* 学习目标

 * 能够掌握Session技术

    * 会话：服务端技术

    * 持久化两种：Cookies,重写URL

    * id， 创建时间，最近访问时间,....

    * login

 * 能够掌握服务端作用域（Request,Session.ServletContext）的区别

 * 能够掌握JSP的概念

    * Java Server Pages

    * JSP=html+Java

 * 能够理解JSP的本质是Servlet

 * 能够理解JSP的执行流程

    * 第一次：浏览器--hello.jsp--hello.java--hello.class(Servlet)

   * 第二次： 浏览器--hello.class(Servlet)

 * 能够掌握JSP常见的语法

    * JSP模板元素（html）

    * JSP表达式:<%=new Date()%> out.print

    * JSP脚本 <%...%>

    * JSP声明：定义成员变量，静态代码块，方法，...

    * JSP注释:<!-- --> <%!-- --%>

    * JSP 指令：page，include,taglib

    * JSP 标签：自定义标签

 * 能够掌握JSP的九大内置对象

    * request,response,session,application,config,page,pageContext,exception,out

---

 * 回顾

    * Response

* 获得输出流

* 设置响应编码

reps.setContentType("text/html;charset=utf-8");

* 下载

* resp.setHeader("content-disposition", "attachment;filename="+ URLEncoder.encode("美女.jpg","UTF-8"));

* 验证码码例子：BufferedImage，ValidateCode

* 生成二维码作业

* 解决中文乱码问题

* response.setContentType("text/htm;charset=utf-8");

* request.setCharacterEncoding("UTF-8");

* 会话技术

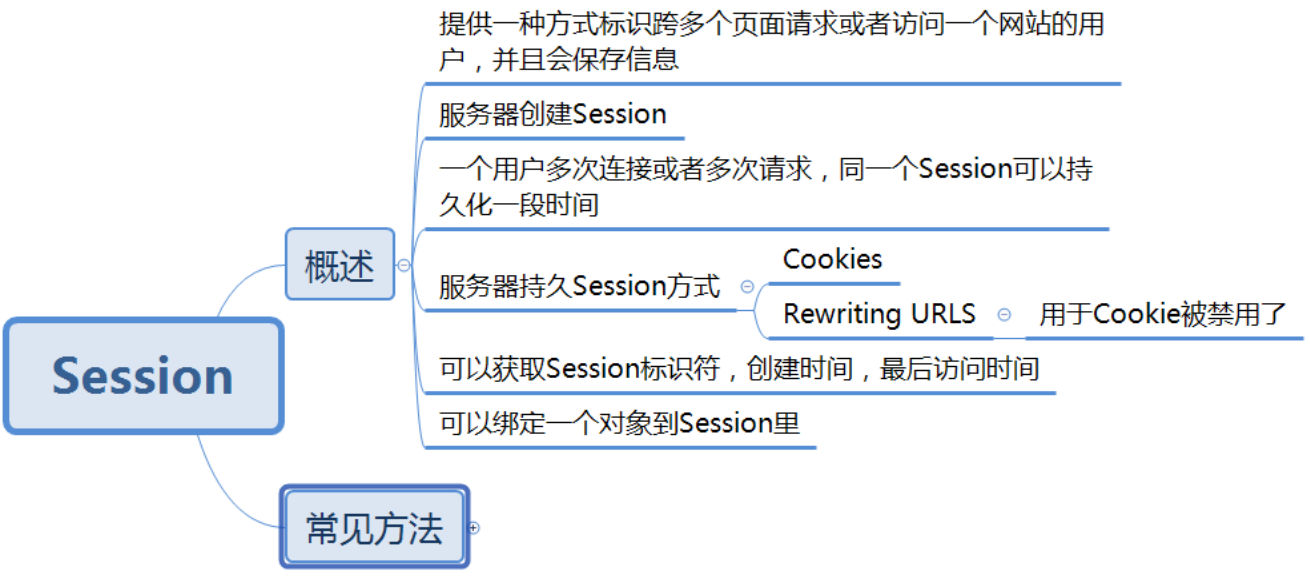
* 客户端技术：Cookie

* 上次访问的时间

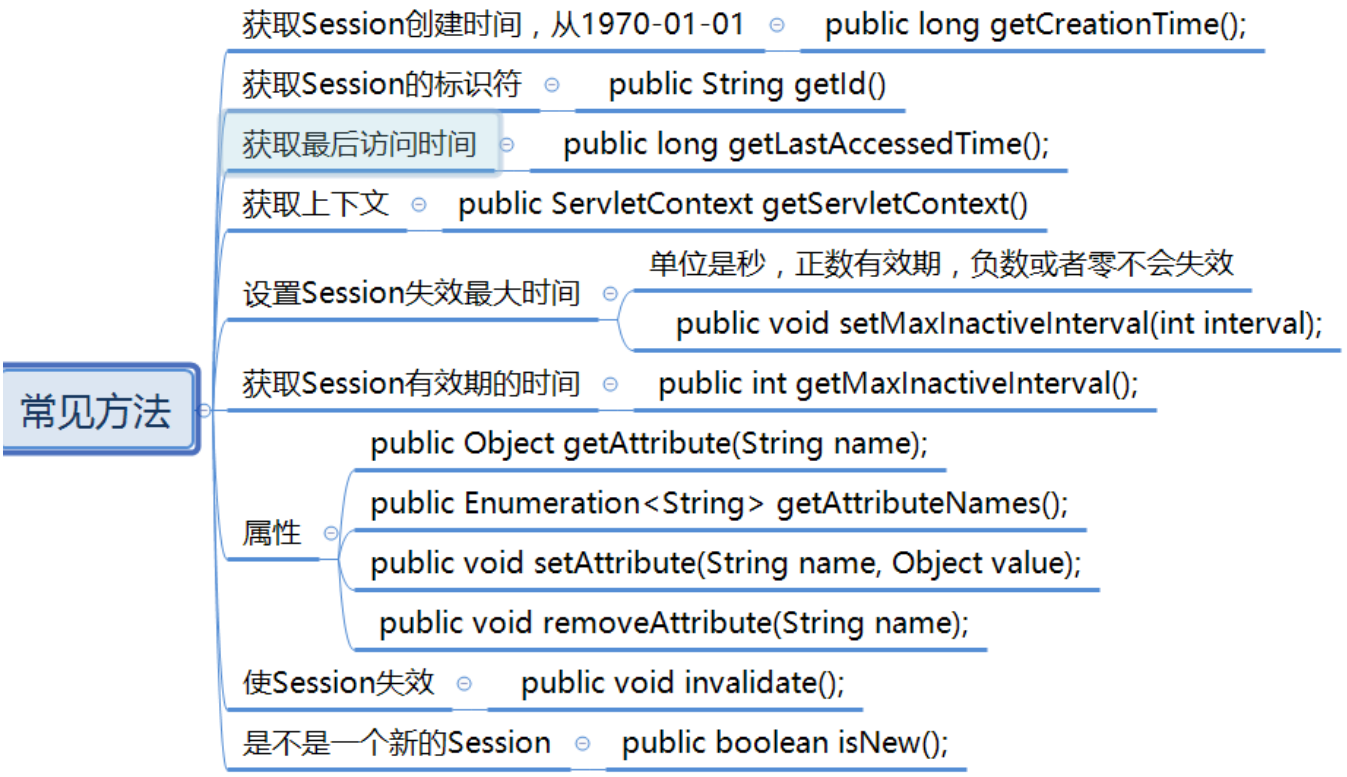
* 登录了：记住我

* 能够掌握Session技术

* Session的概述



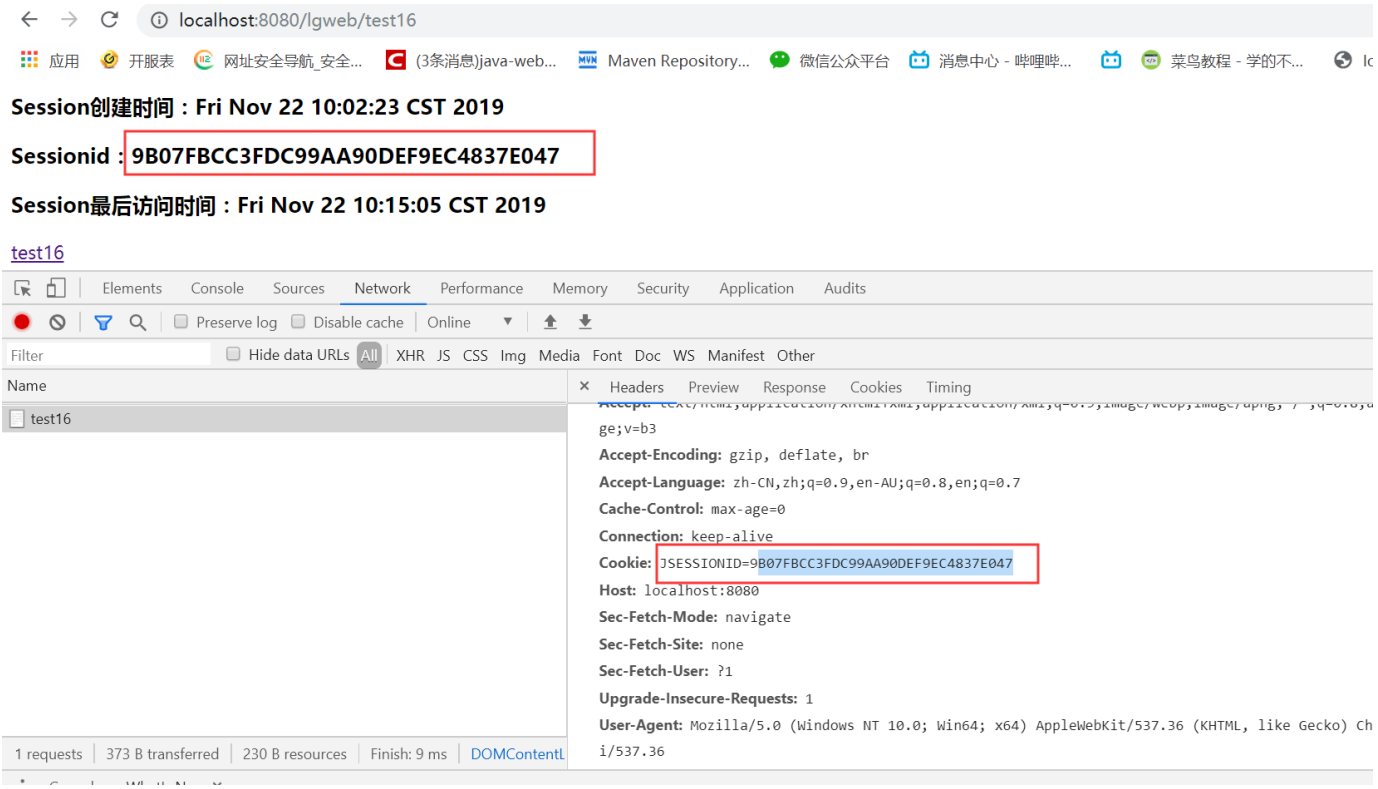
* 操作Session常见的方法




```
* 案例
@WebServlet("/test16")
public class TestServlet16 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
        resp.setContentType("text/html;charset=utf-8");
        req.setCharacterEncoding("UTF-8");
        // 怎么样获得一个Session
        HttpSession session = req.getSession();
        long creationTime = session.getCreationTime();
        String id = session.getId();
        long lastAccessedTime = session.getLastAccessedTime();
        resp.getWriter().write("<h3>Session创建时间：  "+new Date(creationTime)+"<
        resp.getWriter().write("<h3>Sessionid：  "+id+"</h3>");
        resp.getWriter().write("<h3>Session最后访问时间：  "+new Date(lastAccessed
        // 假如禁用了Cookie，可以使用重写URL的形式
        String path = resp.encodeRedirectURL("http://localhost:8080/lgweb/test1
        resp.getWriter().write("<a href="+path+">test16</a>");

```

```
20      }
21
22      @Override
23      protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
24          doGet(req, resp);
25      }
26  }
27
```




```
1   * Session登录的案例
2   @WebServlet(value = "/login2")
3   public class LoginServlet2 extends HttpServlet {
4       @Override
5       protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
6           resp.setContentType("text/html;charset=utf-8");
7           req.setCharacterEncoding("UTF-8");
8           // 1 获取用户从页面输入的数据：例如表单提交的数据
9           String username = req.getParameter("username");
10          String password = req.getParameter("password");
```

```
11
12          // 2 调用登录业务--调用dao--查询数据库
13          LoginService loginService=new LoginServiceImpl();
14          boolean isOk = loginService.login(username, password);
15          if(isOk){
16              req.getSession().setAttribute("username",username);
17              req.setAttribute("msg","login success");
18              getServletContext().getRequestDispatcher("/success2").forward(req,r
19          }else{
20              req.setAttribute("msg","login fail");
21              getServletContext().getRequestDispatcher("/fail2").forward(req,resp
22          }
23      }
24
25      @Override
26      protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
27          doGet(req,resp);
28      }
29  }
30  @WebServlet(value = "/success2")
31  public class SuccessServlet2 extends HttpServlet {
32      @Override
33      protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
34          // 获得信息，响应给页面
35          String msg = (String) req.getAttribute("msg");
36          resp.getWriter().write(msg);
37      }
38
39      @Override
40      protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
41         doGet(req, resp);
42      }
43  }
44
45  @WebServlet(value = "/fail2")
46  public class FailServlet2 extends HttpServlet {
47      @Override
48      protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
49          // 获得信息，响应给页面
50          String msg = (String) req.getAttribute("msg");
```

```
51          resp.getWriter().write(msg);
52      }
53
54      @Override
55      protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
56          doGet(req, resp);
57      }
58  }
59
60  @WebServlet("/test17")
61  public class TestServlet17 extends HttpServlet {
62      @Override
63      protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
64          resp.setContentType("text/html;charset=utf-8");
65          req.setCharacterEncoding("UTF-8");
66          Object username = req.getSession().getAttribute("username");
67          if(username==null){
68              resp.getWriter().write("<h3>请登录</h3>");
69          }else{
70              resp.getWriter().write("<h3>"+username+"</h3>");
71          }
72      }
73
74      @Override
75      protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
76          doGet(req,resp);
77      }
78  }
79
```

* 能够掌握服务端作用域（Request,Session.ServletContext）的区别

Request(生命周期：一次请求)

*** 请求转发数据**

Session（生命周期：多次请求，在会话中）

*** 一次会话：登录，会员信息，验证码，购车网车数据，...**

ServletContext（生命周期：多次请求，整个Web应用）

*** 全局--->统计整个网站访问量**

* 能够掌握JSP的概念

* 概述

* JSP全称是Java Server Pages，它和servle技术一样，都是SUN公司定义的一种用于开发动态web资源的技术。

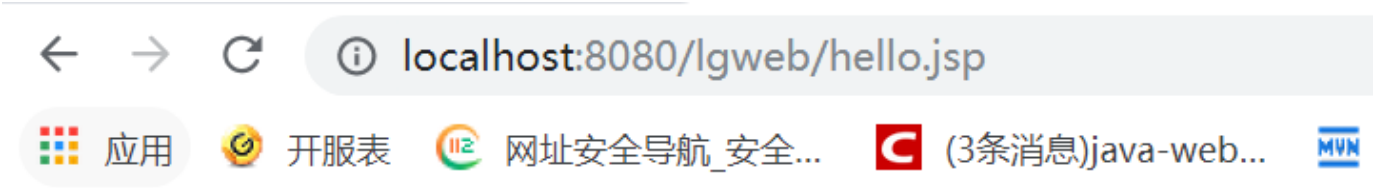
* JSP这门技术的最大的特点在于，写jsp就像在写html，但它相比html而言，html只能为用户提供静态数据，而Jsp技术允许在页面中嵌套java代码，为用户提供动态数据。

* JSP=html+Java

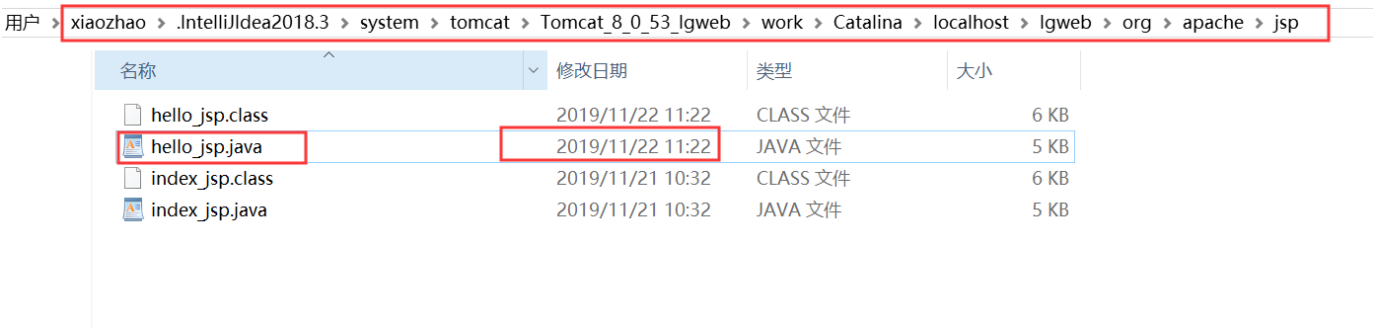
* 能够理解JSP的本质是Servlet

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Hello JSP</title>
</head>
<body>
    <h1>Hello 亮哥教育</h1>
```

```
</body>
</html>
```



Hello 亮哥教育




```
C:\Users\xiaozhao\.IntelliJIdea2018.3\system\tomcat\Tomcat_8_0_53_lgweb\work
\Catalina\localhost\lgweb\org\apache\jsp
```

```
public final class hello_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent,
               org.apache.jasper.runtime.JspSourceImports {
```

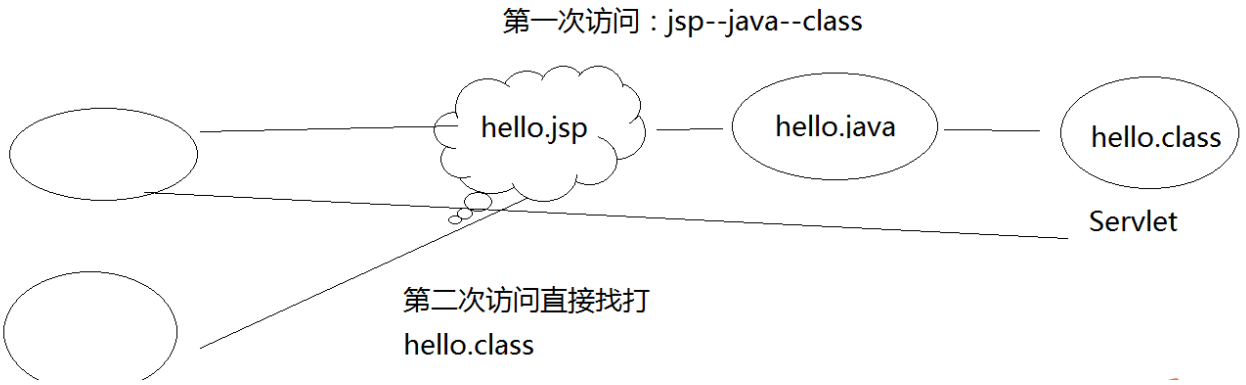
```
public abstract class HttpJspBase extends HttpServlet
```

```
out.write("\r\n");
out.write("\r\n");
out.write("<html>\r\n");
out.write("<head>\r\n");
out.write("    <title>Hello JSP</title>\r\n");
out.write("</head>\r\n");
out.write("<body>\r\n");
out.write("    <h1>Hello 亮哥教育</h1>\r\n");
out.write("</body>\r\n");
out.write("</html>\r\n");
```

* JSP的本质就是Servlet

* 能够理解JSP的执行流程

  * 在Tomcat访问JSP的过程



* 第一次执行：

  a. 客户端通过电脑连接服务器，因为是请求是动态的，所以所有的请求交给WEB容器来处理

  b. 在容器中找到需要执行的*.jsp文件

  c. 之后*.jsp文件通过转换变为*.java文件

  d. *.java文件经过编译后，形成*.class文件

  e. 最终服务器要执行形成的*.class文件

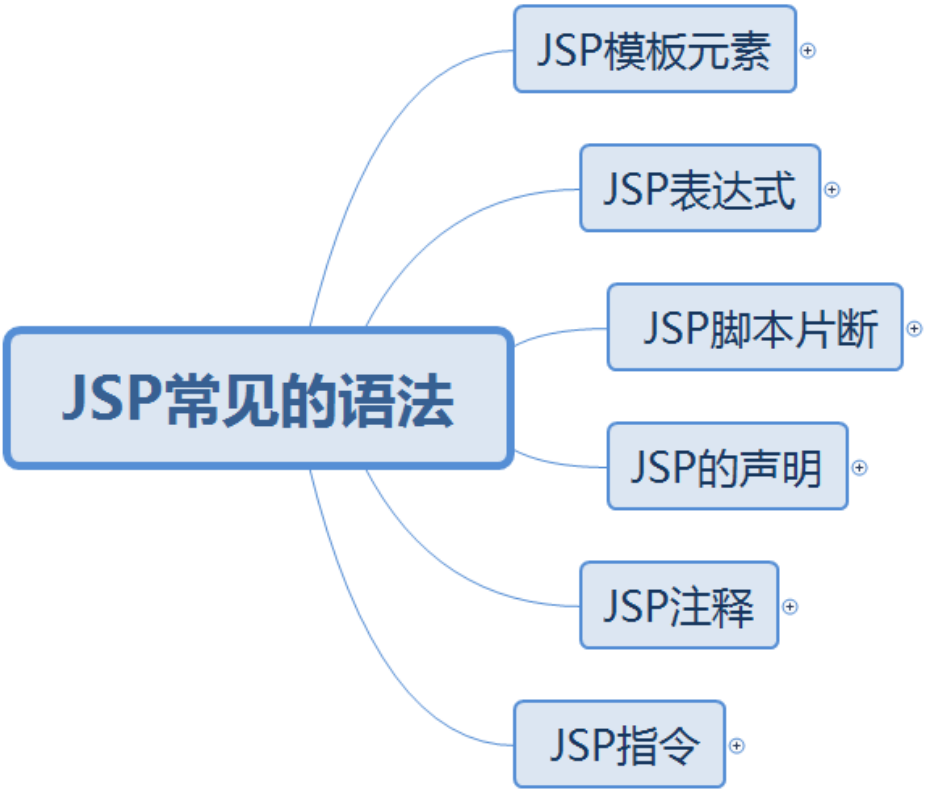
* 第二次执行：

  a. 因为已经存在了*.class文件，所以不在需要转换和编译的过程

* 修改后执行：

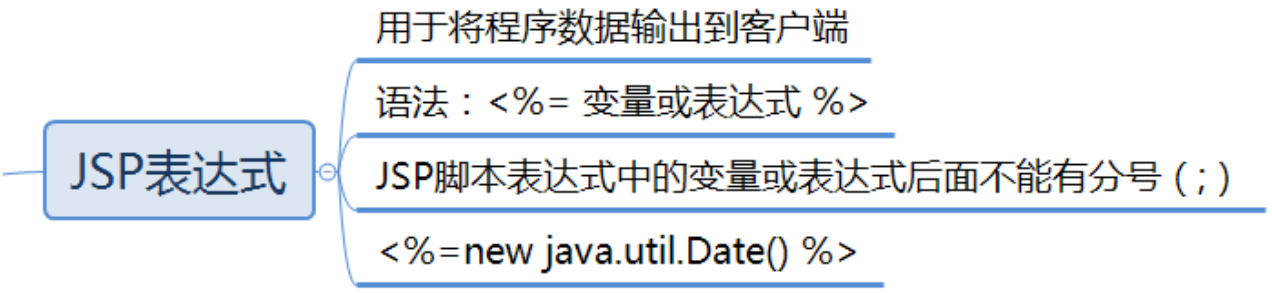
  1.源文件已经被修改过了，所以需要重新转换，重新编译。

* 能够掌握JSP常见的语法

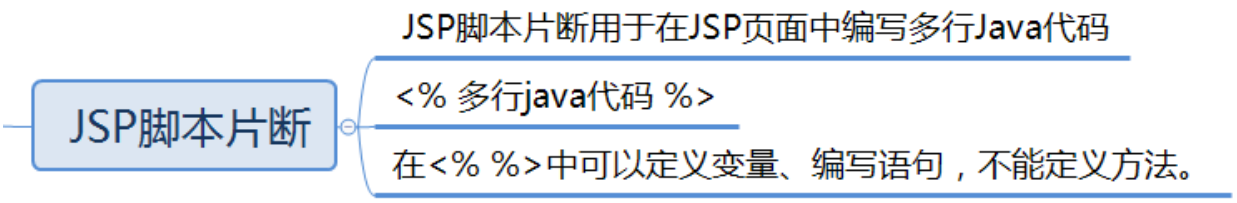


* JSP模板元素

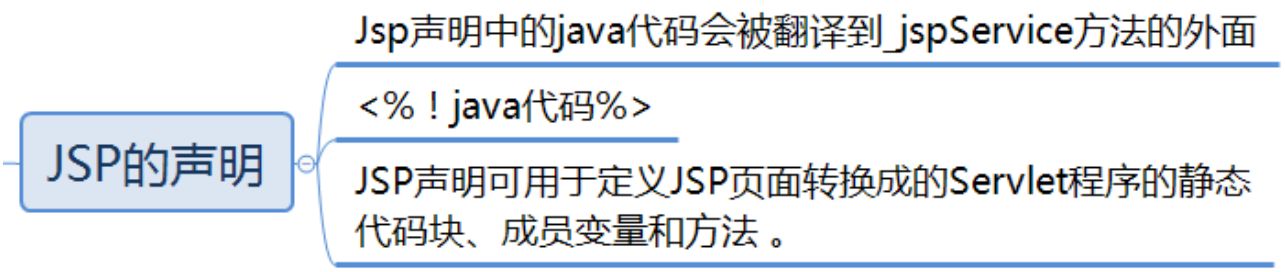
* JSP表达式



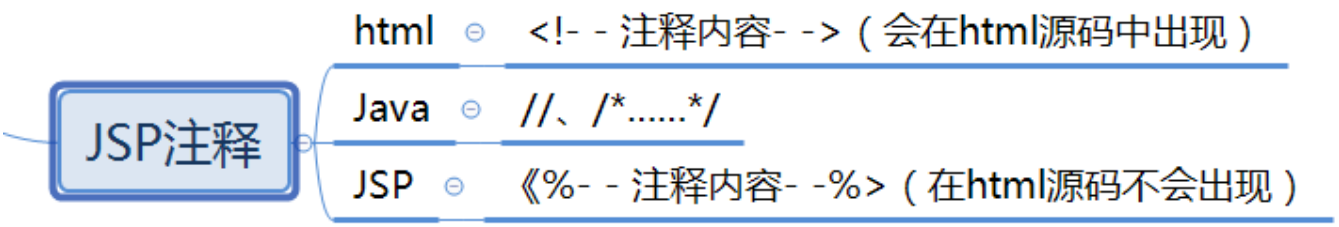
* JSP脚本片断



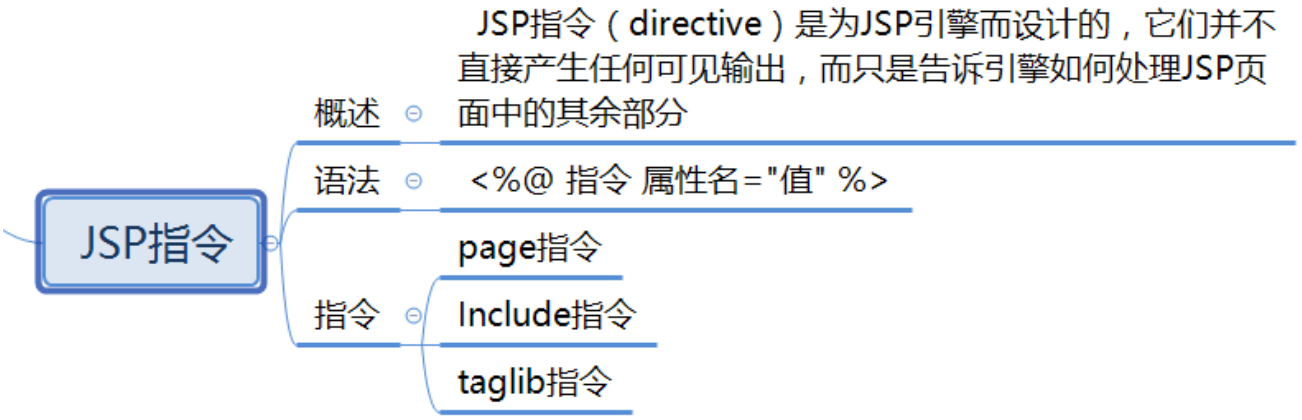
* JSP的声明



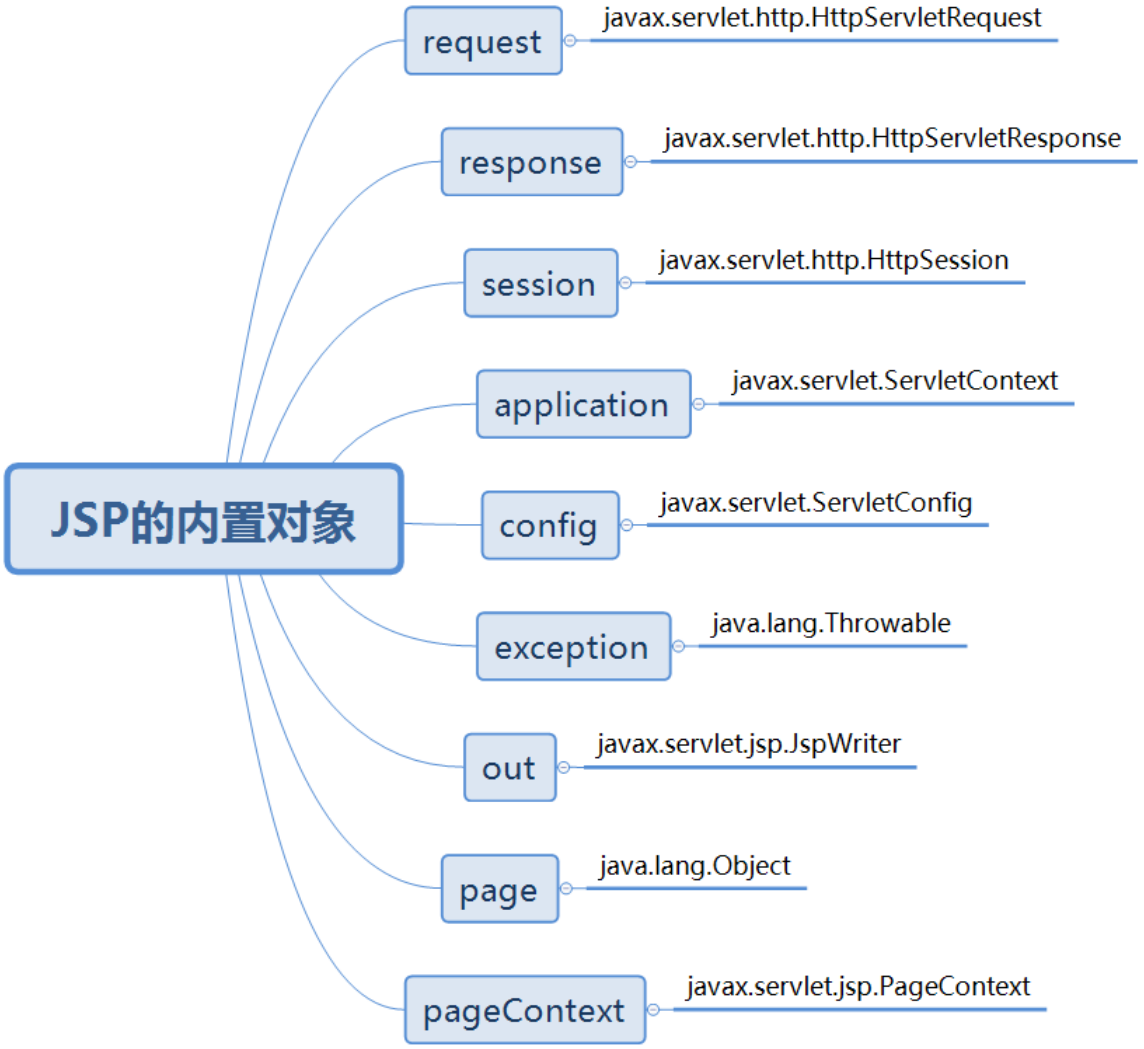
* JSP注释



* JSP指令



* 能够掌握JSP的九大内置对象


JSP的内置对象
request
javax.servlet.http.HttpServletRequest
response
javax.servlet.http.HttpServletResponse
session
javax.servlet.http.HttpSession
application
javax.servlet.ServletContext
config
javax.servlet.ServletConfig
exception
java.lang.Throwable
out
javax.servlet.jsp.JspWriter
page
java.lang.Object
pageContext
javax.servlet.jsp.PageContext