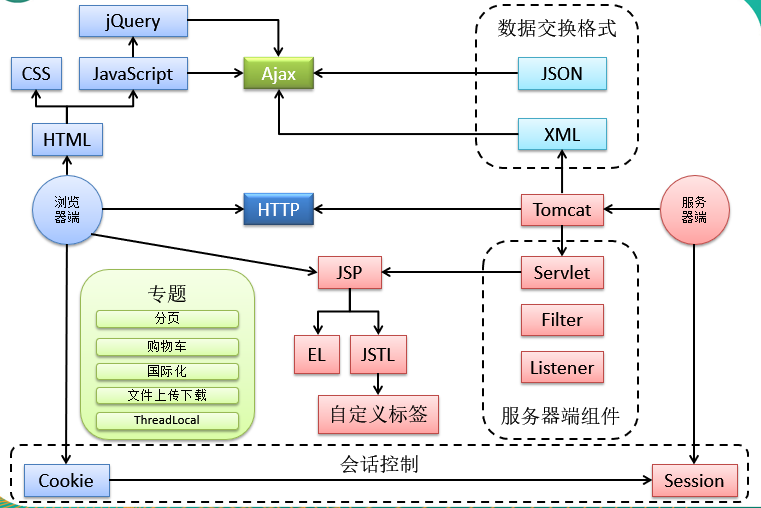
**尚硅谷大数据技术之JavaWeb**

# 第1章 简介

该阶段的学习目标是了解javaWEB的整个技术体系，熟悉JavaWEB前后台交互的实现方式及相关原理。明确未来大数据所分析数据的来源以及分析完成后的数据如何实现页面可视化.

# 第2章 JavaWeb的技术体系



# 第3章 前端三剑客

## 3.1 HTML

### 3.1.1 HTML是什么?

HTML指的超文本标记语言(Hyper Text Markup Language)，是一种用来描述网页的语言。超文本指的是除了可以包含文字之外，还可以包含图片、链接、音乐、视频、程序等内容。

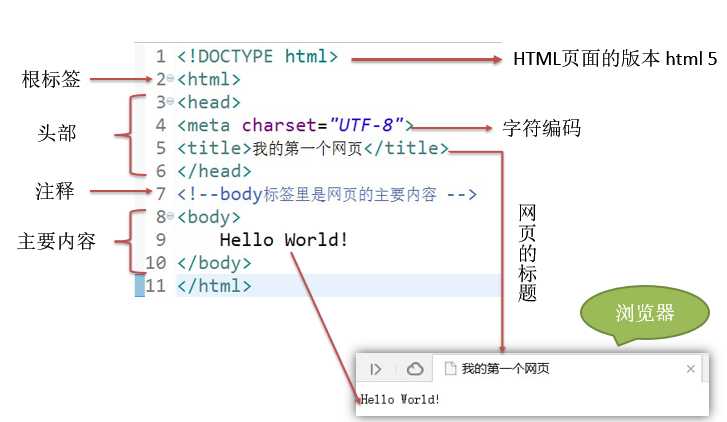
### 3.1.2 HTML网页的组成:



### 3.1.3 常用的HTML标签

1. html 根标记
2. head 头标记
3. body 体标记
4. a 超链接
5. form 表单
6. table 表格
7. h1~h6 标题
8. br 换行
9. img 图片

### 3.1.4 一个基本结构的HTML页面



### 3.1.5 登录页面的示例



## 3.2 CSS

### 3.2.1 CSS是什么

CSS 指层叠样式表 (Cascading Style Sheets)。主要用来设置网页中元素的样式。如边框，颜色，位置等…

CSS即可以写在HTML中，也可以写在元素的style属性里面，还可以写在.css外部文件里然后引入到页面

### 3.2.2 编写位置

1. 写在元素的style属性里面

|  |
| --- |
| <p style=”color:red;font-size:16px;”> Hello CSS </p> |

1. 写在HTML head标签中的style标签中

|  |
| --- |
| <style type="text/css">  p {  color: blue;  background-color: yellow;  }  </style> |

1. 写在外部的CSS文件中

|  |
| --- |
| <link rel="stylesheet" type="text/css" href="style.css" /> |

### 3.2.3 选择器

1. 标签(元素)选择器

按照标签名选中相应的元素

|  |
| --- |
| p {  color:red;  } |

1. Id选择器

按照元素的id选中相应的元素，使用#id值

|  |
| --- |
| #abc {  color:red;  }  <p id=”abc”>大家好</p> |

1. class类选择器

按照元素的类名选中相应的元素，使用.class值

|  |
| --- |
| .foot {  color:red;  }  <p class=”foot”>你好</p>  <b class=”foot”>你也好</b> |

### 3.2.4 常用样式

|  |  |  |
| --- | --- | --- |
| color | 设置颜色 | 颜色可以写颜色名如：black, blue, red, green, white, yellow等  颜色也可以写rgb值和十六进制表示值：如rgb(255,0,0)，#00F6DE，如果写十六进制值必须加# |
| background-color | 设置背景色 | 同上 |
| width/height | 设置宽度/高度 | 可以写像素值：19px；  也可以写百分比值：20%； |
| font-size | 设置字体大小 | 写像素值：19px； |
| text-align | 设置文本位置 | 左中右 left|cenger|right |
| border | 设置边框 | 例如:  黑色1像素实线边框  border: 1px solid black; |
| margin-left/margin-right/margin-top/margin-bottom | 设置左外边距/右外边距/上外边距/下外边距 | 例如: div居中  margin-left: *auto*;  margin-right: *auto*; |

## 3.3 JavaScript

### 3.3.1 什么是JavaScript

JavaScript是一门客户端脚本语言，主要运行在浏览器中，浏览器中负责运行JavaScript脚本代码的程序叫JavaScript引擎

### 3.3.2 编写位置

js需要包括在<script>标签中，这个标签可以出现在页面的任何位置,一般建议写在HTML head标签中.

|  |
| --- |
| <head>  <script type=”text/javascript”>  //Js 代码  </script>  </head> |

### 3.3.3 基本语法

1. 变量

* 声明：使用var 如：var x=65; var y=”你好”;

1. 、变量的声明不需要指定数据类型，可以接受所有的数据类型
2. 、变量名区分大小写，abc和aBc是两个不同的变量

* 赋值：

x=44; x=”abc”; x=new Date();

1. 变量可以接受任何值
2. 声明和赋值也可同时进行 如var x=”abc”;
3. 函数

* 声明：使用function关键字，没有指定返回值一说！参数列表也没有指定参数类型

一说，因为js所有类型都使用var来声明

1. 函数在js中也是一种对象，可以将函数的引用赋值给变量

|  |
| --- |
| var abc=function(a,b){ return a+b;} |

1. 为函数起个名字叫add

|  |
| --- |
| function add(a,b){ return a+b;} |

* 调用

1. 如果是声明方式第一种的，使用变量名+()的方式进行调用

abc(1,2);

1. 如果是声明方式第二种的，直接使用函数名调用

add(1,2);

1. 注意：js调用函数的时候不会检查参数列表，所以js中没有重载一说，add(1,2)；add(1)；add(1,”abc”)；add(“abc”)；add(1,”666”,true);add()；都是调用的同一个方法。
2. 对象

* 创建

1. var obj = new Object();
2. var obj = {};

* 为对象的属性赋值

1. 动态添加

|  |
| --- |
| var obj = new Object();  obj.name = “张三”;  obj.age = 18;  obj.work = function(){ //对象的属性可以赋值为一个函数  alert(“我在工作！”);  } |

1. 创建时指定

|  |
| --- |
| obj = {  name:”张三”,  age:18,  work:function(){  alert(“我在工作”);  }  } |

* 使用对象的属性,直接以.的方式来访问

|  |
| --- |
| alert(obj.name)  obj.work(); |

### 3.3.4 其他语法

java中的for，while，if-else，switch，try-catch，break，continue，以及各种运算符，在js中也是按照同样的方式使用的。这里不再赘述。

### 3.3.5 JavaScript事件

1. 什么是事件

事件就是浏览器或者用户交互时触发的行为。比如按钮点击，表单提交，鼠标滑动等等…

1. 事件的分类

* 系统事件：如：文档加载完成。
* 用户事件：如：鼠标移入移出，单击双击等。

1. 事件的触发

* 系统事件会由系统触发，如window.onload事件，用户事件由用户行为触发如click事件。
* 常见的系统事件

window.onload

* 常见的用户事件

onclick当用户点击某个对象时调用的事件句柄。

onblur 元素失去焦点。

onfocus元素获得焦点。

onchange 域的内容被改变。

onkeydown某个键盘按键被按下

onmouseover 鼠标移入

onmouseout 鼠标移出

1. 事件响应

* 我们希望某个事件发生的时候我们可以做一些事情。这个称为事件的响应，比如用户点击了一个按钮，我弹出一个框告诉用户，你成功的点击了这个按钮。

事件触发后我们要执行的方法称为响应函数。

* 响应函数与事件关联

1. 直接为事件绑定响应函数

|  |
| --- |
| window.onload = function(){  alert(“文档加载完成了！”)  } |

1. 使用标签的事件属性

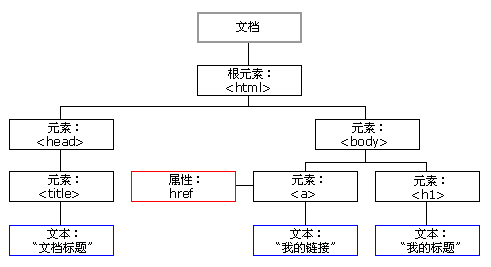
|  |
| --- |
| <a href=”atguigu.com” onclick=”gotoguigu()”>尚硅谷</a>  我们在<script></script>中定义这个函数  <script type=”text/javascript”>  function gotoguigu(){  alert(“我要去上硅谷”)  }  </script> |

### 3.2.6 DOM

1. 什么是DOM

Document Object Model(文档对象模型)，我们浏览器把整个网页会当成一个大的对象，利用面向对象的方式操作网页内容。

1. DOM模型



1. document对象

document对象是window对象的一个属性，代表当前整个HTML文档，将这个文档抽象成了document对象，包含了整个dom树的所有内容。

其本质是window.document，而window.可以省略。直接使用document

1. DOM操作

* 查询元素

|  |  |  |
| --- | --- | --- |
| 根据id值查询 | document.getElementById(“id值”) | 一个具体的元素节点 |
| 根据标签名查询 | document.getElementsByTagName(“标签名”) | 元素节点数组 |
| 根据name属性值查询 | document.getElementsByName(“name值”) | 元素节点数组 |

* 属性操作

|  |  |
| --- | --- |
| 读取属性值 | 元素对象.属性名 |
| 修改属性值 | 元素对象.属性名=新的属性名 |

* 文本操作

|  |  |
| --- | --- |
| 获取元素内容 | 元素.innerHTML |
| 给元素内容赋值 | 元素.innerHTML=值 |

# 第4章 服务端技术

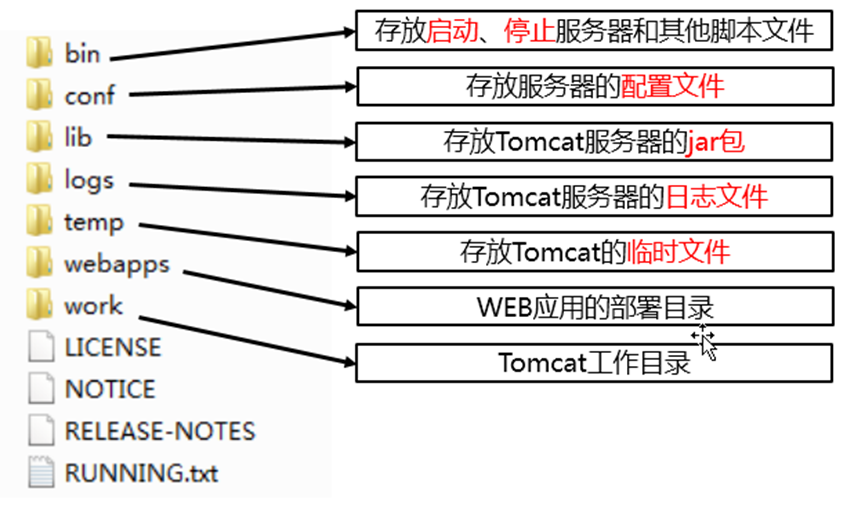
## 4.1 Web服务器

### 4.1.1 常见的Web服务器

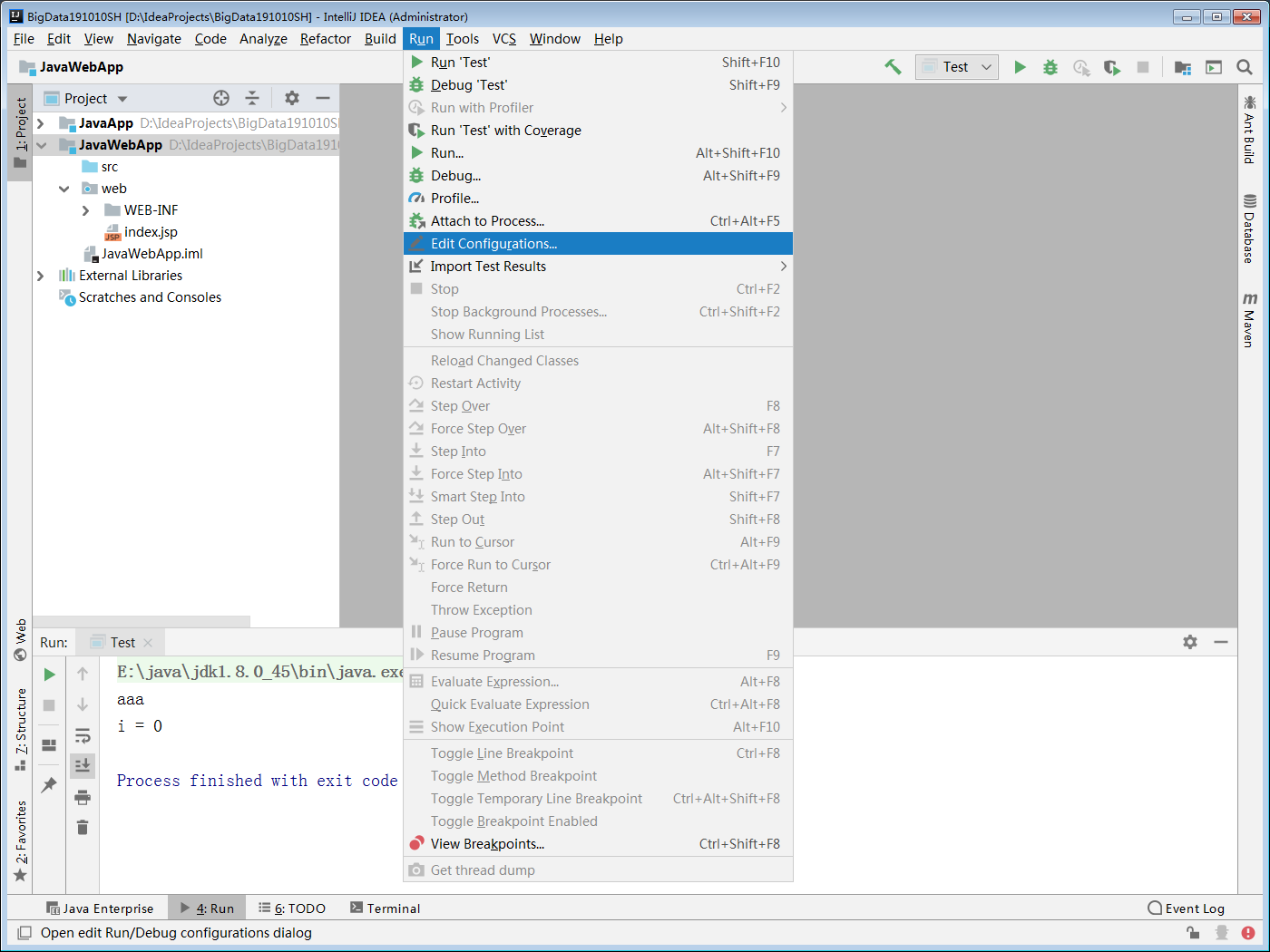
1. Web服务器主要用来接收客户端发送的请求和响应客户端请求。
2. **Tomcat（Apache）**：当前应用最广的JavaWeb服务器；
3. JBoss（Redhat红帽）：支持JavaEE，应用比较广EJB容器 –> SSH轻量级的框架代替
4. GlassFish（Orcale）：Oracle开发JavaWeb服务器，应用不是很广；
5. Resin（Caucho）：支持JavaEE，应用越来越广；
6. Weblogic（Orcale）：要钱的！支持JavaEE，适合大型项目；
7. Websphere（IBM）：要钱的！支持JavaEE，适合大型项目

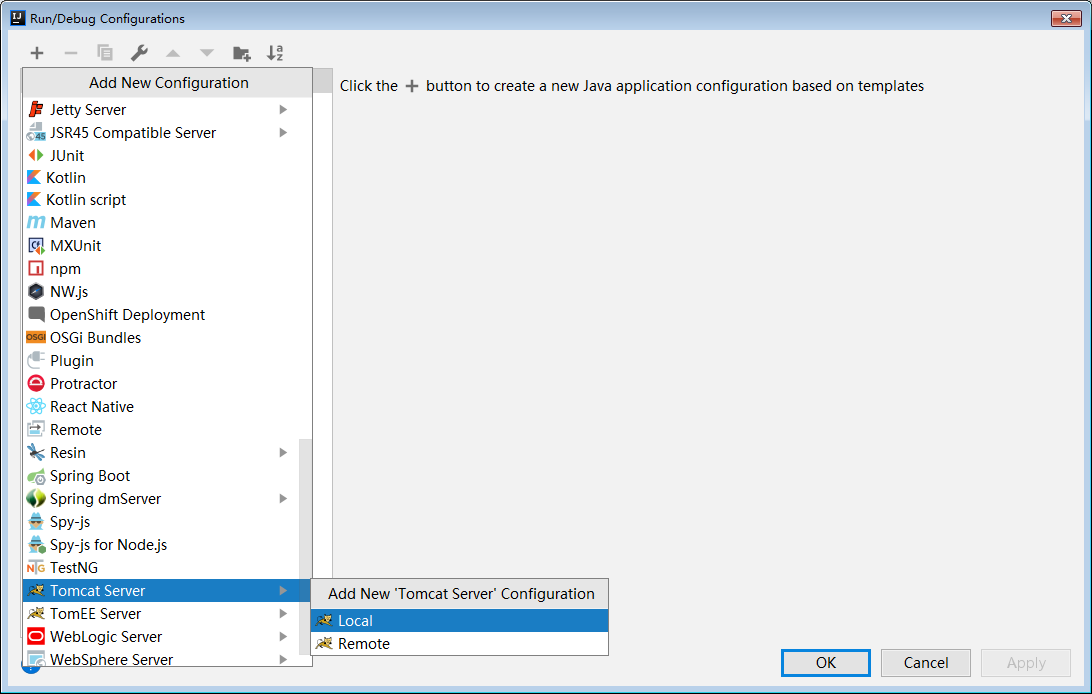
### 4.1.2 Tomcat服务器的安装及配置

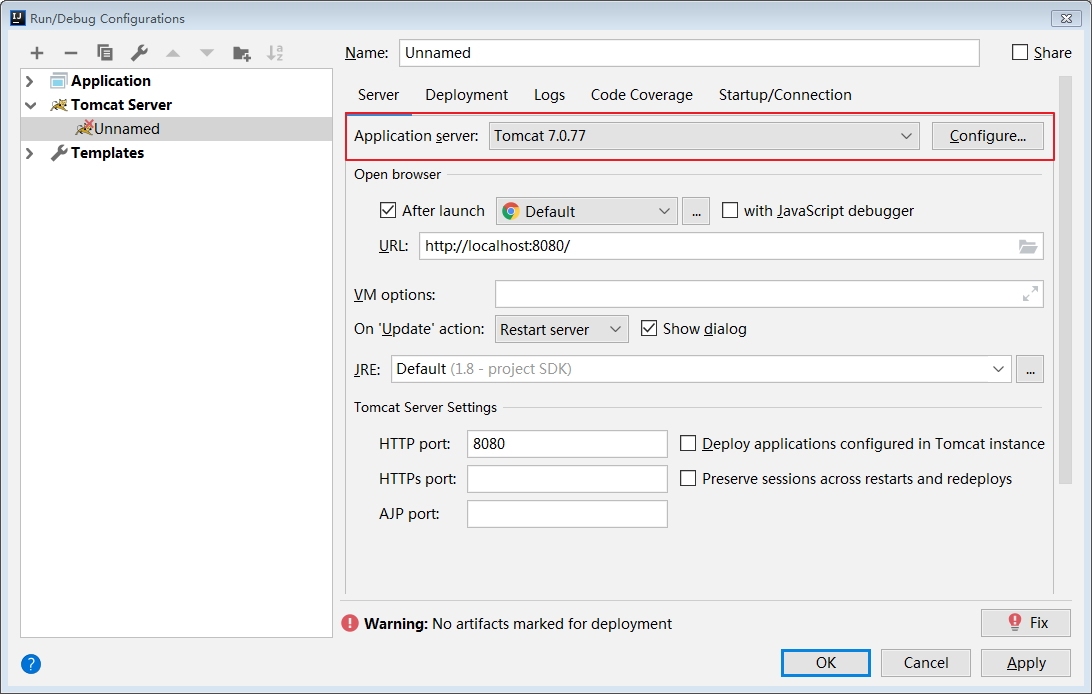
1. 将Tomcat的安装包解压到磁盘的任意位置(非中文无空格)
2. Tomcat服务的目录结构



### 4.1.3 在IDEA中配置tomcat

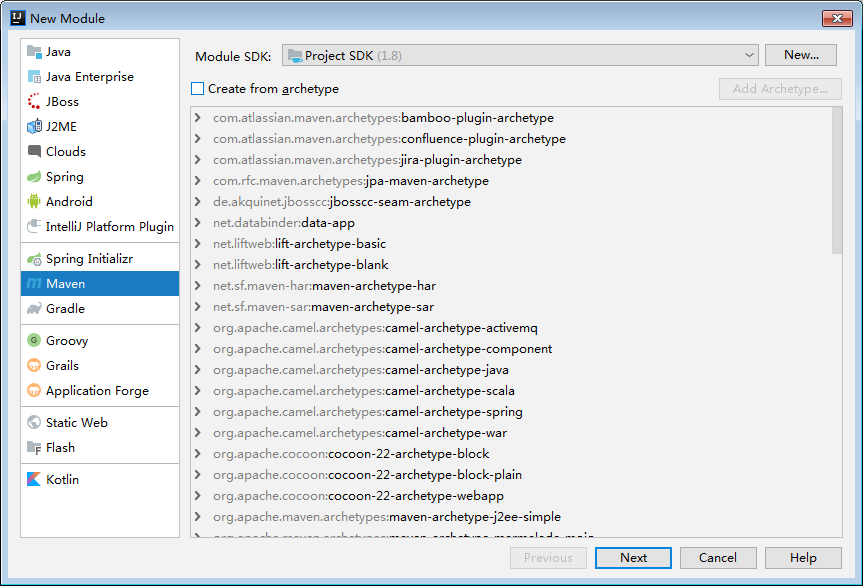


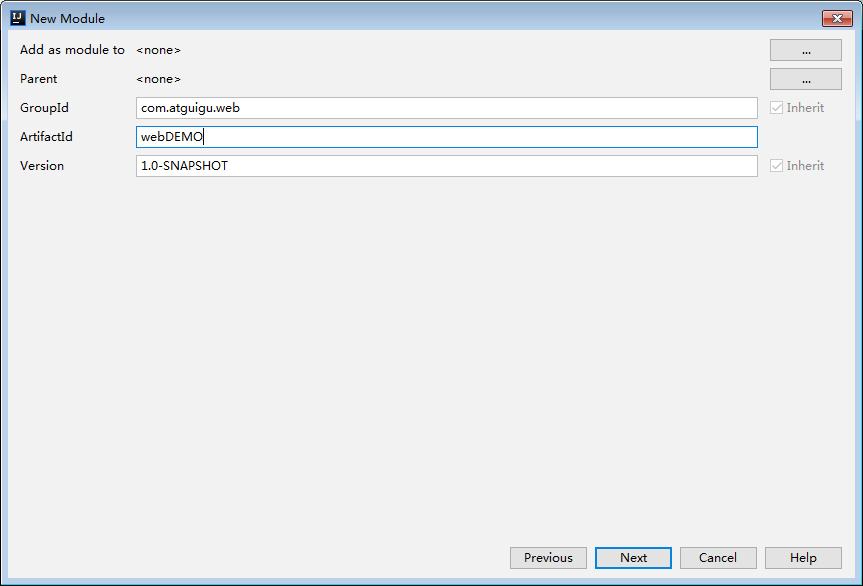


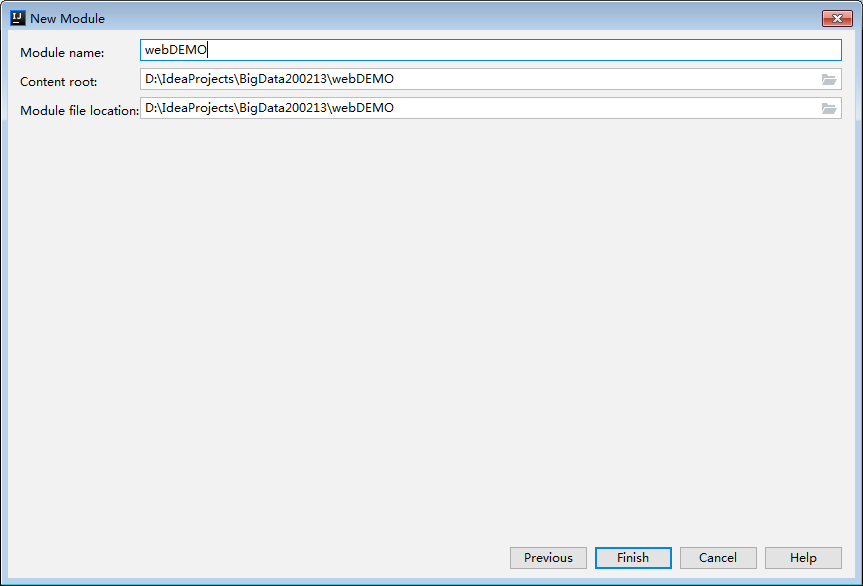


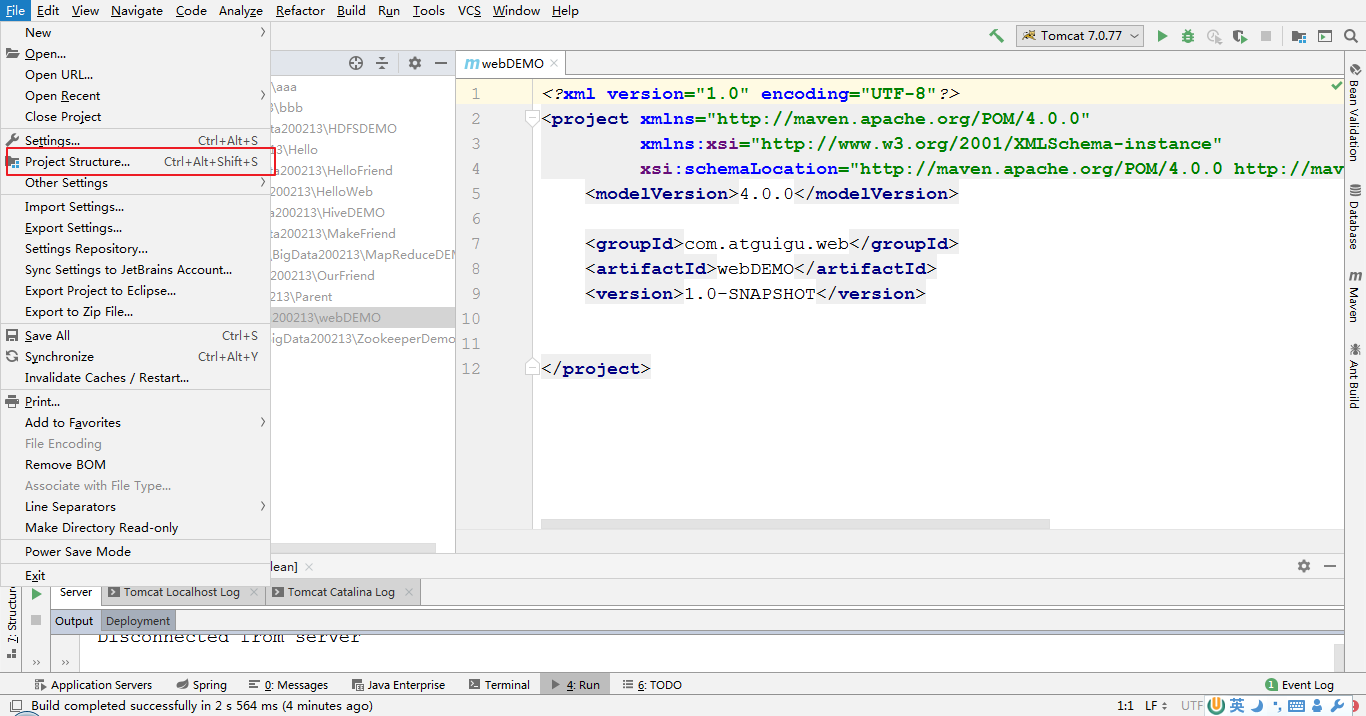
## 4.2 web Module

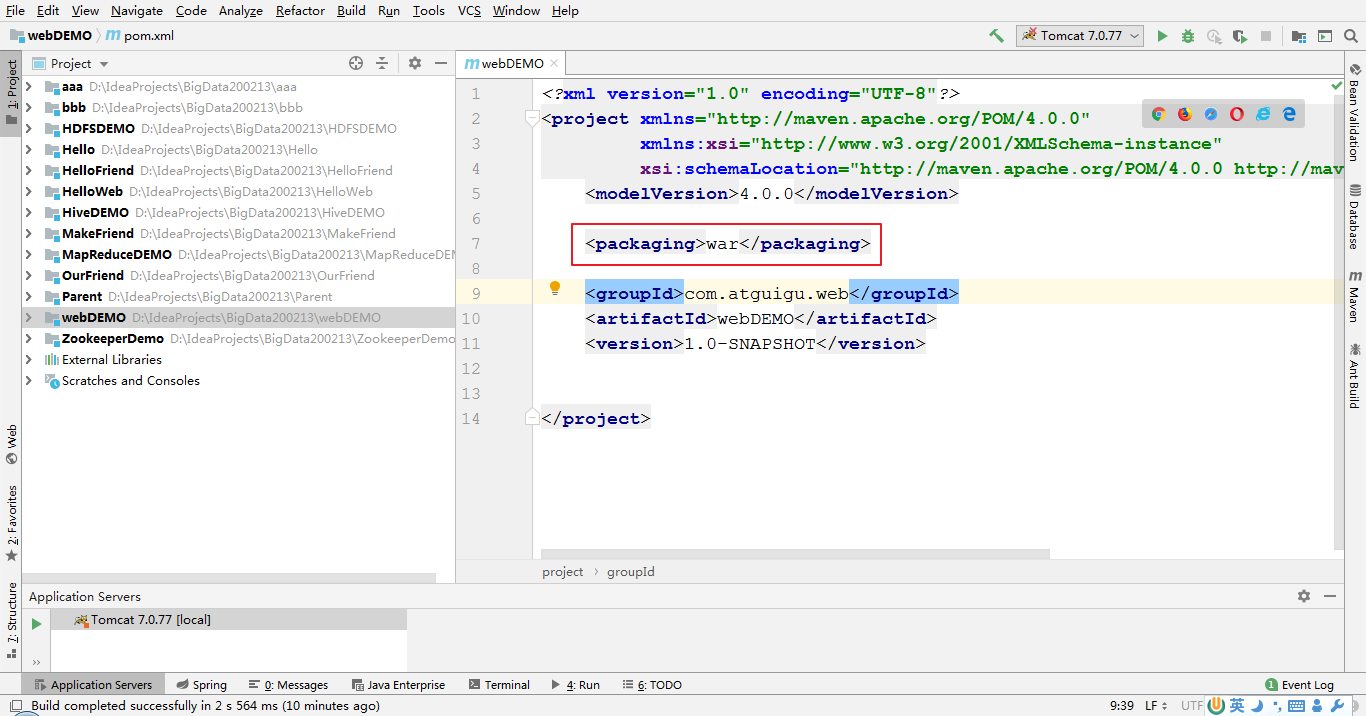
### 4.2.1 创建web Module

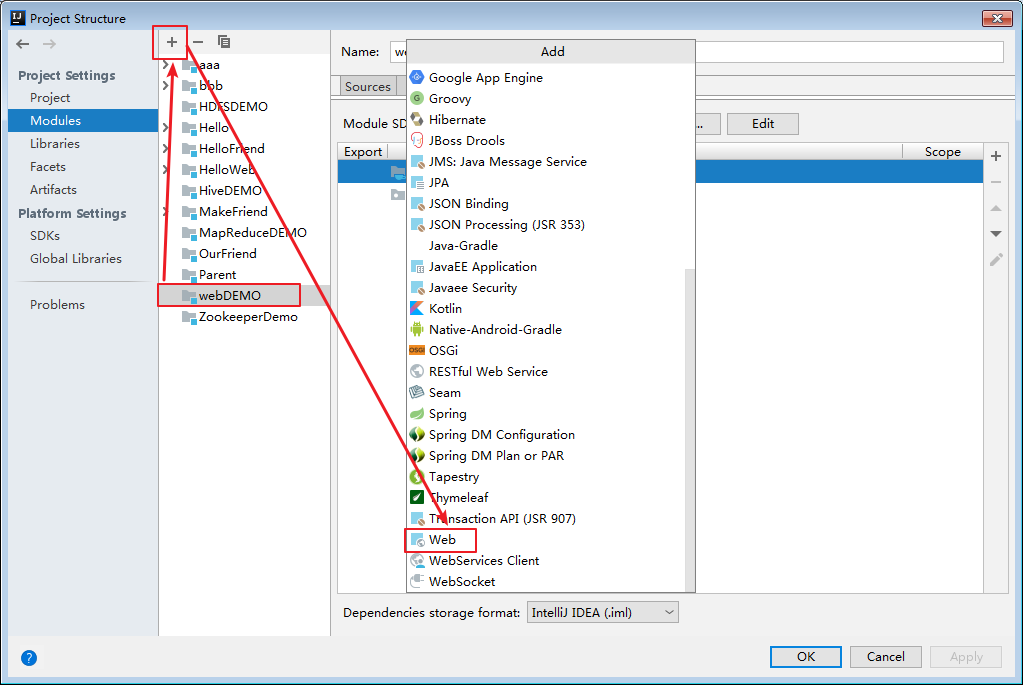


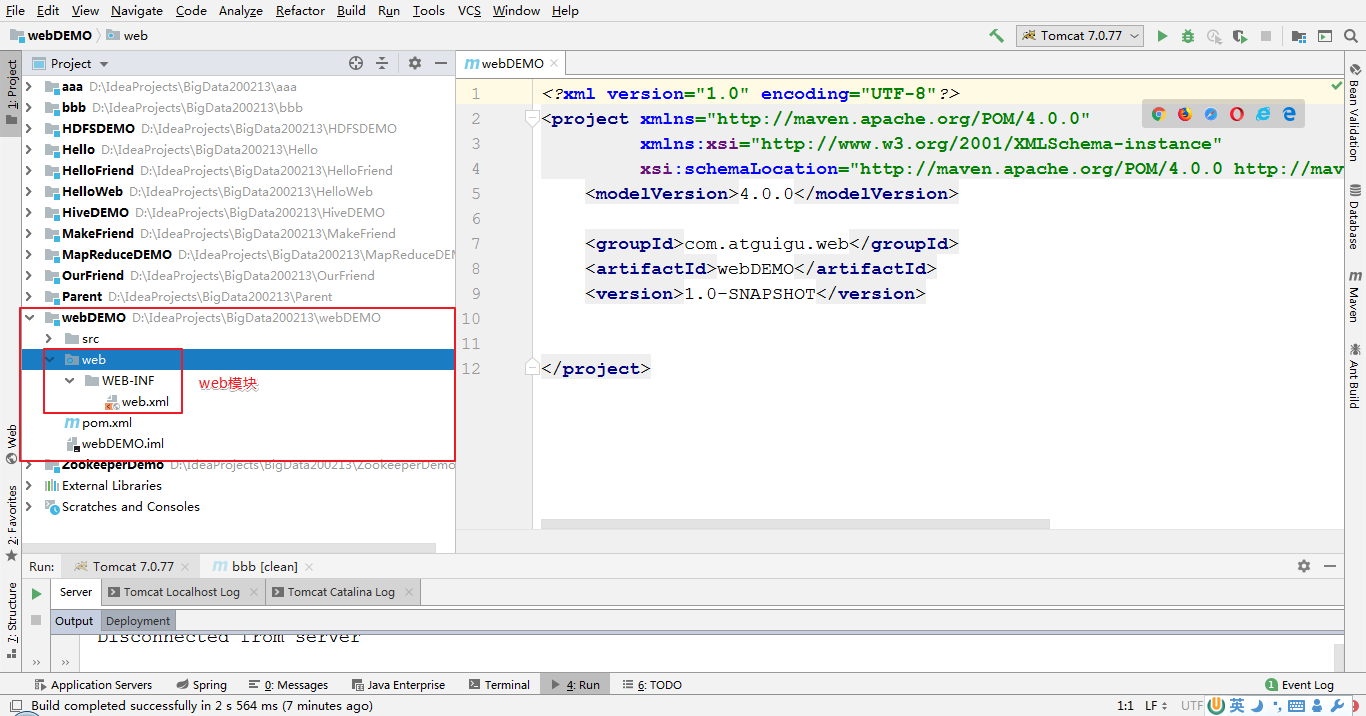




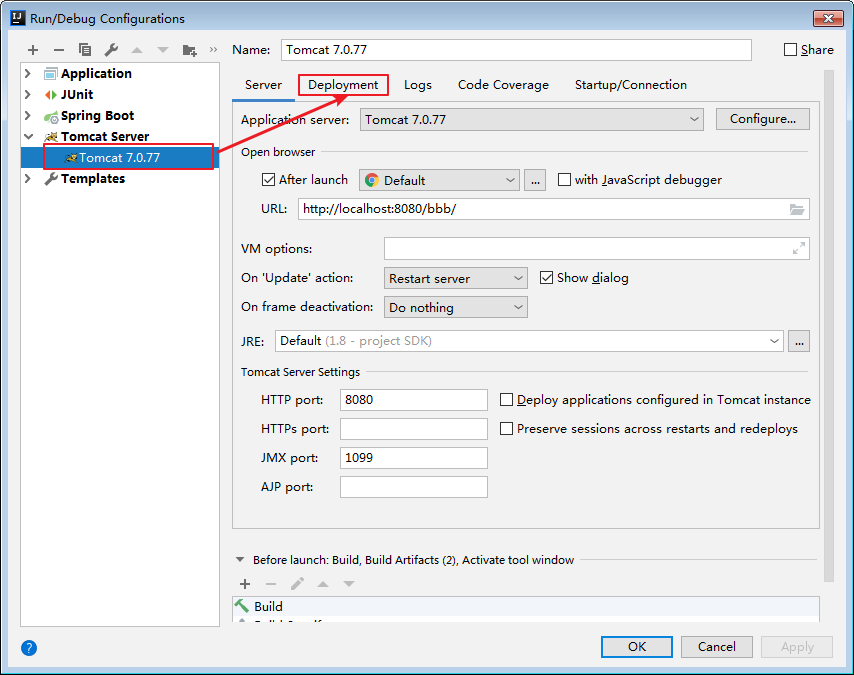


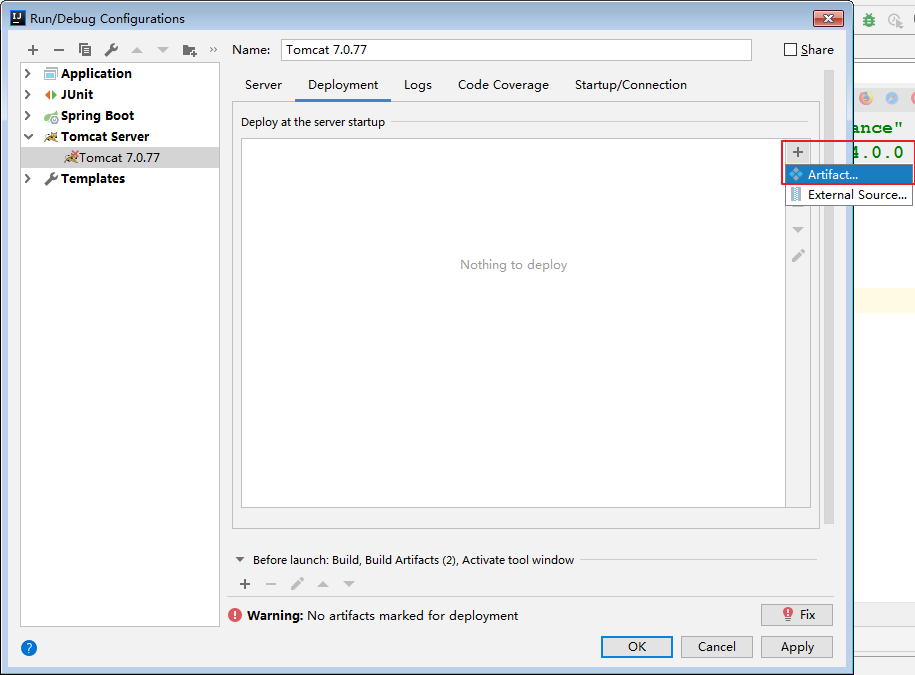


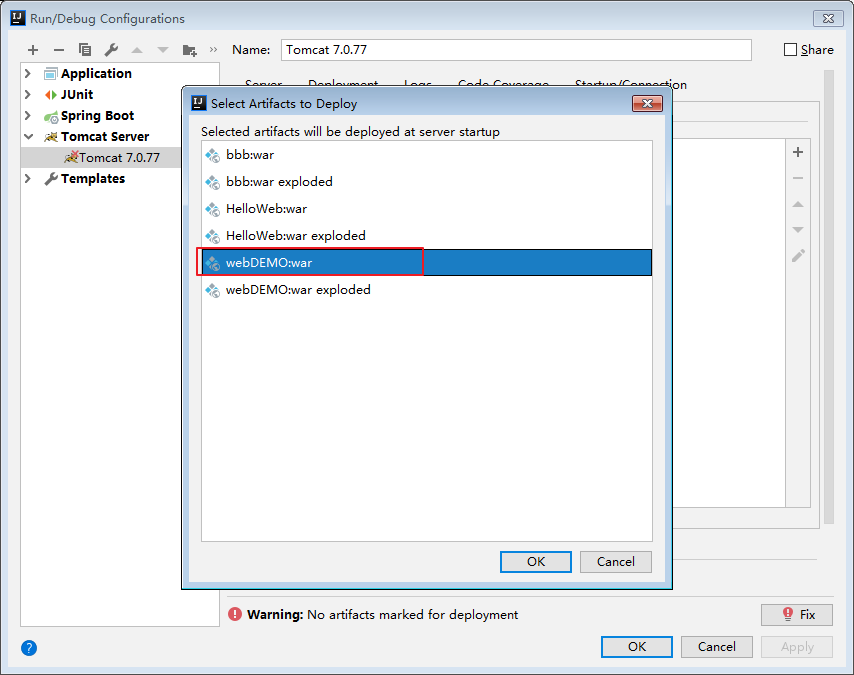


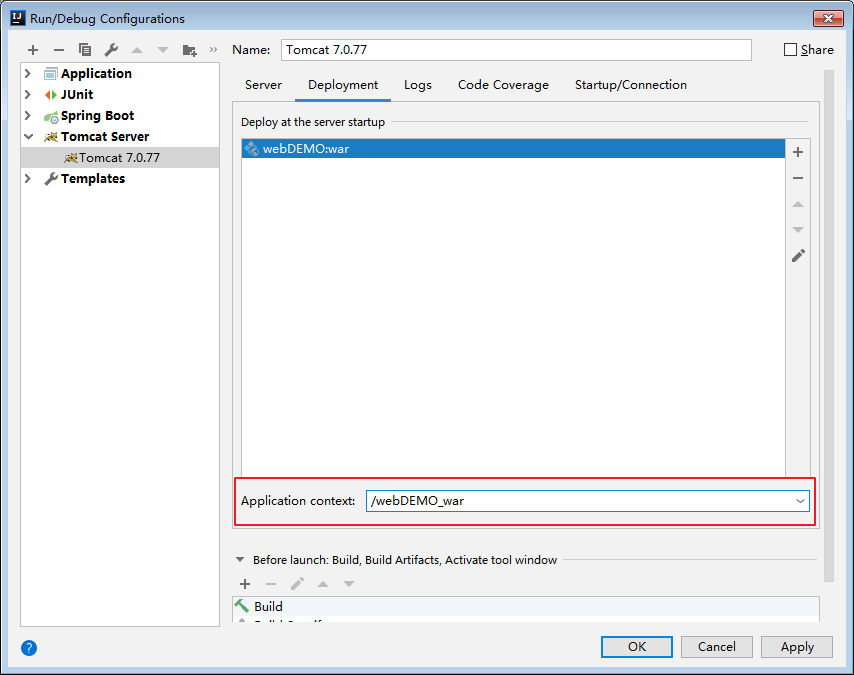


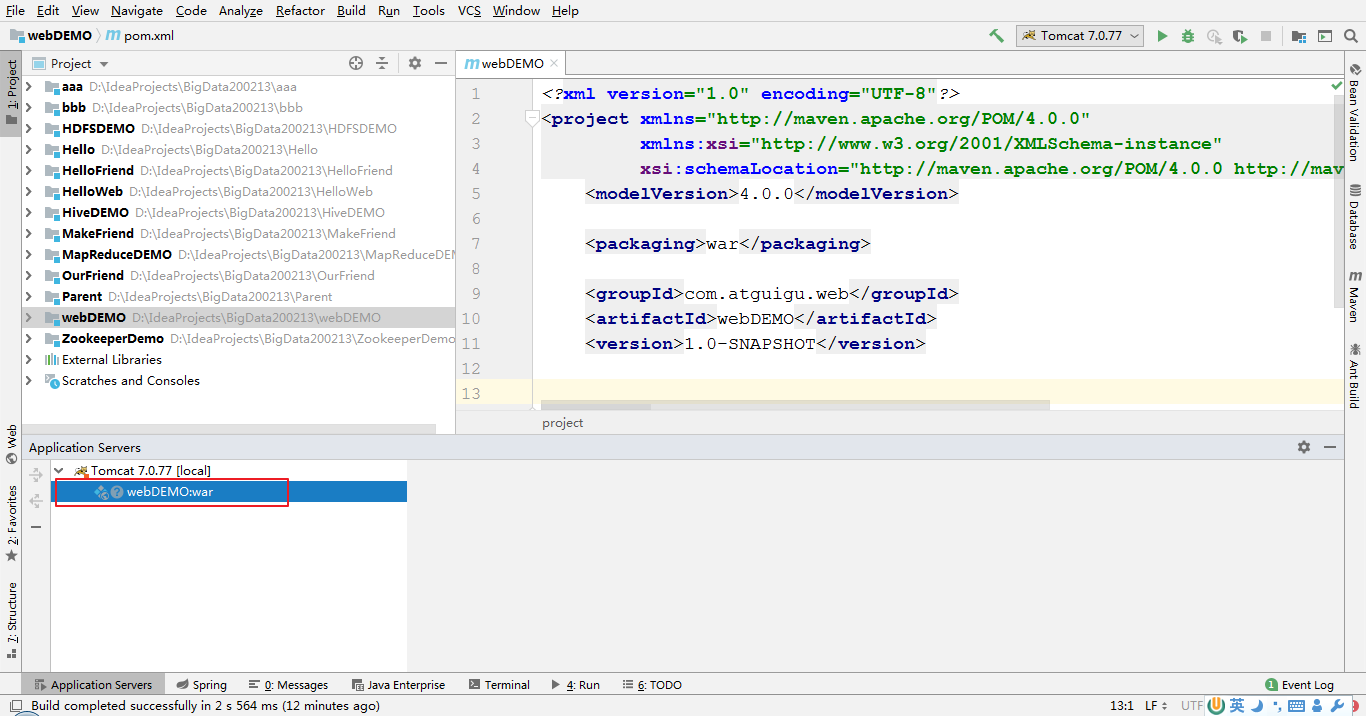
### 4.2.2 部署web Module











## 4.3 Servlet

### 4.3.1 什么是Servlet

1. Servlet是Sun公司制定的一套技术标准，包含与Web应用相关的一系列接口，是Web应用实现方式的宏观解决方案。而具体的Servlet容器负责提供标准的实现。
2. Servlet作为服务器端的一个组件，它的本意是“服务器端的小程序”。Servlet的实例对象由Servlet容器负责创建；Servlet的方法由容器在特定情况下调用；Servlet容器会在Web应用卸载时销毁Servlet对象的实例。
3. 简单可以理解为 Servlet就是用来处理客户端的请求 及 响应客户端.

### 4.3.2 Servlet开发规则

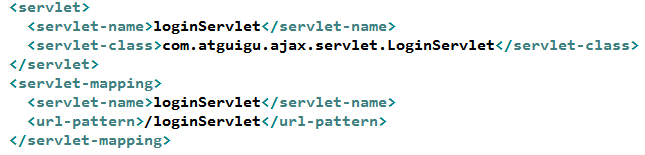
1. 实际编码通过继承HttpServlet来完成Servlet的开发



### 4.3.3 Servlet类的相关方法:

1. doGet方法 Servlet中用于处理get请求的方法
2. doPost方法 Servlet中用于处理post请求的方法
3. service方法
4. 在Servlet的顶层实现中，在service方法中调用的具体的doGet或者是doPost
5. 在实际开发Servlet的过程中，可以选择重写doGet以及doPost 或者 直接重写service方法来处理请求。

### 4.3.4 Servlet在web.xml中的配置



## 4.4 获取请求参数值

1. HttpServletRequest
2. 该接口是ServletRequest接口的子接口，封装了HTTP请求的相关信息，由Servlet容器创建其实现类对象并传入service(ServletRequest req, ServletResponse res)方法中。以下我们所说的HttpServletRequest对象指的是容器提供的HttpServletRequest实现类对象。
3. HttpServletRequest对象的主要功能有

获取请求参数

在请求域中绑定数据

将请求转发给另外一个URL地址 [转发]

## 4.5 响应结果

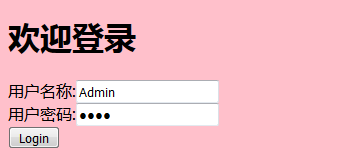
1. HttpServletResponse
2. 该接口是ServletResponse接口的子接口，封装了HTTP响应的相关信息，由Servlet容器创建其实现类对象并传入service(ServletRequest req, ServletResponse res)方法中。以下我们所说的HttpServletResponse对象指的是容器提供的HttpServletResponse实现类对象
3. 主要功能

使用PrintWriter对象向浏览器输出数据

实现请求的重定向[重定向]

## 4.6 具体登录功能的实现步骤

1. 在登录页面中录入用户名和密码，点击登录按钮提交登录请求



1. 在LoginServlet中通过request对象获取到页面表单提交的用户名和密码

|  |
| --- |
| //获取用户输入的用户名和密码  String username = req.getParameter("username");  String password = req.getParameter("password"); |

1. 调用Dao对象，将用户提交的用户名和密码与数据库的用户表的数据进行匹配

|  |
| --- |
| UserDao dao = **new** UserDaoJdbcImpl();  User user = dao.findUserByUsernameAndPassword(username, password); |

1. 得出结果，完成响应.

|  |
| --- |
| //设置响应头信息  resp.setContentType("text/html;charset=utf-8");  //获取输出流  PrintWriter out = resp.getWriter();  **if**(user == **null** ) {  //登录失败  out.println("登录失败！！！！");  }**else** {  //登录成功  out.println("登录成功！！！！");  }  out.close(); |

## 4.7 请求重定向 redirect

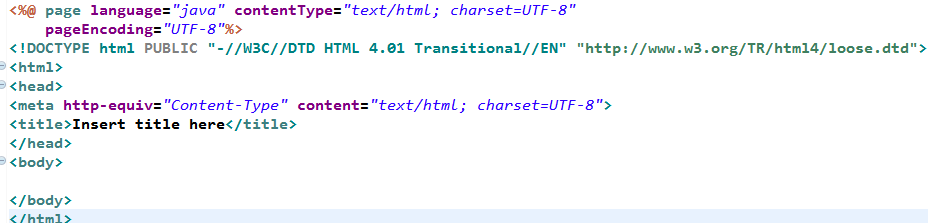
1. Servlet接收到浏览器端请求并处理完成后，给浏览器端一个特殊的响应，这个特殊的响应要求浏览器去请求一个新的资源，整个过程中浏览器端会发出两次请求，且浏览器地址栏会改变为新资源的地址。
2. 重定向的情况下，原Servlet和目标资源之间就不能共享请求域数据了
3. 实现重定向的API



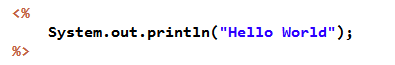
4) 登录失败后重定向到登录页面

## 4.8 JSP页面

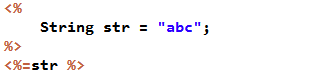
1. JSP全称Java Server Pages，顾名思义就是运行在java服务器中的页面，也就是在我们JavaWeb中的动态页面，其本质就是一个Servlet。
2. 其本身是一个动态网页技术标准，它的主要构成有HTML网页代码、Java代码片段、JSP标签几部分组成，后缀是.jsp
3. 相比于Servlet，JSP更加善于处理显示页面，而Servlet跟擅长处理业务逻辑，两种技术各有专长，所以一般我们会将Servlet和JSP结合使用，Servlet负责业务，JSP负责显示。
4. 一般情况下， 都是Servlet处理完的数据，转发到JSP，JSP负责显示数据的工作
5. JSP的基本语法:



1. JSP的脚本元素
2. 脚本片段是嵌入到JSP中Java代码段，格式以<%开头，%>结尾，两个%号之间就可以编写Java代码了



1. JSP的表达式
2. JSP表达式用来直接将Java变量输出到页面中，格式以<%=开头，以%>结尾，中间是我们要输出的内容



1. JSP的内置对象
   1. out（JspWriter）：相当于response.getWriter()获取的对象，用于在页面中显示信息。
   2. config（ServletConfig）：对应Servlet中的ServletConfig对象。
   3. page（Object）：对应当前Servlet对象，实际上就是this。
   4. pageContext（PageContext）：当前页面的上下文，也是一个域对象。
   5. exception（Throwable）：错误页面中异常对象
   6. request（HttpServletRequest）：HttpServletRequest对象
   7. response（HttpServletResponse）：HttpServletResponse对象
   8. application（ServletContext）：ServletContext对象
   9. session（HttpSession）：HttpSession对象
2. EL表达式
3. EL是JSP内置的表达式语言，用以访问页面的上下文以及不同作用域中的对象 ，取得对象属性的值，或执行简单的运算或判断操作。EL在得到某个数据时，会自动进行数据类型的转换。
4. **EL表达式用于代替JSP表达式(<%= %>)在页面中做输出操作。**
5. EL表达式仅仅用来读取数据，而不能对数据进行修改。
6. 使用EL表达式输出数据时，**如果有则输出数据，如果为null则什么也不输出。**
7. **EL表达式的语法:**



1. EL取值的四个域:

pageScope

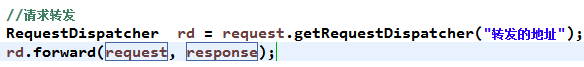
requestScope

sessionScope

applicationScope

## 4.9 请求转发

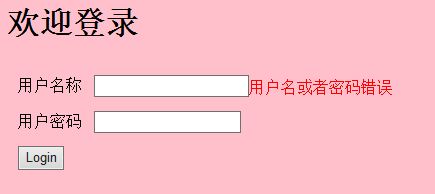
1. Servlet接收到浏览器端请求后，进行一定的处理，先不进行响应，而是在服务器端内部“转发”给其他Servlet程序继续处理。在这种情况下浏览器端只发出了一次请求，浏览器地址栏不会发生变化，用户也感知不到请求被转发了。
2. 转发请求的Servlet和目标Servlet共享同一个request对象。
3. 实现转发的API



## 4.10 重定向与转发的区别

|  |  |  |
| --- | --- | --- |
|  | 转发 | 重定向 |
| 浏览器地址栏 | 不改变 | 改变 |
| 发送请求次数 | 1 | 2 |
| 能否共享request对象数据 | 能 | 否 |
| 目标资源：WEB-INF下的资源 | 能访问 | 不能访问 |
| Request中绑定的数据是否可以传递 | 能 | 不能 |

## 4.11 页面中错误提示的功能效果



## 4.12 Cookie

1. HTTP是无状态协议，服务器不能记录浏览器的访问状态，也就是说服务器不能区分中两次请求是否由一个客户端发出。这样的设计严重阻碍的Web程序的设计。如：在我们进行网购时，买了一条裤子，又买了一个手机。由于http协议是无状态的，如果不通过其他手段，服务器是不能知道用户到底买了什么。而Cookie就是解决方案之一。
2. Cookie实际上就是服务器保存在浏览器上的一段信息。浏览器有了Cookie之后，每次向服务器发送请求时都会同时将该信息发送给服务器，服务器收到请求后，就可以根据该信息处理请求。
3. Cookie的用途

网上商城购物车

用户登录状态的保持

1. Cookie的限制性
   1. Cookie作为请求或响应报文发送，无形中增加了网络流量。
   2. Cookie是明文传送的安全性差。
   3. 各个浏览器对Cookie有限制，使用上有局限
2. Cookie的具体使用
3. 创建cookie



1. 读取cookie



## 4.13 Session

1. 使用Cookie有一个非常大的局限，就是如果Cookie很多，则无形的增加了客户端与服务端的数据传输量。而且由于浏览器对Cookie数量的限制，注定我们不能再Cookie中保存过多的信息，于是Session出现。
2. Session的作用就是在服务器端保存一些用户的数据，然后传递给用户一个名字为JSESSIONID的Cookie，这个JESSIONID对应这个服务器中的一个Session对象，通过它就可以获取到保存用户信息的Session。
3. Session的工作原理
   1. Session的创建时机是在request.getSession()方法第一次被调用时。
   2. Session被创建后，同时还会有一个名为JSESSIONID的Cookie被创建。
   3. 这个Cookie的默认时效就是当前会话。
   4. 简单来说，Session机制也是依赖于Cookie来实现的
4. Session的具体使用



1. Session的时效问题

Session默认有效时间为30分钟，可以在服务器的web.xml配置中修改.

## 4.14 登录成功跳转主页面显示登录用户

# 第5章 异步请求Ajax

## 5.1 什么是Ajax

1. AJAX 是 Asynchronous JavaScript And XML 的简称。直译为，异步的JS和XML。
2. AJAX的实际意义是，不发生页面跳转、异步载入内容并改写页面内容的技术。
3. AJAX也可以简单的理解为通过JS向服务器发送请求。

## 5.2 异步处理

1. 同步处理

AJAX出现之前，我们访问互联网时一般都是同步请求，也就是当我们通过一个页面向 服务器发送一个请求时，在服务器响应结束之前，我们的整个页面是不能操作的，也就 是直观上来看他是卡主不动的。

这就带来了非常糟糕的用户体验。首先，同步请求时，用户只能等待服务器的响应，而 不能做任何操作。其次，如果请求时间过长可能会给用户一个卡死的感觉。最后，同步 请求的最大缺点就是即使整个页面中只有一小部分内容发生改变我们也要刷新整个页 面。

1. 异步处理

而异步处理指的是我们在浏览网页的同时，通过AJAX向服务器发送请求，发送请求的过程中我们浏览网页的行为并不会收到任何影响，甚至主观上感知不到在向服务器发送请求。当服务器正常响应请求后，响应信息会直接发送到AJAX中，AJAX可以根据服务器响应的内容做一些操作。

使用AJAX的异步请求基本上完美的解决了同步请求带来的问题。首先，发送请求时不会影响到用户的正常访问。其次，即使请求时间过长，用户不会有任何感知。最后，AJAX可以根据服务器的响应信息局部的修改页面，而不需要整个页面刷新。

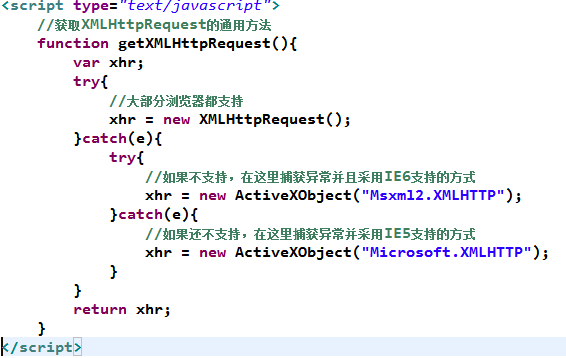
## 5.3 异步请求对象

1. XMLHttpRequest对象是AJAX中非常重要的对象，所有的AJAX操作都是基于该对象的。

XMLHttpRequest对象用来封装请求报文，我们向服务器发送的请求信息全部都需要封装到该对象中。

这里需要稍微注意一下，XMLHttpRequest对象并没有成为标准，但是现在的主流浏览器都支持该对象，而一些如IE6的老版本浏览器中的创建方式有一些区别，但是问题不大。

### 5.4.1 XMLHttpRequest对象的获取



### 5.4.2 XMLHttpRequest对象的方法

* 1. open(method,url,async)

open()用于设置请求的基本信息，接收三个参数。

* + - 1. method
         * 请求的方法：get或post
         * 接收一个字符串
      2. url
         * 请求的地址，接收一个字符串
      3. Assync
         * 发送的请求是否为异步请求，接收一个布尔值。
         * true 是异步请求
         * false 不是异步请求（同步请求）
  1. send(string)

send()用于将请求发送给服务器，可以接收一个参数

* + - 1. string参数
         * 该参数只在发送post请求时需要。
         * string参数用于设置请求体
  1. setRequestHeader(header,value)

用于设置请求头

* + - 1. header参数
         * 字符串类型，要设置的请求头的名字
      2. value参数
         * 字符串类型，要设置的请求头的值

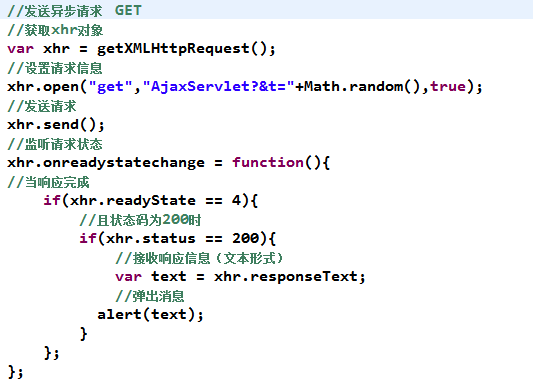
### 5.4.3 XMLHttpRequest对象的属性

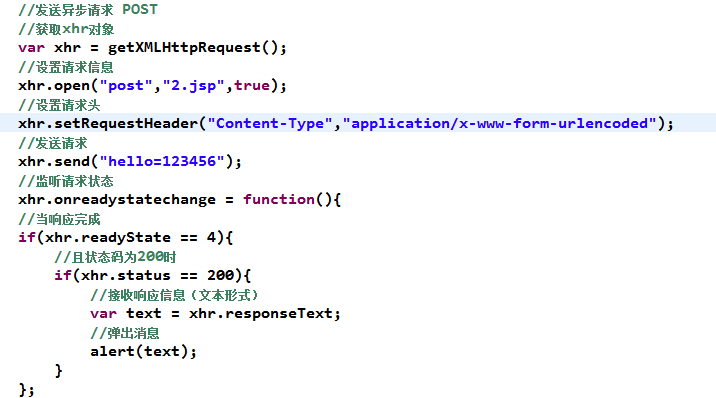
1. readyState
2. 描述XMLHttpRequest的状态
3. 一共有五种状态分别对应了五个数字：
   * + - 0 ：请求尚未初始化，open()尚未被调用
       - 1 ：服务器连接已建立，send()尚未被调用
       - 2 ：请求已接收，服务器尚未响应
       - 3 ：请求已处理，正在接收服务器发送的响应
       - 4 ：请求已处理完毕，且响应已就绪。
   1. status
      1. 请求的响应码
         * 200 响应成功
         * 404 页面为找到
         * 500 服务器内部错误

… … … …

* 1. onreadystatechange

1. 该属性需要指向一个函数
2. 该函数会在readyState属性发生改变时被调用
   1. responseText
3. 获得字符串形式的响应数据。
   1. responseXML（用的比较少）
4. 获得 XML 形式的响应数据。
   1. 示例代码





## 5.4. JQuery对异步的支持

1. JQuery是当前比较主流的 JavaScript 库，封装了很多预定义的对象和实现函数，帮助使用者建立有高难度交互的页面，并且兼容大部分主流

的浏览器.

JQuery对同样提供了对Ajax的支持，可以更加方便快速的进行Ajax的开发，相关的方法有$.get $.post $.ajax等.

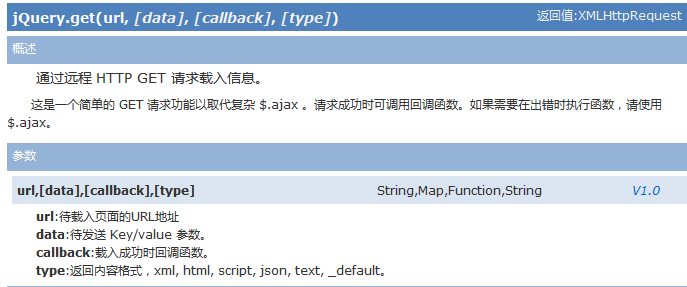
JQuery的对象的本质就是dom对象的数组/集合

1. JQuery对象与dom对象的相互转换

JS转JQuery: var jObj = $(dObj);

JQuery转JS: var dObj = jObj[0] 或者 var dObj = jObj.get(0)

1. $.get方法

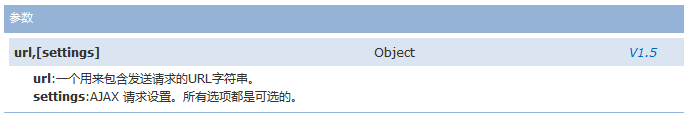


1. $.post方法

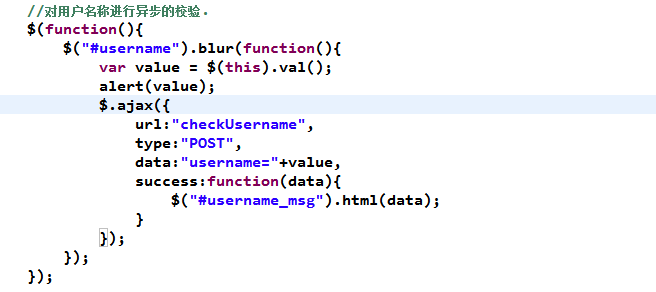


1. $.ajax方法

jQuery 底层 AJAX 实现。简单易用的高层实现见 $.get, $.post 等。$.ajax() 返回其创建的 XMLHttpRequest 对象。大多数情况下你无需直接操作该函数，除非你需要操作不常用的选项，以获得更多的灵活性。



1. 具体的示例代码



## 5.5 基于异步请求校验用户名

# 第6章 JSON

## 6.1 JSON 简介

1. AJAX一开始使用的时XML的数据格式，XML的数据格式非常简单清晰，容易编写，但是由于XML中包含了过多的标签，以及十分复杂的结构，解析起来也相对复杂，所以目前来讲，AJAX中已经几乎不使用XML来发送数据了。取而代之的是一项新的技术JSON。
2. JSON是JavaScript Object Notation 的缩写，是JS提供的一种数据交换格式。
3. JSON对象本质上就是一个JS对象，但是这个对象比较特殊，它可以直接转换为字符串，在不同语言中进行传递，通过工具又可以转换为其他语言中的对象。
4. 例，有如下一个JSON对象：
5. {“name”:”sunwukong” , ”age”:18 , ”address”:”beijing” }
6. 这个对象中有三个属性name、age和address
7. 如果将该对象使用单引号引起了，那么他就变成了一个字符串
8. ‘{“name”:”sunwukong” , ”age”:18 , ”address”:”beijing” }’
9. 变成字符串后有一个好处，就是可以在不同语言之间传递。
10. 比如，将JSON作为一个字符串发送给Servlet，在Java中就可以把JSON字符串转换为一个Java对象。

## 6.2 JSON通过6种数据类型来表示

1. 字符串
   * + 例子：”字符串”
     + 注意：不能使用单引号
2. 数字：
   * + 例子：123.4
3. 布尔值：
   * + 例子：true、false
4. null值:
   * + 例子：null
5. 对象
   * + 例子：{“name”:”sunwukong”, ”age”:18}
6. 数组
   * + 例子：[1,”str”,true]

## 6.3 在JS中操作JSON

1. 创建JSON对象
   * + var json = {“name1”:”value1”,”name2”:”value2” , “name3”:[1,”str”,true]};
     + var json = [{“name1”:”value1”},{“name2”:”value2”}];
2. JSON对象转换为JSON字符串
   * + JSON.stringify(JSON对象)
3. JSON字符串转换为JSON对象
   * + JSON.parse(JSON字符串)

## 6.4 在Java中操作JSON

1. 在Java中可以从文件中读取JSON字符串，也可以是客户端发送的JSON字符串，所以第一个问题，我们先来看如何将一个JSON字符串转换成一个Java对象。
2. 首先解析JSON字符串我们需要导入第三方的工具，目前主流的解析JSON的工具大概有三种json-lib、jackson、gson。三种解析工具相比较json-lib的使用复杂，且效率较差。而Jackson和gson解析效率较高。使用简单，这里我们以gson为例讲解。
3. Gson是Google公司出品的解析JSON工具，使用简单，解析性能好。
4. Gson中解析JSON的核心是Gson的类，解析操作都是通过该类实例进行。
5. JSON字符串转换为对象

|  |
| --- |
| String json = "{\"name\":\"张三\",\"age\":18}";  Gson gson = **new** Gson();  //转换为集合  Map<String,Object> stuMap = gson.fromJson(json, Map.**class**);  //如果编写了相应的类也可以转换为指定对象  Student fromJson = gson.fromJson(json, Student.**class**); |

1. 对象转换为JSON字符串

|  |
| --- |
| Student stu = **new** Student("李四", 23);  Gson gson = **new** Gson();  //{"name":"李四","age":23}  String json = gson.toJson(stu);    Map<String , Object> map = **new** HashMap<String, Object>();  map.put("name", "孙悟空");  map.put("age", 30);  //{"age":30,"name":"孙悟空"}  String json2 = gson.toJson(map);    List<Student> list = **new** ArrayList<Student>();  list.add(**new** Student("八戒", 18));  list.add(**new** Student("沙僧", 28));  list.add(**new** Student("唐僧", 38));  //[{"name":"八戒","age":18},{"name":"沙僧","age":28},  {"name":"唐僧","age":38}]  String json3 = gson.toJson(list);  // 如果将一个数组格式的json字符串转换成java对象需要用到  //Gson提供的一个匿名内部类： TypeToken  TypeToken tk= new TypeToken<List<User>>(){};  List<User> list2 = gson.fromJson(json,tk.getType());  System.out.println(list2.get(0)); |

## 6.5 JQuery 异步请求返回JSON数据

1. Servlet 返回json数据

|  |
| --- |
| **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {    List<Employee> emps = **new** EmployeeDaoJdbcImpl().getAllEmps();  Gson gson = **new** Gson();  String jsonStr = gson.toJson(emps);  response.setContentType("text/html;charset=utf-8");  PrintWriter out = response.getWriter();  out.println(jsonStr);  out.close();  } |

1. 页面中处理 json数据

|  |
| --- |
| **function** getJsonStr(){  //通过JQuery发送异步请求， 将所有的员工信息通过json的格式返回  $.ajax({  url:"getEmpsJsonStr",  type:"post",  dataType:"json",  success:**function**(data){ // 会直接将后台返回的json字符串转换成js对象  **var** str = "<tr><th>Id</th><th>LastName</th><th>Email</th><th>Gender</th></tr>";  **for**(**var** i= 0 ;i <data.length;i++){  **var** emp = data[i];  str+="<tr align='center'><td>"  +emp.id+  "</td><td>"  +emp.lastName+  "</td><td>"  +emp.email+  "</td><td>"  +emp.gender+  "</td></tr>"  }  $("#tb").html(str);  }  });  } |

|  |
| --- |
| <body>  <input type=*"button"* value=*"getJsonStr"* onclick="getJsonStr();"/>  <table id=*"tb"* border=*"1px"* align=*"center"* width=*"60%"* cellspacing=*"0px"* >  </table>  </body> |