明周期，容器通过配置文件管理组件。  \*  \* Servlet组件(web组件) + Tomcat容器(web容器) + web.xml(web部署描述文件) JSP组件(web组件)  \*  \*  \* weblogic、websphere、jboss：可以叫做web容器、web服务器、因为它们都可以运行servlet组件和jsp组件。  \* 但是，通常管它们叫做应用服务器，原因是，它们可以运行EJB组件，而tomcat不可以。  \*  \* Servlet运行机制：Tomcat只会创建servlet的一个对象，如果有10个用户发起请求，tomcat会开启10个线程处理。  \*  \* servlet组件的生命周期：  \*  \* (1)初始化阶段init()、init(ServletConfigconfig)  \* 在请求第一次到达的时候web服务器会调用它、而且只调用一次。  \* 初始化方法有两个，一个无参，一个有参，如果只写一个就调用这个初始化方法。  \* 如果两个同时写，则调用有参的初始化方法。  \*  \* (2)业务处理阶段 service方法、doGet方法、doPost方法请求多少次就调用多少次。  \* 当请求访问tomcat的时候，tomcat会先调用HttpServlet中的service方法，  \* servcie方法判断请求类型，如果是get请求就调用doGet方法，如果是post请求就调用doPost方法。  \*  \* (3)销毁阶段 destroy方法在服务器停止时候web服务器会调用它、只调用一次。  \*  \*  \*/  PS:service方法重写及不重写http://blog.csdn.net/w363403269/article/details/37810317  **publicclass**Action01**extends**HttpServlet {  /\*\*  \* 无参的初始化方法  \*/  **publicvoid**init() **throws**ServletException {  System.*out*.println("生命周期的第一阶段：初始化阶段init无参方法。");  }  /\*\*  \* 有参的初始化方法如果需要读取web.xml中的配置信息的时候，采用此种方法初始化。否则通常使用无参初始化。  \*/  **publicvoid**init(ServletConfigconfig) **throws**ServletException {  System.*out*.println("生命周期的第一阶段：初始化阶段init有参方法。");  }  /\*\*  \* service方法  \*/  **protectedvoid** service(HttpServletRequest arg0, HttpServletResponse arg1)  **throws**ServletException, IOException {  System.*out*.println("生命周期的第二阶段：业务处理阶段service方法。");  }  /\*\*  \* doGet方法用来处理get请求。  \*/  **protectedvoid**doGet(HttpServletRequestreq, HttpServletResponseresp)  **throws**ServletException, IOException {  System.*out*.println("doGet方法");  }  /\*\*  \* doPost方法用来处理post请求。  \*/  **protectedvoid**doPost(HttpServletRequestreq, HttpServletResponseresp)  **throws**ServletException, IOException {  System.*out*.println("doPost方法");  }  /\*\*  \* 销毁方法  \*/  **publicvoid** destroy() {  System.*out*.println("生命周期的第三阶段：销毁阶段destroy方法。");  }  }  <!-- 配置Servlet组件 -->  <servlet>  <servlet-name>action01</servlet-name>  <servlet-class>com.zhaoyang.action.Action01</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>action01</servlet-name>  <url-pattern>/abc</url-pattern>  </servlet-mapping> |

## (5.3)Servlet开发中的模板

|  |
| --- |
| /\*\*  \*  \* 开发中Servlet的标准写法  \*  \*/  **publicclass**Action02**extends**HttpServlet {  /\*\*  \* get请求时默认的请求类型。用来处理get请求。例如(表单的get请求、超链接的get请求、地址栏的get请求)  \*/  **protectedvoid**doGet(HttpServletRequest request, HttpServletResponseresponse)  **throws**ServletException, IOException {  // 在doGet方法中调用doPost方法  **this**.doPost(request, response);  }  /\*\*  \* 用来处理post请求。例如(表单的post请求。)  \*/  **protectedvoid**doPost(HttpServletRequest request, HttpServletResponse response)  **throws**ServletException, IOException {  // 调用Service层的方法、实现业务功能。  }  } |

## (5.4)Servlet启动加载

|  |
| --- |
| /\*\*  \*  \* 用来初始化的控制器  \*  \*/  **publicclass**Action**extends**HttpServlet {  /\*\*  \*  \* 有参的初始化方法：读取web.xml文件中参数  \*  \*/  **publicvoid**init(ServletConfigconfig) **throws**ServletException {  System.*out*.println("#####有参的初始化方法#####");    String username = config.getInitParameter("username");  String password = config.getInitParameter("password");  System.*out*.println(username);  System.*out*.println(password);    }  }  <!-- 用来初始化的Servlet的配置 -->  <!--如果配置了load-on-startup标签，tomcat会在启动时初始化Servlet。 -->  <servlet>  <servlet-name>action</servlet-name>  <servlet-class>com.zhaoyang.action.Action</servlet-class>  <init-param>  <param-name>username</param-name>  <param-value>张三</param-value>  </init-param>  <init-param>  <param-name>password</param-name>  <param-value>123456</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet> |

## (5.5)登陆模块

|  |
| --- |
| **publicclass**LoginAction**extends**HttpServlet {  **protectedvoid**doGet(HttpServletRequest request,  HttpServletResponse response) **throws**ServletException, IOException {  **this**.doPost(request, response);  }  /\*\*  \* HttpServletRequest表示请求对象：HttpServletResponse表示响应对象：  \*/  **protectedvoid**doPost(HttpServletRequest request,  HttpServletResponse response) **throws**ServletException, IOException {  // tomcat处理请求的中文乱码问题：  // 原因：tomcat会对发送过来的请求进行了编码(iso8859-1)  // 数据->JSP(UTF-8)->tomcat(iso8859-1)->Servlet  // 解决：设置编码方式为UTF-8。前提条件：请求必须是post请求。  request.setCharacterEncoding("UTF-8");  PS:spring和tomcat解决乱码：http://blog.csdn.net/rchm8519/article/details/41930393  Tomcat自身对于get,post的编码方式  http://www.iteye.com/problems/7295  Gbk与utf-8的区别：http://jingyan.baidu.com/article/7f41ececf3ca46593c095c56.html  Servlet与Tomcat的关系  http://blog.csdn.net/wabiaozia/article/details/50313017  // **TODO**核心业务  // 第一步：获取表单中的数据，参数是属性标签中name的值。  // byte[] arr = request.getParameter("username").getBytes("iso8859-1");  // String username = new String(arr, "UTF-8");  String username = request.getParameter("username");  String password = request.getParameter("password");  // 第二步：处理业务(设置跳转路径)  String path = "";  **if** ("张三".equals(username) &&"123".equals(password)) {  path = "/page/main.jsp";  } **else** {  path = "/page/login.jsp";  }  // 第三步：跳转  request.getRequestDispatcher(path).forward(request, response);  }  } |

## (5.6)请求转发和重定向

|  |
| --- |
| // web应用中的跳转方式有两种：  // (1)请求转发(服务器端跳转)：浏览器发起请求，服务器去找资源，  // 然后把资源的内容直接发给浏览器。所以此时地址栏不发生变化。  // 使用的场合：组件之间数据共享采用此种方式跳转。  request.getRequestDispatcher(path).forward(request, response);  // (2)请求重定向(客户端跳转)：浏览器发起请求、服务器将资源的URL地址发给浏览器，  // 然后浏览器自己再次发起请求访问资源。所以此时地址栏发生变化。  // 使用的场合：通常添加和删除操作的时候为了避免重复提交采用此种方式。外网地址。  PS:重定向可以跨域，所以可以访问外网，请求转发只能在web服务器内部作用  //response.sendRedirect(path); |

## (5.7)路径问题

|  |
| --- |
| 1. **相对路径：不以”/”开头，相对于浏览器地址栏的路径。**   **<a href=”page/success.jsp”></a>**   1. **绝对路径：以”/”开头，从web应用的访问路径开始写。**   **<a href=”/we01/page/success.jsp”></a>**   1. **设定基准路径、写相对路径。**   **<base href=”http://127.0.0.1:8080/web01” />**  **<ahref=”page/success.jsp”></a>**  **<form action=”loing”></form>**  **<imgsrc=”img/tu.jpg”/>**  **PS:新建jsp页面都会出现base标签而且必须是在head标签里（一直延伸到项目）,如果表单写相对路径（action=”映射地址”） (若使用绝对路径,表单action：/项目名/映射地址)**  **表单提交写相对路径则相对的是basePath(即传输协议+IP+端口+项目名+/)**  **绝对路径是相对path（即传输协议+IP+端口）** |

## (5.8)get请求和post请求

|  |
| --- |
| Get请求：参数值显示在地址栏。将数据放在地址栏中发送。4k-5k。  Post请求：参数值不显示。将数据放在http的请求头中发送。没有大小限制。  PS：本质区别链接：http://www.techweb.com.cn/network/system/2016-10-11/2407736.shtml |

## (5.9)web组件之间跳转

|  |
| --- |
| Web组件：JSP组件、Servlet组件。  (1)JSP->跳转->JSP：通常是超链接的静态跳转。  **(2)JSP->跳转->Servlet：常用。**  **(3)Servlet->跳转->JSP：常用。**  **(4)Servlet->跳转->Servlet：常用。**  PS:request.getRequestDispatcher("/你要跳转的servlet").forward(request,response);  **PS:jsp->jsp a标签跳转即可或**jsp:forward亦或window.location.href等 |

## (5.10)乱码问题

|  |
| --- |
| // tomcat处理请求的中文乱码问题：  // 原因：tomcat会对发送过来的请求进行了编码(iso8859-1)  // 数据->JSP(UTF-8)->tomcat(iso8859-1)->Servlet  // 解决：设置编码方式为UTF-8。前提条件：请求必须是post请求。  request.setCharacterEncoding("UTF-8");  // byte[] arr = request.getParameter("username").getBytes("iso8859-1");  // String username = new String(arr, "UTF-8"); |

## (5.11)过滤器

|  |
| --- |
| **publicclass**PathFilter**implements** Filter {  /\*\*  \* 服务器启动时初始化  \*/  **publicvoid**init(FilterConfig arg0) **throws**ServletException {  System.*out*.println("#####初始化阶段#####");  }  /\*\*  \* 核心业务方法  \*/  **publicvoid**doFilter(ServletRequest request, ServletResponse response,  FilterChain chain) **throws**IOException, ServletException {  // 第一件事：处理请求  HttpServletRequestreq = (HttpServletRequest) request;  HttpSession session = req.getSession();  Person person = (Person) session.getAttribute("user");  // 第二件事：将请求继续向下传递给控制器  **if** (person == **null**) {  // 用户不合法跳转到登陆页面  HttpServletResponse res = (HttpServletResponse) response;  res.sendRedirect("/member/login.jsp");  } **else** {  // 用户合法将请求继续向下传递  chain.doFilter(request, response);  }  }  /\*\*  \* 服务器停止时销毁  \*/  **publicvoid** destroy() {  System.*out*.println("#####销毁阶段#####");  }  }  <!-- (11)配置过滤器 -->  <filter>  <filter-name>path</filter-name>  <filter-class>com.zhaoyang.filter.PathFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>path</filter-name>  <url-pattern>/page/\*</url-pattern>  </filter-mapping> |

# JSP

java服务器页面[1] 是由[Sun Microsystems](http://baike.baidu.com/view/1061350.htm)公司倡导、许多公司参与一起建立的一种[动态网页](http://baike.baidu.com/view/348756.htm)技术标准。JSP技术有点类似ASP技术，它是在传统的[网页](http://baike.baidu.com/view/828.htm)HTML（[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)的子集）文件(\*.htm,\*.[html](http://baike.baidu.com/view/692.htm))中插入Java[程序段](http://baike.baidu.com/view/1005329.htm)(Scriptlet)和JSP标记(tag)，从而形成JSP文件，后缀名为(\*.jsp)。用JSP开发的Web应用是跨平台的，既能在Linux下运行，也能在其他[操作系统](http://baike.baidu.com/view/880.htm)上运行。

JSP技术、JSP组件、web组件、视图层的技术。

JSP本质：HTML + Java。

JSP运行原理：\*.jsp-> \*.java -> \*.class(所以JSP第一次运行慢)

## (6.1)基础语法

|  |
| --- |
| <!--显式注释：用来注释HTML，会被发送到浏览器端。 -->  <%  // 隐式注释  /\* Java代码注释 \*/  // Java代码注释  %>  <h1>JSP就是在HTML中写入Java程序，只要将Java程序写在"尖括号百分号"中即可。</h1>  <h1>JSP的基础语法：(1)声明(2)表达式(3)代码</h1>    <!-- (1)声明变量，全局变量。(此用法几乎不用！) -->  <%!**int** a = 2; %>    <!-- (2)表达式 -->  <%=a++%><p/>  <%=(2 + 5 \* 1)%><p/>  <%="表达式的值"%><p/>    <!-- (3)Java程序代码 -->  <%  String str = "你好";  out.println(str);  %> |

## (6.2)常用指令和标签

|  |
| --- |
| (1)page指令：用来指定编码方式、导包。  <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  (2)include指令：用来包含文件、静态包含、先包含后执行。  <%@ include file="head.jsp"%>  (3)include标签：用来包含文件、动态包含、先执行后包含。强烈建议采用此方式包含。  <jsp:include page="/page/head.jsp"></jsp:include>  (4)forward标签：  <jsp:forward page="/page/c.jsp"></jsp:forward> |

## (6.3)JSP的9个内置对象

**1.pageContext：PageContext。**

**2.page：Object。**

**3.request：HttpServletRequest。**

**4.response：HttpServletResponse。**

**5.session：HttpSession。**

**6.application：ServletContext。**

**7.out：JspWriter。**

**8.config：ServletConfig。**

**9.exception：Throwable。**

### (6.3.1)request对象

|  |
| --- |
| javax.servlet.http.HttpServletRequest  (1)setCharacterEncoding方法：设置编码方式  (2)getParameter方法：获取表单数据的单值  (3)getParameterValues方法：获取表单数据的多值  (4)getRequestDispatcher方法：请求转发 |

### (6.3.2)response对象

|  |
| --- |
| javax.servlet.http.HttpServletResponse  (1)sendRedirect方法：重定向  (2)setHeader方法：设置请求头信息  (3)addCookie方法：设置cookie信息 |

### (6.3.3)session对象

|  |
| --- |
| javax.servlet.http.HttpSession  (1)getId方法：获取会话ID的值  (2)isNew方法：判断是否是新的请求，是返回true，否则返回false  (3)invalidate方法：注销、会话销毁  (4)getCreationTime方法：获取创建会话的毫秒数  (5)getLastAccessedTime方法：获取最后一次访问的毫秒数 |

### (6.3.4)web组件之间的数据共享

|  |
| --- |
| pageContext：  page范围：将数据存储在当前组件的范围中。不常用。  request范围：将数据存储在一次请求的范围中，必须是请求转发。常用。  session范围：将数据存储在一次会话的范围中，转发和重定向都可以。常用。  application范围：将数据存储在一个应用的范围中。不常用。 |

## (6.4)JSTL和EL

|  |
| --- |
| JSTL是JSP的标准标签库。  EL是JSTL中的表达式语言，可以独立使用。  要使用JSTL标签库，首先要添加taglib指令  <%@ tagliburi="http://java.sun.com/jsp/jstl/core" prefix="c"%>  out标签用来输出  <c:out value="${sessionScope.key.name}"></c:out>  解释：  <c:out>是：JSTL标签  ${}是：EL表达式  pagetScope页面范围  requestScope请求范围  sessiontScope会话范围  applicationScope应用范围  ${key.age}说明：  如果EL表达式没有指定范围，则会从小到大去找，直到找到为止。  forEach标签用来遍历  <c:forEachvar="mem" items="${requestScope.list}">  ${mem.name}  ${mem.age}  </c:forEach> |

# AJAX

AJAX即“Asynchronous Javascript + XML[1] ”（异步JavaScript和XML[1] ），是指一种创建交互式[网页](http://baike.baidu.com/view/828.htm)应用的网页开发技术。

AJAX = 异步 [JavaScript](http://baike.baidu.com/view/16168.htm)和[XML](http://baike.baidu.com/view/63.htm)（[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)的子集）。

AJAX 是一种用于创建快速动态网页的技术。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页面。

**要求掌握查看XMLHttpRequst文档手册的能力。**

## (7.1)get请求实现

|  |
| --- |
| <html>  <head>  <base href="<%=basePath%>">  <title>AJAX案例练习</title>  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  <meta http-equiv="description" content="This is my page">  <!--  <link rel="stylesheet" type="text/css" href="styles.css">  -->    <script type="text/javascript">    varxmlhttp;      // AJAX核心的业务方法  // 只有当http的状态码为200，xmlhttp的状态为4，才处理业务功能  function method() {  if (xmlhttp.status == 200 &&xmlhttp.readyState == 4) {  varinfo = xmlhttp.responseText;  document.getElementById("info").innerHTML = decodeURI(info, "UTF-8");  }  }    function AJAXMethod() {    // 第一步：创建异步对象XMLHttpRequest对象  if (window.XMLHttpRequest) {  xmlhttp = new XMLHttpRequest();  } else {  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");  }    // 第二步：拼写URL路径以及参数列表(get请求的写法、时间戳欺骗浏览器每次发起新的请求、两次编码解决乱码问题)  varurl = "/ajax/ajax?username=" + document.getElementById("username").value + "&time=" + new Date().getTime();  url = encodeURI(encodeURI(url, "UTF-8"), "UTF-8");    // 第三步：设置回调函数(当服务器成功响应后调用的函数，是整个AJAX的核心业务实现)  xmlhttp.onreadystatechange = method;    // 第四步：设置请求的类型  xmlhttp.open("get",url,true);    // 第五步：发送  xmlhttp.send(null);      }    </script>    </head>  <body>    <h1>AJAX的学习案例</h1>    注册用户名：<input type="text" id="username" name="username" /><span id="info"></span><p/>    <input type="button" value="AJAX\_\_get请求按钮" onclick="AJAXMethod()" />    </body>  </html> |

## (7.2)post请求实现

|  |
| --- |
| <html>  <head>  <base href="<%=basePath%>">  <title>AJAX案例练习--Post请求</title>  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  <meta http-equiv="description" content="This is my page">  <!--  <link rel="stylesheet" type="text/css" href="styles.css">  -->  <script type="text/javascript">    varxmlhttp;      // AJAX核心的业务方法  // 只有当http的状态码为200，xmlhttp的状态为4，才处理业务功能  function method() {  if (xmlhttp.status == 200 &&xmlhttp.readyState == 4) {  varinfo = xmlhttp.responseXML;  alert(info);  document.getElementById("info").innerHTML = info.getElementsByTagName("span")[0].childNodes[0].nodeValue;  }  }    function AJAXMethod() {    // 第一步：创建异步对象XMLHttpRequest对象  if (window.XMLHttpRequest) {  xmlhttp = new XMLHttpRequest();  } else {  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");  }    // 第二步：拼写URL路径以及参数列表(post请求的写法、时间戳欺骗浏览器每次发起新的请求、两次编码解决乱码问题)  PS:时间戳作用  http://www.cnblogs.com/liqone/p/6163417.html  varurl = "/ajax/ajax?" + "time=" + new Date().getTime();  // post请求，参数的值是作为http请求头发送的，而不是通过地址栏发送  varparam = "username=" + document.getElementById("username").value;  param = encodeURI(encodeURI(param, "UTF-8"), "UTF-8");    // 第三步：设置回调函数(当服务器成功响应后调用的函数，是整个AJAX的核心业务实现)  xmlhttp.onreadystatechange = method;    // 第四步：设置请求的类型  xmlhttp.open("post",url,true);  // 设置请求头信息  xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  // 第五步：发送  xmlhttp.send(param);    }    </script>  </head>  <body>  <h1>  AJAX的学习案例--Post请求  </h1>  注册用户名：  <input type="text" id="username" name="username" />  <span id="info"></span>  <p />  <input type="button" value="AJAX\_\_post请求按钮" onclick="AJAXMethod()" />  </body>  </html> |

## (7.3)servlet响应实现

|  |
| --- |
| **publicclass**AJAXAction**extends**HttpServlet {  **protectedvoid**doGet(HttpServletRequest request,  HttpServletResponse response) **throws**ServletException, IOException {  // 从请求中获取数据  request.setCharacterEncoding("UTF-8");  String username = request.getParameter("username");  username = URLDecoder.*decode*(username, "UTF-8");  // 向客户端发送数据，设置回应的类型和编码方式  response.setContentType("text/xml;charset=utf-8");  PrintWriter out = response.getWriter();  **if** (username.equals("张三")) {  out.println("抱歉!该用户已被注册！");  } **else** {  out.println("恭喜你！可以注册！");  }  }  **protectedvoid**doPost(HttpServletRequest request,  HttpServletResponse response) **throws**ServletException, IOException {  // 从请求中获取数据  request.setCharacterEncoding("UTF-8");  String username = request.getParameter("username");  username = URLDecoder.*decode*(username, "UTF-8");  // 向客户端发送数据，设置回应的类型和编码方式  response.setContentType("text/xml;charset=utf-8");  PrintWriter out = response.getWriter();  **if** (username.equals("张三")) {  out.println("<div><span>抱歉!该用户已被注册！</span></div>");  } **else** {  out.println("<div><span>恭喜你！可以注册！</span></div>");  }  }  } |

## (7.4)XMLHttpRequest对象

|  |
| --- |
|  |

# jQuery

[Jquery](http://baike.baidu.com/view/1020297.htm)是继[prototype](http://baike.baidu.com/view/1217697.htm)之后又一个优秀的[Javascript](http://baike.baidu.com/view/16168.htm)[框架](http://baike.baidu.com/view/66971.htm)。它是轻量级的js库，它兼容CSS3，还兼容各种[浏览器](http://baike.baidu.com/view/7718.htm)（IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+），jQuery2.0及后续版本将不再支持IE6/7/8浏览器。jQuery使用户能更方便地处理[HTML](http://baike.baidu.com/view/692.htm)（[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)下的一个应用）、events、实现动画效果，并且方便地为网站提供AJAX交互。jQuery还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细，同时还有许多成熟的[插件](http://baike.baidu.com/view/18979.htm)可供选择。jQuery能够使用户的html页面保持代码和html内容分离，也就是说，不用再在html里面插入一堆js来调用命令了，只需定义id即可。

**要求掌握查看jQuery文档手册的能力。**

## (8.1)选择器

|  |
| --- |
| (1)html选择器  $("div")[0].childNodes[0].nodeValue  (2)class选择器  $(".b")[1].childNodes[0].nodeValue  (3)id选择器(思考！)  $("#c")[0].childNodes[0].nodeValue  (4)组合选择器  $("p,#a,.b").css("background-color","#00ff00");  (5)包含选择器  $("div span").css("background-color", "#0000ff");  (6)选择第一个元素  $("div:first").css("background-color", "#0000ff");  (7)选择最后一个元素  $("div:last").css("background-color", "#ff0000");  (8)偶数，下标从零开始  $("div:even").css("background-color", "#ff0000");  (9)奇数，下标从零开始  $("div:odd").css("background-color", "#00ff00");  (10)选择大于2的元素  $("div:gt(1)").css("background-color", "#00ff00");  (11)选择属性，匹配包含给定属性的元素  $("div[id]").css("background-color", "#00ff00");  $("div[id=a]").css("background-color", "#00ff00");  $("div[id=b][class=b]").css("background-color", "#00ff00");  (12)选择属性，以某值开头  $("div[id^=abc]").css("background-color", "#00ff00"); |

## (8.2)DOM

|  |
| --- |
| (1)向每个匹配的元素内部追加内容  $("div").append("<span>123</span>");  (2)向每个匹配的元素内部前置内容  $("div").prepend("<span>123</span>");  (3)显示文本内容  alert($("div").text());  (4)修改文本内容  $("div").text("<a href='#'>百度</a>");  (5)显示html内容  alert($("div").html());  (6)修改html内容  $("div").html("<a href='#'>百度</a>");  (7)添加属性节点  $("a").attr("href","http://www.baidu.com");  (8)修改属性节点  $("a").attr({href: "http://www.google.com"});  (9)删除属性节点  $("a").removeAttr("href"); |

## (8.3)AJAX

|  |
| --- |
| $.ajax({  type:"POST",  url:"/web13/ajax",  data:"username=" + encodeURI(encodeURI($("#username").val(), "UTF-8"), "UTF-8"),  dataType: "text",  success:**function**(msg) { $("#info").html(msg); }  }); |