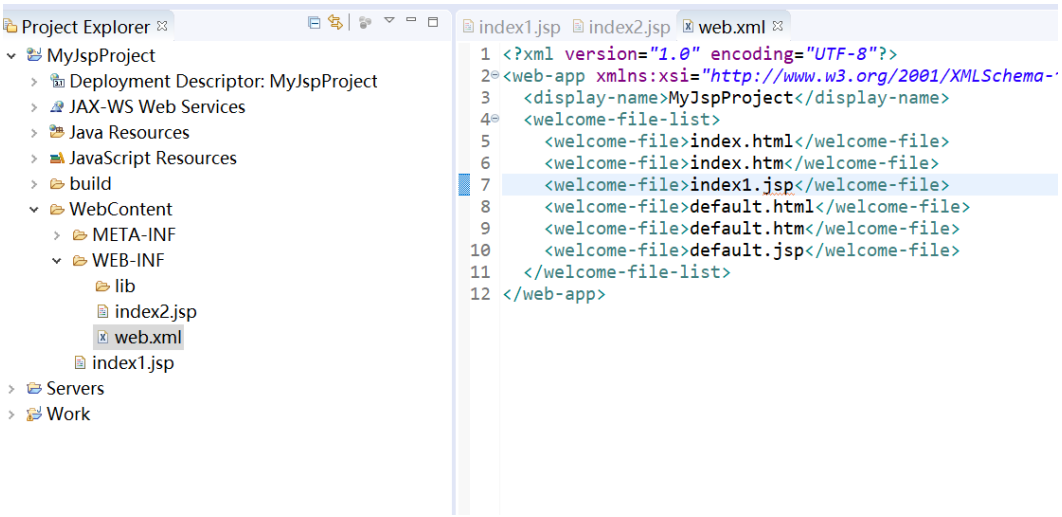
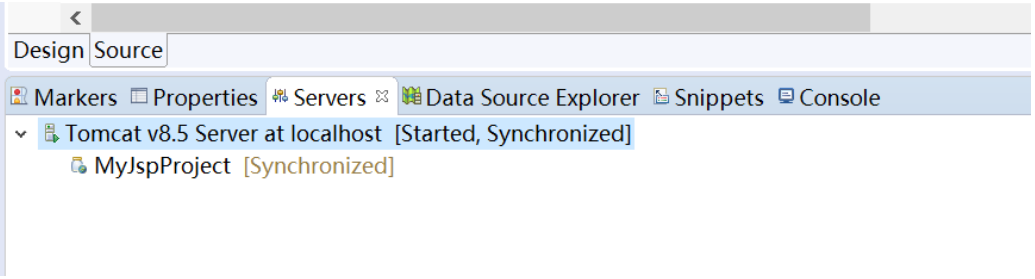
1.访问WebContent的子文件



默认的文件访问在web.xml中，因为这里面没有index1.jsp所以访问http://localhost:8888/MyJspProject/的时候显示的而是查找不到文件，正在里面修改一下，修改之后要重启下图



就可以直接访问index1.jsp文件了，或者直接写死http://localhost:8888/MyJspProject/index1.jsp这样也可以访问这个文件，但前提是Index1.jsp是在WebContent这个文件夹的子文件，

http://localhost:8888/MyJspProject/相当于文件夹WebContent的虚拟路径

2.访问WEB-INF里面的文件

EClipse中的web项目：浏览器可以直接访问Webcontent中的文件，

但是web-inf中的文件不能通过客户端（浏览器）直接访问，只能通过请求转发来访问，因为权限较高

注意：并不是任何的内部跳转都能访问web-inf ，因为跳转有两种方式：请求转发和重定向（因为考虑安全问题

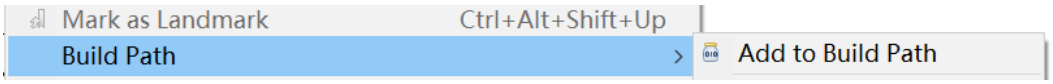
3.配置tomcat运行的环境

jsp<->servlet//如果没有配置就只能运行JSP

a.

直接复制粘贴D:\各种软件的安装包\apache-tomcat-8.5.50\lib\servlet-api.jar

先粘贴到Java Resources中的src然后点击右键，点击下图



b.

右键项目—>Build Path->Add library->Server Runtime(本质将Tomcat中lib目录下的jar包全部加进去)//把这个jre包加入项目的构建路径：项目能够直接识别的代码

4.同意字符集编码

a.编码的分类：

设置jsp文件的编码（jsp文件的pageEncoding属性）：　　jsp ->java

设置浏览器读取jsp文件的编码（jsp文件中的content属性）

一般需要将上述设置成一样，推荐使用国际编码UTF-8

文本编码：

i.将整个eclipse中的文件统一设置，在windows中->proferencers->jsp file->encoding

ii.设置某一个项目中的编码右键项目 properties->text file encoding->other->utf-8

iii.设置单独的文件

4.部署tomcat

tomcat的conf中 的server和eclipse的server文件不一样，所以在eclipse中启动，并且修改的是tomcat的server文件的端口是访问不了的

解决办法：在servers面板新建一个tomcat实例，再在该实例中部署项目（右键-add）之后运行

注意：一般建议，将eclipse中的tomcat与本地的配置信息保持一致：将eclipse中的tomcat设置为托管模式：【第一次】创建tomcat实例之后，双击serverlocaltion中选择第二项

5.jsp的页面元素：HTML Java代码（脚本scriplet）、注释、指令

a.脚本scriplet：

i.<%

JAVA代码（局部变量）

%>

ii.<%!

java代码（定义全局变量或者方法）

%>

iii.<%=输出表达式%>

一般而言，修改web.xml、配置文件、Java 需要重启tomcat服务

但是如果是修改jsp\html\css\js，不需要重启

注意：在这里面用println是不能回车的，必须要自己加一个

在<%=%>和out.print()中可以直接解析html代码

b.指令：<%@ page....%> <%@ include....%> <%@ taglib....%>

page指令：

属性：

language:jsp页面使用的脚本语言

import:导入的类

pageEncoding:jsp文件自身编码 java->jsp

contentType:浏览器解析jsp的编码

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8" import="java.util.Date"%><!alt+/有提示>

c.注释

html注释：<!--.....--> 可以通过浏览器客户端通过右键查看源代码观察到

java注释：//....or/*.....*/

jsp注释：<%--.....--%>

7.JSP9大内置对象

内置对象：指自带的，不用使用new 也能使用的对象

1.out:输出对象，向客户端输出内容

2.request:请求对象，存储”客户端向服务端发送的请求信息“

request对象的常见方法：

String getParameter(String name)//根据请求的字段名key,返回字段值value

String[] getParameterValue(String name)//根据请求的字段名key,返回多个字段值value（checkbox多选按钮值）

void setCharacterEncoding("编码格式 utf-8")：设置请求编码（tomcat7以前默认iso-8859-1,tomcat8及以后改为了utf-8）

getRequestDispatcher(” “).forward(request,response); :请求转发的跳转页面 从页面A跳转到页面B

ServletContext getServerContext():获取项目的ServletContext对象

3.response:响应对象，

提供的方法：

void addCookie(Cookie cookie); 服务端向客户端增加cookie对象

void sendRedirect("a.jsp") throws IOException:页面跳转的一种方式（重定向）

示例：登陆

login.jsp->check.jsp->success.jsp

请求转发和重定向的区别;

	请求转发	重定向
地址栏是否改变	不改变	改变
是否保留第一次	保留	不保留
请求时的数据		
请求的次数•	1	

2

跳转位置	服务器	客

户端发出的第二次

void setContentType(string type)：设置服务端响应的编码（设置服务端的ContentType类型）

4.session:

session(服务端)

Cookie(客户端，不是内置对象)：Cookei是由服务端产生，再发送给客户端保存。相当于【本地缓存】的作用：客户端(hello.mp4 zs.abc)->服务端(hello.mp4 zs/abc)：第二次就可以再本地查看了

作用：提高访问服务端的效率，但安全性较差。

Cookie:name=value

javax.servlet.http.Cookie

public Cookie(String name.String value)

String getName() ：获取name

String getValue()：获取value

void setMAxAge(int expiry):最大有效期（秒）

服务端准备Cookie：

response.addCookie(Cookie cookie)

页面跳转（转发，重对象）

客户端获取Cookie:

request.getCookies();

注意：获取是一次性将所有Cookie全部拿到

通过F12可以发现除了自己设置的Cookie对象外，还有一个name为JSESSION的cookie,cookie在本机

建议cookie只保存英文数字，否则需要进行编码和解码

使用Cookie实现记住用户名的功能。

session:会话

a.浏览网站：开始-关闭

b.购物：浏览、付款、退出，这一整个操作完成之后就是一次会话

c.电子邮件：浏览、写邮件、退出

一次开始到一次结束

session机制:

客户端第一次请求服务端是，（）服务端会产生一个session对象（用于保存该用户的信息）；并且每个session对象都会有一个唯一的sessionId(用于区分其他的session)服务端会产生一个cookie,并且该ccokie的name=JSESSIONID,value=服务端sessionId的值；然后服务端会响应客户端的同时将该cookie发送给客户端，至此客户端就有了一个cookie(JSESSIONID)；因此，客户端的cookie就可以和服务端的session一一对应（JSESSIONID=sessionId）

客户端第n次请求服务端：服务端先用客户端cookie的JSESSIONID取服务端的session中取匹配sessionId如果匹配成功，说明此用户不是第一次访问，无需登陆。

例子：客户端：　　顾客　（客户端）

服务端：存包处　-商场（服务端）

顾客第一次存包：商场判断此人是之前存过包（通过你手里是否有钥匙）

如果是新顾客（没钥匙），分配一个钥匙给顾客；钥匙会和柜子一一对应

如果老顾客（有钥匙），则不需要分配：该顾客手里的钥匙会和柜子一一对应。

session:

a.session存储在客户端

b.session是在同一个用户（客户）请求时共享。

c.实现机制：第一次用户请求时产生一个sessionid并复制给jsessionid然后发送给客户端，最终通过sessionid-jsessionid让客户端和服务端一一对应

session的方法：

```
/**
 * 用于演示Servlet API中的Session管理机制
 */
public class SessionInfoServlet extends HttpServlet {
    /**
     * Builds an HTML document containing session information and returns it to
     * the client.
     */
    public void doGet(HttpServletRequet request, HttpServletResponse response)
            throws ServletException, IOException {
        // get current session or, if necessary, create a new one
        HttpSession mySession = request.getSession(true);

        // MIME type to return is HTML
        response.setContentType("text/html");

        // get a handle to the output stream
        PrintWriter out = response.getWriter();

        // generate HTML document
        out.println("<HTML>");
```

false就是当有session的时候就拿到那个session，如果没有就不创建；如果是true的时候，当session存在就那存在的session，若没有，就创建一个

String getId()：获取 sessionId

boolean isNew()：判断是否是新用户（第一次访问）

void invalidate()：使session失效（退出登陆、注销）

void setAttribute():获取用户信息

object getAttribute():填充用户信息

void setMaxInactiveInterval(秒)：设置最大有效非活动时间【比如你访问一个网站时，然后去访问其他页面了，当你没有超过有效时间时，再回到原来的页面可以继续原来的操作，超过有效时间时页面就会要求你重新登陆】

int getMaxInactiveInterval()：获取最大有效非活动时间

示例：

登录

客户端在第一次请求服务端时，如果服务端发现此请求没有JSESSIONID，则会创建一个name=jsessionId的cookie

Cookie:

a.不是内置对象，要使用必须new

b.但是服务端会自动生成一个（服务端自动new一个）name=JSESSIONID的cookie并返回给客户端

cookie和session的区别：

	cookie	session
保存的位置	客户端	服务端
安全性	较不安全	较安全
保存的内容	cookie(String,String)	session(String,Object)

5.pageCotext:

6.application:全局对象

String application.getContextPath()//获取当前项目的虚拟路径

String application.getRealPath(String a[])//获取虚拟路径对应的相对路径，参数是虚拟路径的值

示例在index1.jsp 中

7.config:

8.exception:

9.page:

总结一下：

1.pageContext：JSP页面容器

2.request：请求对象

3.response：响应对象

4.session：会话对象

5.appliation：全局对象

6.config：配置对象（服务器配置信息）

7.out：输出对象

8.page：当前JSP页面对象（相当于java中的this）

9.exception：异常对象

四种范围对象（从小到大）：

1.pageContext：JSP页面容器 （也称page对象）：当前页面有效

2.request：请求对象 同一次请求有效（请求转发后有效，重定向后无效）

4.session：会话对象　　同一次会话有效（只要不关/切换浏览器都有效）

5.appliation：全局对象　　全局有效，整个项目运行期间都有效，切换浏览器都有效，关闭服务，其他项目，无效

->关闭服务，其他项目:JNDI可以做到

以上四个对象共有的方法：

Object getAttribute(String name)：根据属性名获取属性值

void setAttribute(String name,Object obj)：设置属性值（新增，修改）

eg：setAttribute("a","b");//如果a对象之前不存在，则新建一个a对象，然后再改

如果a对象之前存在，则将a的值改为b

void removeAttribute(String name)：根据属性名，删除对象。

总结：

1.以上一个范围对象，通过setAttribute()赋值，通过getAttribute()取值

2.以上范围对象，尽量使用最小范围，因为对象范围越大，造成的性能损失越多