Java WEB

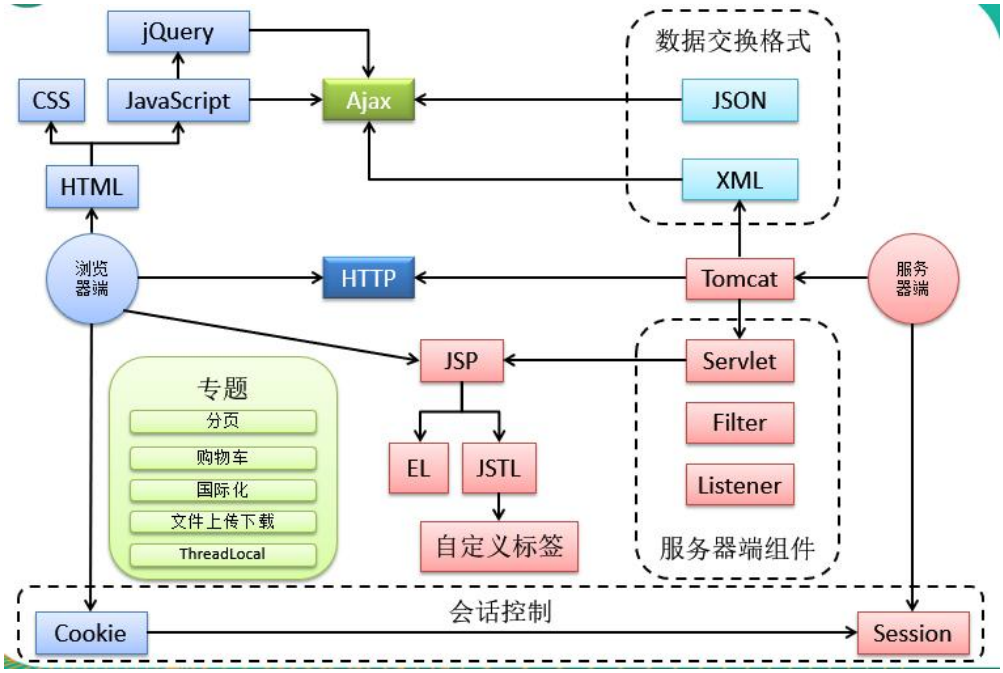
尚硅谷 java 研究院

版本：V 1.0

第 1 章 简介

整个 javaWeb 阶段的内容通过实际的案例贯穿学习， 所涉及到的技术知识点会在案例中根据不同的需求引入。该阶段的学习目标是了解 javaWEB 的整个技术体系，掌握常用的技术知识点。

第 2 章 JavaWeb 的技术体系



第 3 章 登录页面的开发

3.1 涉及的技术知识点

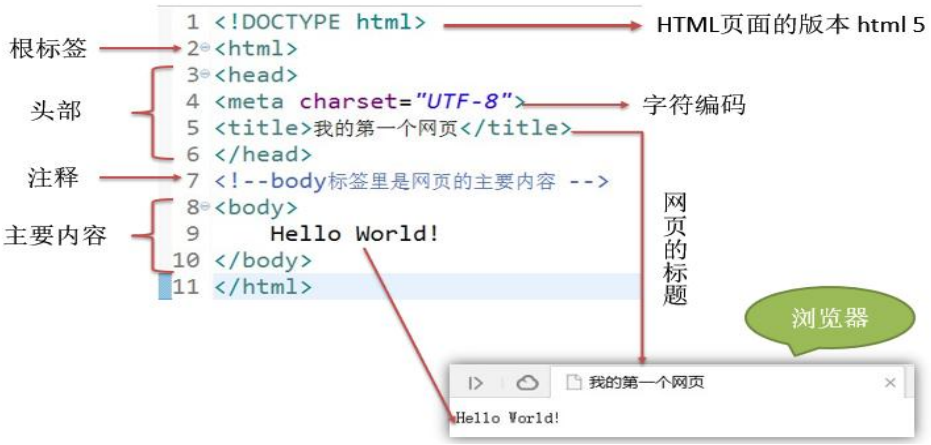
1) HTML

3.2 HTML 是什么?

1) 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。
2) HTML 网页的组成:

3) 常用的 HTML 标签
   ① html 根标记
   ② head 头标记
   ③ body 体标记
   ④ h 标题标签
   ⑤ a 超链接
   ⑥ table 表格
   ⑦ **form** **表单**

4) 一个基本结构的 HTML 页面



5) 登录页面的示例

第 4 章 登录功能实现-环境的搭建

4.1 涉及的技术知识点

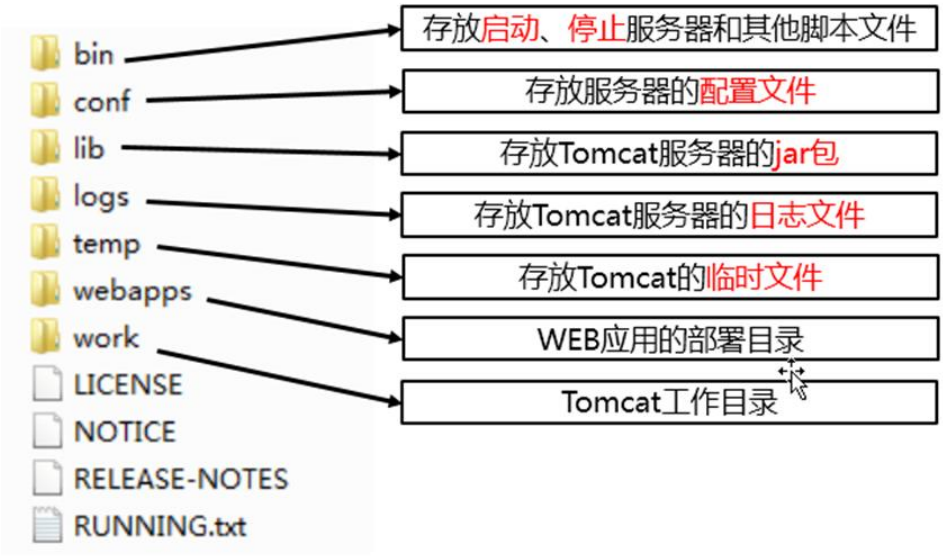
1) WEB 服务器
2) 动态的 web 工程

4.2 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

4.3 Tomcat 服务器的安装及配置

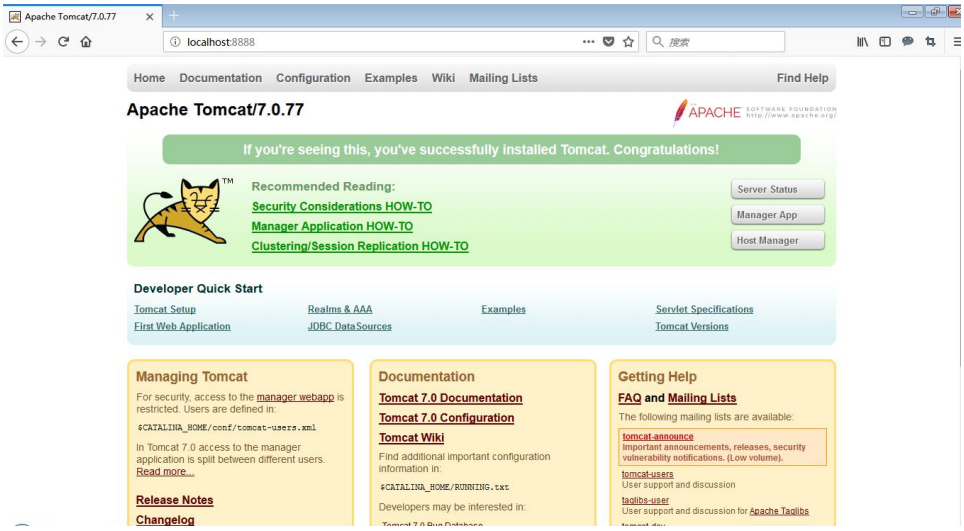
1) 将 Tomcat 的安装包解压到磁盘的任意位(非中文无空格)
2) Tomcat 服务的目录结构



3) 配置环境变量，方便 Tomcat 的启动关闭（可选）
   ① 新建环境变量 CATALINA_HOME=解压目录

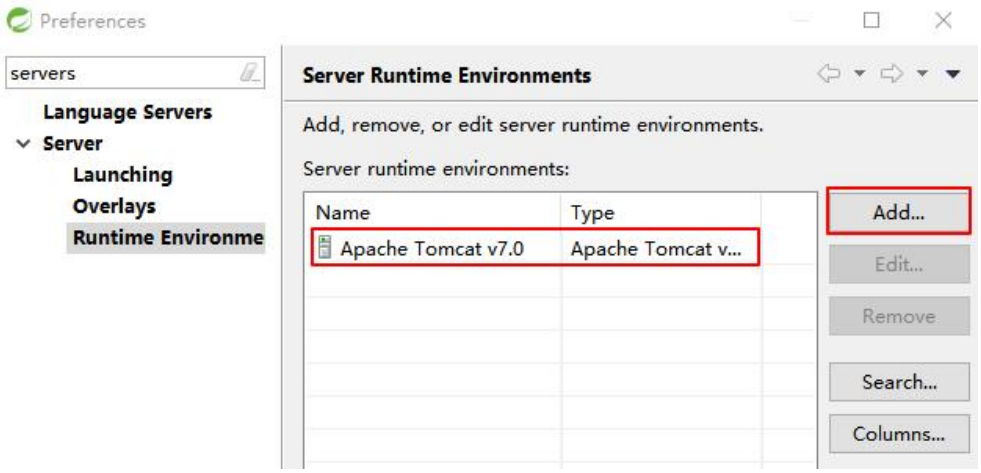
   ② 在 Path 环境变量中加入 Tomcat 解压目录\bin 目录

   ③ 在命令行中运行 catalina run 或者 startup 启动 Tomcat 服务器，在浏览器地址栏访问如下地址进行测试
   http://localhost:8080

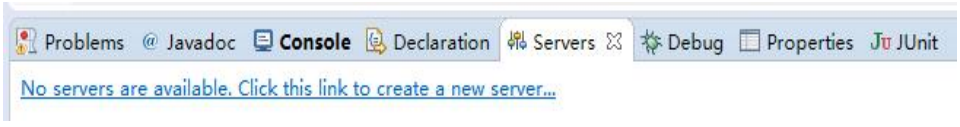


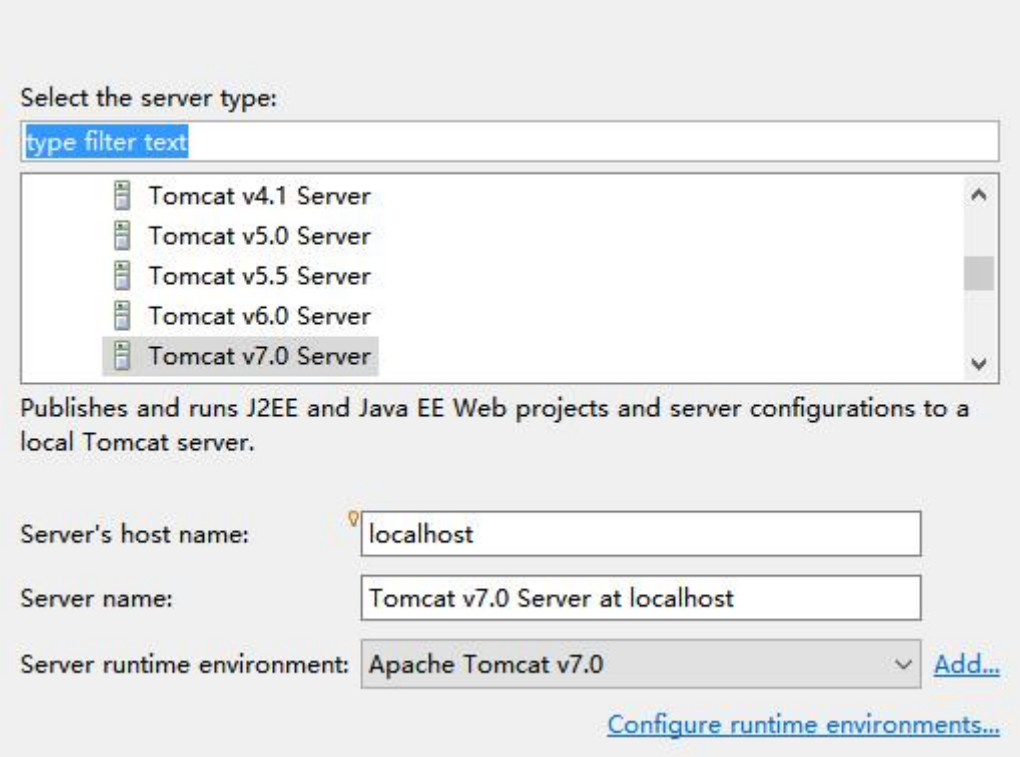
4.4 在 eclipse 中配置 tomcat

1) 在 Eclipse 中配置运行环境



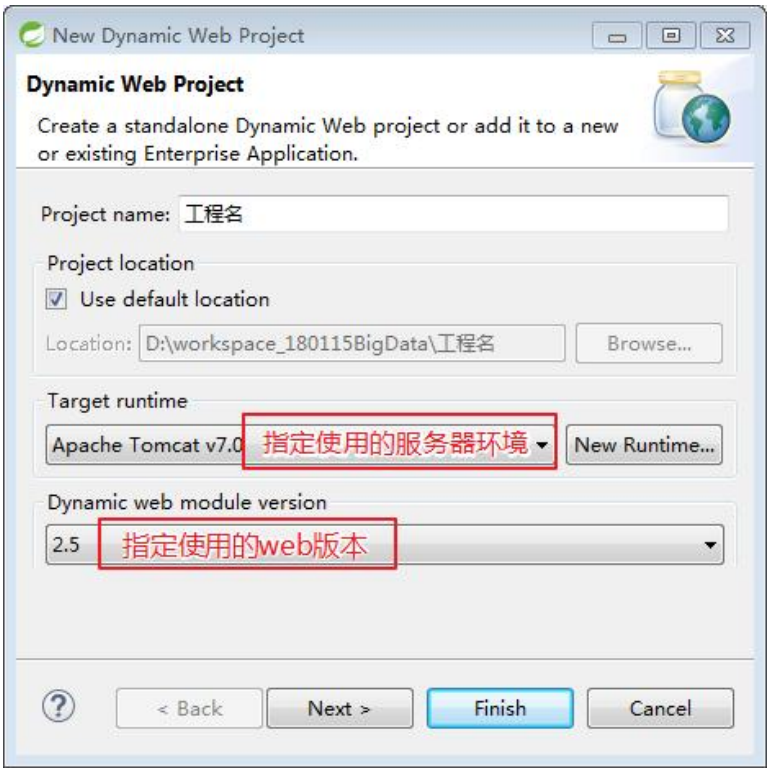
2) 在 Eclipse 中创建新的 Server

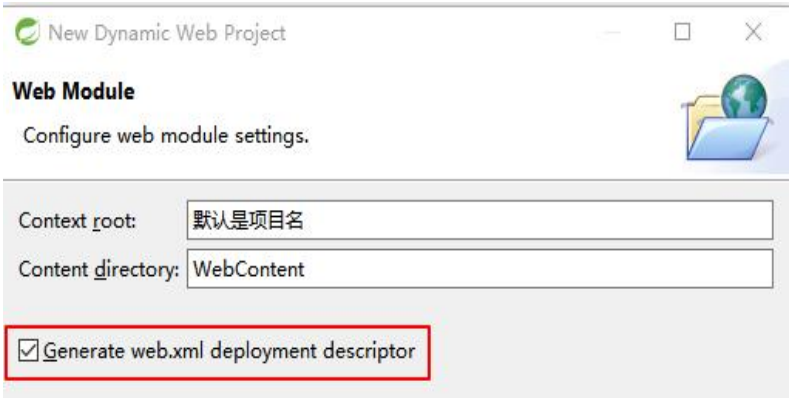




4.5 创建动态的 web 工程

1) 在 Eclipse 中点击 File ->New->Dynamic Web Project





第 5 章 登录功能实现-LoginServlet

5.1 涉及的技术知识点

1) Servlet
2) Request 请求对象
3) Response 响应对象
4) 转发
5) 重定向

5.2 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的.**

5.3 Servlet 开发规则

1) 实际编码通过继承 HttpServlet 来完成 Servlet 的开发

```
public class LoginServlet extends HttpServlet{

}
```

5.4 Servlet 类的相关方法:

2) doGet Servlet 中用于处理 get 请求的方法

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    super.doGet(req, resp);
}
```

3) doPost Servlet 中用于处理 post 请求的方法

```
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    // TODO Auto-generated method stub
    super.doPost(req, resp);
}
```

4) service

① 在 Servlet 的顶层实现中，在 service 方法中调用的具体的 doGet 或者是 doPost

② 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。

5.5 Servlet 在 web.xml 中的配置

```
<servlet>
  <servlet-name>loginServlet</servlet-name>
  <servlet-class>com.atguigu.ajax.servlet.LoginServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>loginServlet</servlet-name>
  <url-pattern>/loginServlet</url-pattern>
</servlet-mapping>
```

5.6 获取请求参数值

1) HttpServletRequest

① 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。

② HttpServletRequest 对象的主要功能有

获取请求参数

获取项目虚拟路径

将请求转发给另外一个 URL 地址 [转发]

5.7 响应结果

1) HttpServletResponse
   ① 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
   ② 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]

5.8 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

```
//请求转发
RequestDispatcher  rd = request.getRequestDispatcher("转发的地址");
rd.forward(request, response);
```

5.9 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

```
//请求重定向
response.sendRedirect("重定向的地址");
```

5.10 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

第 6 章 登录功能实现-页面中错误提示

6.1 涉及的技术知识点

1) Jsp 页面
2) EL 表达式
3) JS 简单应用

6.5 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:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>

</body>
</html>
```

6) JSP 的脚本元素
   ① 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

```
<%
    System.out.println("Hello World");
%>
```

7) JSP 的表达式

① 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

```
<%
    String str = "abc";
%>
<%=str %>
```

8) JSP 的隐含对象

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

9) EL 表达式

① 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。
② **EL 表达式用于代替 JSP 表达式(<%= %>)在页面中做输出操作。**
③ EL 表达式仅仅用来读取数据，而不能对数据进行修改。
④ 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。**
⑤ **EL 表达式的语法:**

```
${ EL表达式(可完成取值、简单的判断、简单的运算等) }
```

⑥ EL 取值的四个域:

pageScope
requestScope
sessionScope
applicationScope

6.6 页面中错误提示的功能效果

6.7 JavaScript

1) 在 1995 年时，由 Netscape 公司的 Brendan Eich，在网景导航者浏览器上首次设计实现而成。Netscape 在最初将其脚本语言命名为 LiveScript，因为 Netscape 与 Sun 合作，网景公司管理层希望它外观看起来像 Java，因此取名为 JavaScript。

2) 特性

① 脚本语言。JavaScript 是一种解释型的脚本语言，C、C++、Java 等语言先编译后执行，而 JavaScript 是在程序的运行过程中逐行进行解释。

② 基于对象。JavaScript 是一种基于对象的脚本语言，它不仅可以创建对象，也能使用现有的对象。

③ 简单。JavaScript 语言中采用的是弱类型的变量类型，对使用的数据类型未做出严格的要求，是基于 Java 基本语句和控制的脚本语言。

④ 动态性。JavaScript 是一种采用事件驱动的脚本语言，它不需要经过 Web 服务器就可以对用户的输入做出响应。

⑤ 跨平台性。JavaScript 脚本语言不依赖于操作系统，仅需要浏览器的支持。因此一个 JavaScript 脚本在编写后可以带到任意机器上使用，前提是机器上的浏览器支持 JavaScript 脚本语言，目前 JavaScript 已被大多数的浏览器所支持。

3) 编写位置

① 编写到 HTML 中<script>标签中。

```
<script type="text/javascript">
        ↵
</script>↵
```

② 写在外部的.js 文件中。然后通过 script 标签引入。

```
<script type="text/javascript" src="script.js"></script>
```

6) JavaScript 的事件驱动

① 用户事件：用户操作，例如单击、鼠标移入、鼠标移出等

② 系统事件：由系统触发的事件，例如文档加载完成。

③ 常用的事件:

onload
onclick
onblur
onfocus
onmouseover
onmouseout

7) BOM

① Borwser Object Model 浏览器对象模型

② 浏览器对象模型提供了独立于内容的、可以与浏览器窗口进行互动的对象结构。BOM 由多个对象组成，其中代表浏览器窗口的 Window 对象是 BOM 的顶层对象，其他对象都是该对象的子对象

③ 常用的对象(window 的子对象)

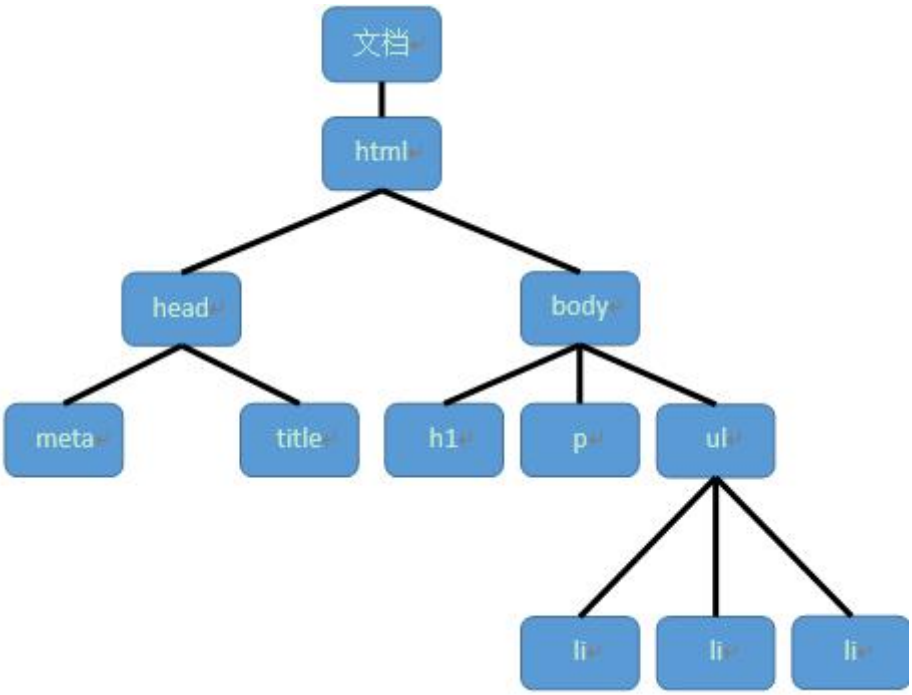
document history location screen navigator frames

8) DOM

① Document Object Model 文档对象模型

② document 对象: window 对象的一个属性，代表当前 HTML 文档，包含了整个文档的树形结构。获 取 document 对象的本质方法是：window.document，而“window.”可以省略。

③ DOM 树



④ 元素查询

功能	API	返回值
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

第 7 章 注册功能实现-异步的表单校验

7.1 涉及的技术知识点

1) Ajax

7.2 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。

7.3 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住不动的。
   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**。
2) 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**。

7.4 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。

7.4.1 Xhr 对象的获取

```
<script type="text/javascript">
    //获取XMLHttpRequest的通用方法
    function getXMLHttpRequest(){
        var xhr;
        try{
            //大部分浏览器都支持
            xhr = new XMLHttpRequest();
        }catch(e){
            try{
                //如果不支持，在这里捕获异常并且采用IE6支持的方式
                xhr = new ActiveXObject("Msxml2.XMLHTTP");
            }catch(e){
                //如果还不支持，在这里捕获异常并采用IE5支持的方式
                xhr = new ActiveXObject("Microsoft.XMLHTTP");
            }
        }
        return xhr;
    }
</script>
```

7.4.2 Xhr 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）

② 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

③ 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

7.4.3 XMLHttpRequest 对象的属性

1) readyState
   ① 描述 XMLHttpRequest 的状态
   ② 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。
2) 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

   ... ...
3) 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
4) responseText
   ① 获得字符串形式的响应数据。
5) responseXML（用的比较少）
   ① 获得 XML 形式的响应数据。
6) 示例代码

```
//发送异步请求 GET
//获取xhr对象
var xhr = getXMLHttpRequest();
//设置请求信息
xhr.open("get","AjaxServlet?&t="+Math.random(),true);
//发送请求
xhr.send();
//监听请求状态
xhr.onreadystatechange = function(){
//当响应完成
    if(xhr.readyState == 4){
        //且状态码为200时
        if(xhr.status == 200){
            //接收响应信息（文本形式）
            var text = xhr.responseText;
            //弹出消息
          alert(text);
        }
    };
};
```

```
//发送异步请求 POST
//获取xhr对象
var xhr = getXMLHttpRequest();
//设置请求信息
xhr.open("post","2.jsp",true);
//设置请求头
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
//发送请求
xhr.send("hello=123456");
//监听请求状态
xhr.onreadystatechange = function(){
//当响应完成
if(xhr.readyState == 4){
    //且状态码为200时
    if(xhr.status == 200){
        //接收响应信息（文本形式）
        var text = xhr.responseText;
        //弹出消息
        alert(text);
    }
};
```

7.4.4 使用 jQuery 框架来发送异步请求

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

2) 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)

3) $.get 方法

jQuery.get(url, *[data], [callback], [type]*) 返回值:XMLHttpRequest

概述

通过远程 HTTP GET 请求载入信息。

这是一个简单的 GET 请求功能以取代复杂 $.ajax 。请求成功时可调用回调函数。如果需要在出错时执行函数，请使用 $.ajax。

参数

url,[data],[callback],[type]	String,Map,Function,String	*V1.0*

url:待载入页面的URL地址
data:待发送 Key/value 参数。
callback:载入成功时回调函数。
type:返回内容格式，xml, html, script, json, text, _default。

4) $.post 方法

jQuery.post(url, *[data], [callback], [type]*) 返回值:XMLHttpRequest

概述

通过远程 HTTP POST 请求载入信息。

这是一个简单的 POST 请求功能以取代复杂 $.ajax 。请求成功时可调用回调函数。如果需要在出错时执行函数，请使用 $.ajax。

参数

url,[data],[callback],[type]	String,Map,Function,String	*V1.0*

url:发送请求地址。
data:待发送 Key/value 参数。
callback:发送成功时回调函数。
type:返回内容格式，xml, html, script, json, text, _default。

5) $.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。最简单的情况下，$.ajax()可以不带任何参数直接使用。

$.ajax 方法的参数

对于 settings 请求设置来说，所有选项都是可选的，详见 jQuery 手册

6） 具体的示例代码

```
//对用户名称进行异步的校验.
$(function(){
    $("#username").blur(function(){
        var value = $(this).val();
        alert(value);
        $.ajax({
            url:"checkUsername",
            type:"POST",
            data:"username="+value,
            success:function(data){
                $("#username_msg").html(data);
            }
        });
    });
});
```

第 8 章 登录功能实现-登录成功跳转主页面

8.1 涉及的技术知识点

1） Session 会话 Cookie
2） JSTL 标签

8.2 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

4) 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
   ④ Cookie 的值只能是 String 类型，不能保存对象

5) Cookie 的具体使用
   ① 创建 cookie

```
Cookie  myCookie = new Cookie("username","ZhangSanFeng");
myCookie.setPath("设置cookie的路径");
myCookie.setMaxAge("设置cookie的时间");
response.addCookie(myCookie);
```

   ② 读取 cookie

```
Cookie [] cookies = request.getCookies();
```

8.3 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

```
HttpSession session = request.getSession();
```

5) 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.

```
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
```

8.4 具体功能展示

ID	LastName	Email	Gender	DeptName
1001	Tom	tom@sina.com	男	开发部
1002	Jerry	jerry@sina.com	女	测试部
1003	Jack	jack@sina.com	男	运维部
1004	Rose	rose@sina.com	女	测试部

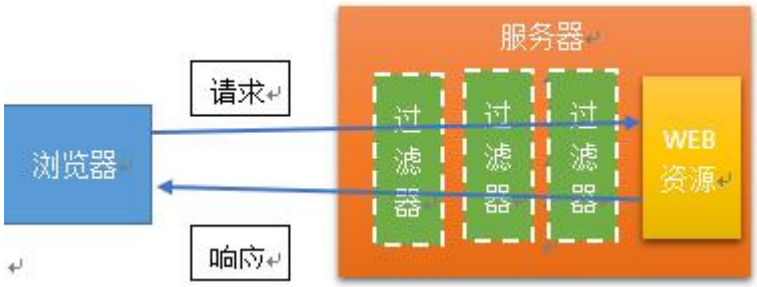
第 9 章主页面访问权限控制

9.1 涉及的技术知识点

1) 过滤器

9.2 过滤器

1) 对于 WEB 应用来说，过滤器是一个驻留在服务器中的 WEB 组件，他可以截取客户端和 WEB 资源之间的请求和响应信息。WEB 资源可能包括 Servlet、JSP、HTML 页面等
2) 当服务器收到特定的请求后，会先将请求交给过滤器，程序员可以在过滤器中对请求信息进行读取修改等操作，然后将请求信息再发送给目标资源。目标资源作出响应后，服务器会再次将响应转交给过滤器，在过滤器中同样可以对响应信息做一些操作，然后再将响应发送给浏览器。
3) 也就是说过滤器可以在 WEB 资源收到请求之前，浏览器收到响应之前，对请求和响应信息做一些相应的操作。
4) 在一个 WEB 应用中可以部署多个过滤器，多个过滤器就组成了一个过滤器链，请求和响应必须在经过多个过滤器后才能到达目标



9.3 过滤器的使用

1) 通过实现 Filter 接口完成过滤器的开发

```
public class LoginFilter  implements Filter{

    @Override
    public void destroy() {
    }
    @Override
    public void doFilter(ServletRequest arg0, ServletResponse arg1, FilterChain arg2)
            throws IOException, ServletException {
    }
    @Override
    public void init(FilterConfig arg0) throws ServletException {

    }
}
```

2) Filter 在 web.xml 中的配置

```
<filter>
  <filter-name>loginFilter</filter-name>
  <filter-class>com.atguigu.ajax.servlet.LoginFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

9.4 主页面访问权限控制要求

1) 在进入主页面必须进行登录状态的判断，如果未登录状态不允许进入主界面。
2) 登录状态的判断再过滤器中实现，更为通用，而且可拔插。

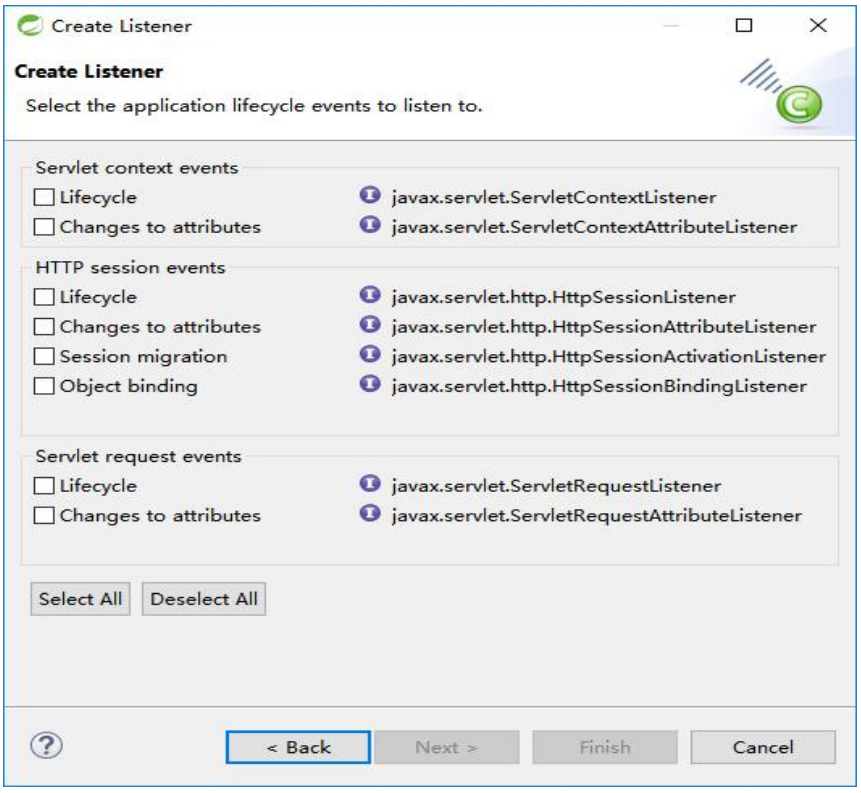
第 10 章 在线人数统计

10.1 涉及的技术知识点

1) 监听器

10.2 监听器

1) Listener 用于监听 JavaWeb 程序中的事件。
2) 例如：ServletContext、HttpSession、ServletRequest 的创建、修改和删除。
3) 监听器的类型分为
   ① 生命周期
   ② 数据绑定



10.3 在线人数统计功能展示

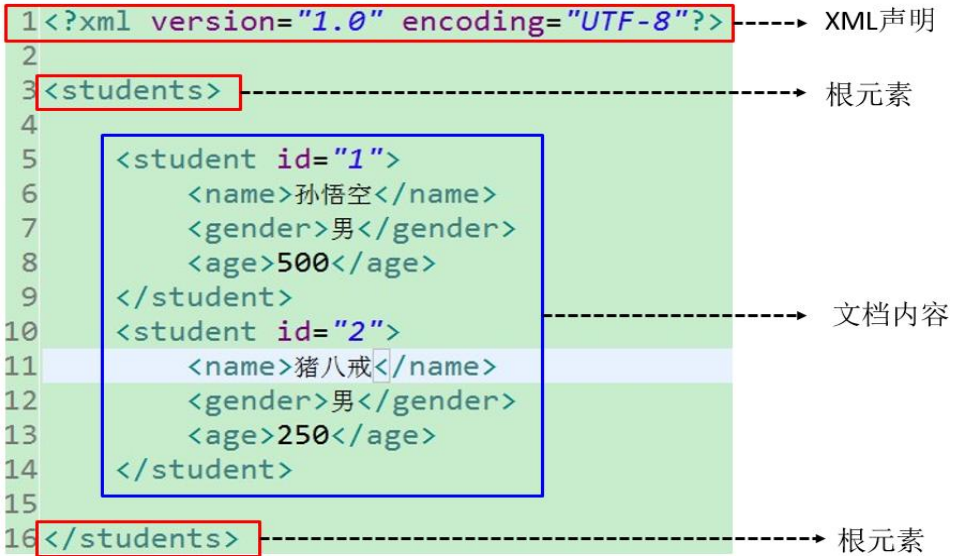
第 11 章 xml

11.1 xml 简介

1) XML--可扩展标记语言 eXtensible Markup Language
2) 由 W3C 组织发布，目前推荐遵守的是 W3C 组织于 2000 年发布的 XML1.0 规范
3) XML 的使命，就是以一个统一的格式，组织有关系的数据，为不同平台下的应用程序服务
4) XML 用来传输和存储数据，HTML 用来显示数据
5) XML 没有预定义标签，均为自定义标签

11.2 xml 用途

1) 配置文件
   JavaWeb 中的 web.xml
   C3P0 中的 c3p0-config.xml
2) 数据交换格式
   Ajax
   WebService
3) 数据存储
   保存关系型数据
4)



11.3 xml 基本语法

1) XML 文档组成
   ① XML 声明
   version 属性指定 XML 版本，固定值是 1.0
   encoding 指定的字符集，是告诉解析器使用什么字符集进行解码，而编码是由文本编辑器决定的
   ② CDATA 区
   当 XML 文档中需要写一些程序代码、SQL 语句或其他不希望 XML 解析器进行解析的内容时，就可以写在 CDATA 区中
   XML 解析器会将 CDATA 区中的内容原封不动的输出
   CDATA 区的定义格式：<![CDATA[…]]>
2) 语法规则
   ① XML 声明要么不写，要写就写在第一行，并且前面没有任何其他字符
   ② 只能有一个根标签
   ③ 标签必须正确结束
   ④ 标签不能交叉嵌
   ⑤ 严格区分大小写
   ⑥ 属性必须有值，且必须加引号
   ⑦ 标签不能以数字开头
   ⑧ 注释不能嵌套

11.4 xml 解析

1) XML 解析是指通过解析器读取 XML 文档，解释语法，并将文档转化成对象
2) 常用的解析方式

DOM（Document Object Model）

SAX（Simple API for XML）

3) DOM 和 SAX 解析的对比

	DOM	SAX
速度	需要一次性加载整个XML文档，然后将文档转换为DOM树，速度较差	顺序解析XML文档，无需将整个XML都加载到内存中，速度快
重复访问	将XML转换为DOM树之后，在解析时，DOM树将常驻内存，可以重复访问。	顺序解析XML文档，已解析过的数据，如果没有保存，将不能获得，除非重新解析。
内存要求	内存占用较大	内存占用率低
修改	既可读取文档内容，又可以修改	只能读取，不能修改
复杂度	完全面向对象的解析方式，容易使用	采用事件回调机制，通过事件的回调函数来解析XML文档，略复杂。

4) Dom4j 解析示例

解析

```
//1.创建解析器对象
SAXReader saxReader = new SAXReader();
//2.解析 xml 文件获取 document 对象
Document document = saxReader.read("students.xml");
//3.得到根元素
Element root = document.getRootElement();
```

第 12 章 JSON

12.1 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：
   ① {“name”:”sunwukong” , ”age”:18 , ”address”:”beijing” }
   ② 这个对象中有三个属性 name、age 和 address
   ③ 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
   ④ ‘{“name”:”sunwukong” , ”age”:18 , ”address”:”beijing” }’
   ⑤ 变成字符串后有一个好处，就是可以在不同语言之间传递。
   ⑥ 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。

12.2 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]

12.3 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)

12.4 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);
```

6) 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));
```

12.5 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();
}
```

2) 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>
```