

知识点列表

| 编号 | 名称 | 描述 | 级别 |
|----|-----------------|---|----|
| 1 | 【案例 1】购物车 | 购物车的第二个版本,使用 Cookie 机制解决如下问题: 用户购物信息保存于 Session 中,当关闭浏览器,用户购物数据即刻消失 | ** |
| 2 | Servlet 基础 | 知识点总结与回顾 | * |
| 3 | Servlet 核心 | 知识点总结与回顾 | ** |
| 4 | 状态管理 | 知识点总结与回顾 | ** |
| 5 | 数据库访问 | 知识点总结与回顾 | ** |
| 6 | 过滤器与监听器 | 知识点总结与回顾 | ** |
| 7 | Servlet 的线程安全问题 | 了解 Servlet 中的线程安全问题 | * |
| 8 | 什么是 jsp | 理解 JSP 的产生与运行原理 | * |
| 9 | jsp 的组成 | 掌握 JSP 的基本组成,包括指令、隐含对象、活动元素等; 通过案例演示,理解这些知识点。 | ** |

注: "*"理解级别 "***"掌握级别 "****"应用级别

1. 购物车_02 **

【案例 1】购物车 **

案例描述

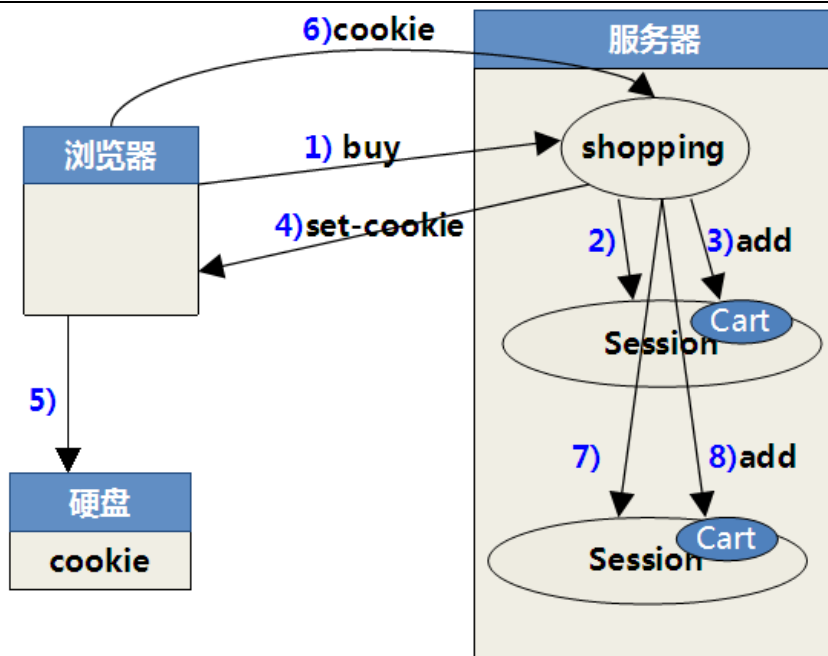
购物车的第二个版本,使用 Cookie 机制解决如下问题:

用户购物信息保存于 Session 中,当关闭浏览器,用户购物数据即刻消失。

在今后的开发过程中,最好使用本版本。

图示演示

购物流程图



- 1) 用户发送购买请求
- 2) 第 1 次购买时，服务器将创建 Session 对象
- 3) 用户购物信息装入购物车 (Cart) 对象中，Cart 是一个 CartItem 的 List 集合
- 4) 将 Cart 中保存的 List 集合转换为一个等价的字符串保存到 Cookie 中
在 CartItem 中最重要的信息就是商品 ID 和购买数量
- 5) 浏览器将 Cookie 信息保存到硬盘上
- 6) 当浏览器被关闭后，用户重新访问该地址时带着 Cookie 信息
- 7) 服务器重新创建一个 Session 对象
- 8) 在该 Session 对象中对 cookie 信息进行恢复
删除、更新操作流程相同

参考代码

- 1) 请下载 web07_shopping.zip
- 2) 新建 web09_shopping&&拷贝

● 修改“购买”功能

3) 修改 Cart.java

增加 2 个方法 store()和 load()

```
/**
 * 将cart中的所有商品信息，即items集合中的数据
 * 转变成一个类似 "3,8;4,11;9,2"这样的字符串。
 * 如果集合为空，返回"0"。
 * @return
```

```

*/
public String store(){
    StringBuffer sb = new StringBuffer();
    if(items.size() == 0){
        sb.append("0");
    }else{
        for(int i=0;i<items.size();i++){
            CartItem item = items.get(i);
            sb.append(item.getC().getId()+ ","
                + item.getQty()+ ";");
        }
    }
    if(sb.length() > 1){
        sb.deleteCharAt(sb.length()-1);
    }
    return sb.toString();
}

/**
 * 依据content(类似 "3,8;4,11;9,2"这样的字符串)
 * 重新恢复cart中用户所购买的商品，即items集合。
 * @param content
 */
public void load(String content){
    if(content == null || content.equals("0")){
        return;
    }
    String[] pcs = content.split(";");
    for(int i=0;i<pcs.length;i++){
        String pc = pcs[i];
        String[] strs = pc.split(",");
        long id = Long.parseLong(strs[0]);
        int qty = Integer.parseInt(strs[1]);
        ComputerDAO dao = new ComputerDAO();
        try {
            Computer c = dao.findById(id);
            CartItem item = new CartItem();
            item.setC(c);

```

```

        item.setQty(qty);
        items.add(item);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

4) 测试

a. 新建 TestCart.java

```

package test;

import bean.Cart;
import bean.CartItem;
import entity.Computer;

public class TestCart {

    public static void main(String[] args) {
        Cart cart = new Cart();
        //测试 1 : store 方法
        Computer c = new Computer(
            "x200", "x200.jpg", "good", 2000);
        c.setId(1);
        CartItem item = new CartItem();
        item.setC(c);
        item.setQty(2);
        cart.add(item);

        Computer c2 = new Computer(
            "x200", "x200.jpg", "good", 2000);
        c2.setId(2);
        CartItem item2 = new CartItem();
        item2.setC(c2);
        item2.setQty(22);
        cart.add(item2);
        String content = cart.store();
        System.out.println(content);
    }
}

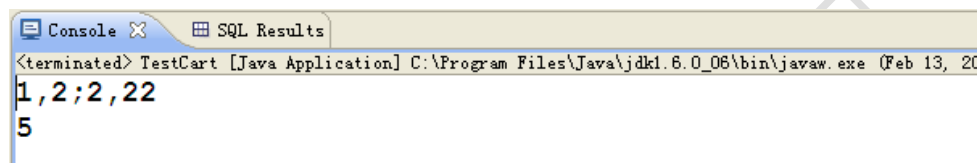
```

```
//测试 2 : load 方法
String content1 = "1,3;2,11;3,8";
cart.load(content1);
System.out.println(cart.list().size());

}

}
```

b. 运行测试



```
<terminated> TestCart [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Feb 13, 2011 1,2;2,22 5
```

5) 拷贝 CookieUtil.java 到项目中

```
package util;

import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.net.URLEncoder;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * cookie工具类
 * @author teacher
 *
 */
public class CookieUtil {
    //缺省的应用名
    private static String default_path =
        "/web09_shopping"; //注意
    //缺省的生存时间
    private static int default_age = 365 * 24 * 3600;

    /**
     * 添加一个cookie
     */
}
```

```

* @param name
* @param value
* @param response
* @param age
* @throws UnsupportedOperationException
*/
public static void addCookie(String name,
    String value, HttpServletResponse response,
    int age) throws UnsupportedOperationException{
    Cookie cookie =
        new Cookie(
            name, URLEncoder.encode(value, "utf-8"));
    cookie.setMaxAge(age);
    cookie.setPath(default_path);
    response.addCookie(cookie);
}

public static void addCookie(String name,
    String value, HttpServletResponse response)
    throws UnsupportedOperationException{
    addCookie(name, value, response, default_age);
}

/**
 * 依据cookie的名字，查找cookie的值，如果
 * 找不到，返回null。
 * @param name
 * @param request
 * @return
 * @throws UnsupportedOperationException
 */
public static String findCookie(String name,
    HttpServletRequest request)
    throws UnsupportedOperationException{
    String value = null;
    Cookie[] cookies = request.getCookies();
    if(cookies != null){
        for(int i=0; i<cookies.length; i++){
            Cookie curr = cookies[i];

```

```

        if(curr.getName().equals(name)){
            value =
                URLDecoder.decode(
                    curr.getValue(),"utf-8");
        }
    }
}
return value;
}

/**
 * 删除cookie
 * @param name
 * @param response
 */
public static void deleteCookie(String name,
    HttpServletResponse response){
    Cookie cookie = new Cookie(name,"");
    cookie.setMaxAge(0);
    cookie.setPath(default_path);
}
}

```

6) 修改 ActionServlet

```

package web;

import java.io.IOException;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import util.CookieUtil;

```

```
import bean.Cart;
import bean.CartItem;
import dao.ComputerDAO;
import entity.Computer;

public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

        String uri = request.getRequestURI();
        String path =
            uri.substring(uri.lastIndexOf("/"),
                uri.lastIndexOf("."));
        if(path.equals("/list")){
            ComputerDAO dao =
                new ComputerDAO();
            try {
                List<Computer> computers =
                    dao.findAll();
                request.setAttribute("computers", computers);
                request.getRequestDispatcher("computer_list.jsp")
                    .forward(request, response);
            } catch (Exception e) {
                e.printStackTrace();
                throw new ServletException(e);
            }
        }else if(path.equals("/buy")){
            long id = Long.parseLong(request.getParameter("id"));
            ComputerDAO dao =
                new ComputerDAO();
            try {
                Computer c = dao.findById(id);
                CartItem item = new CartItem();
                item.setC(c);
                item.setQty(1);
                HttpSession session =
```



```

        request.getSession();
        Cart cart = (Cart)session.getAttribute("cart");
        if(cart == null){
            //如果是第一次购买，需要先创建好 cart 对象
            //然后绑定到 session 对象上。
            cart = new Cart();
            //尝试查找名叫 cart 的 cookie,恢复之前购买的商
            //品数据
            cart.load(CookieUtil.findCookie("cart",request));
            session.setAttribute("cart", cart);
        }
        boolean flag = cart.add(item);
        if(!flag){
            //已经购买过该商品，则提示用户
            request.setAttribute("buy_error" + id,
                "已经购买过该商品");
            request.getRequestDispatcher("list.do")
                .forward(request, response);
        }else{
            //没有买过，返回到商品列表
            //将 cart 中的数据以 cookie 的形式备份到客户端
            CookieUtil.addCookie("cart",
                cart.store(), response);
            response.sendRedirect("list.do");
        }
    } catch (Exception e) {
        e.printStackTrace();
        throw new ServletException(e);
    }
}
}else if(path.equals("/dear")){
    HttpSession session = request.getSession();
    Cart cart = (Cart)session.getAttribute("cart");
    cart.clear();
    CookieUtil.deleteCookie("cart", response);
    response.sendRedirect("cart.jsp");
}else if(path.equals("/delete")){
    long id = Long.parseLong(request.getParameter("id"));
    HttpSession session = request.getSession();

```

```

        Cart cart = (Cart)session.getAttribute("cart");
        cart.delete(id);
        CookieUtil.addCookie("cart", cart.store(), response);
        response.sendRedirect("cart.jsp");
    }else if(path.equals("/update")){
        long id = Long.parseLong(request.getParameter("id"));
        int qty = Integer.parseInt(request.getParameter("qty"));
        HttpSession session = request.getSession();
        Cart cart = (Cart)session.getAttribute("cart");
        cart.modify(id, qty);
        CookieUtil.addCookie("cart", cart.store(), response);
        response.sendRedirect("cart.jsp");
    }
}
}

```

- 修改“查看购物车”功能

7) 修改 cart.jsp

```

<%@page pageEncoding="utf-8" contentType="text/html; charset=utf-8"%>
<%@page import="bean.* java.util.*,util.*" %>
<html>
    <head>
        <meta http-equiv=Content-Type
            content="text/html; charset=utf-8" />
        <link href="css/main/style.css"
            type="text/css" rel="stylesheet" />
        <script type="text/javascript">
            function f1(v1){
                alert("hello!");
                location='update.do?id='+v1+'&qty='
                    + document.getElementById('num_'+v1).value;
            }
        </script>
    </head>

    <body topMargin="10">
        <div id="append_parent"></div>

```

[illegible]

```

        &nbsp;
    </td>
    <td class="altbg1" width="10%">
        &nbsp;
    </td>
    <td class="altbg1">
        &nbsp;
    </td>
</tr>
</tbody>

<tbody>
    <%
        Cart cart = (Cart)session.getAttribute("cart");
        //尝试恢复之前购买过的商品
        if(cart == null){
            cart = new Cart();
            cart.load(CookieUtil.findCookie("cart",request));
            session.setAttribute("cart",cart);
        }
        if(cart != null && cart.list().size() > 0){
            List<CartItem> items = cart.list();
            for(int i=0;i<items.size();i++){
                CartItem item = items.get(i);
                %>
            <tr>
                <td class="altbg2">
                    <%=item.getC().getModel()%>
                </td>
                <td class="altbg2">
                    <%=item.getC().getPrice()%>
                </td>
                <td class="altbg2">
                    <%=item.getQty()%>
                </td>
                <td class="altbg2">
                    <input type="text" size="3" value=""
                        id="num_<%=item.getC().getId()%>" />
                </td>
            </tr>
        }
    %>

```

```

        </td>
        <td class="altbg2">
            <a href="javascript:;"
                onclick="f1(<%=item.getC().getId()%>);">更改数量</a>
        </td>
        <td class="altbg2">
            <a href="delete.do?id=<%=item.getC().getId()%>">删除</a>
        </td>
    </tr>

    <%
    }
    %>
    <tr>
        <td class="altbg1" colspan="6">
            <b>总价格：¥ <%=cart.total()%> </b>
        </td>
    </tr>
    <%
    }else{
        //还没有购买商品
        %>
        <tr>
            <td class="altbg2" colspan="6">
                <b>还没有选购商品</b>
            </td>
        </tr>
        <%
    }
    %>
</tbody>
</table>

        <br />
<center>
    <input class="button" type="button" value="返回商品列表"
        name="settingsubmit" onclick="location = 'list.do';">
    <input class="button" type="button" value="清空购物车"
        name="settingsubmit"
        onclick="location = 'clear.do';">

```

```
</center>

        </td>

    </tr>

</tbody>

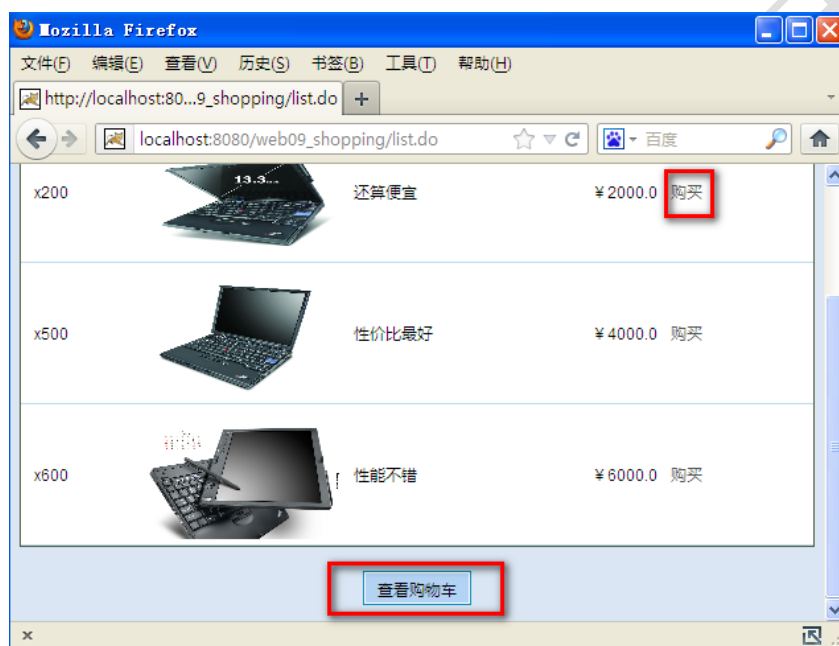
</table>

</body>

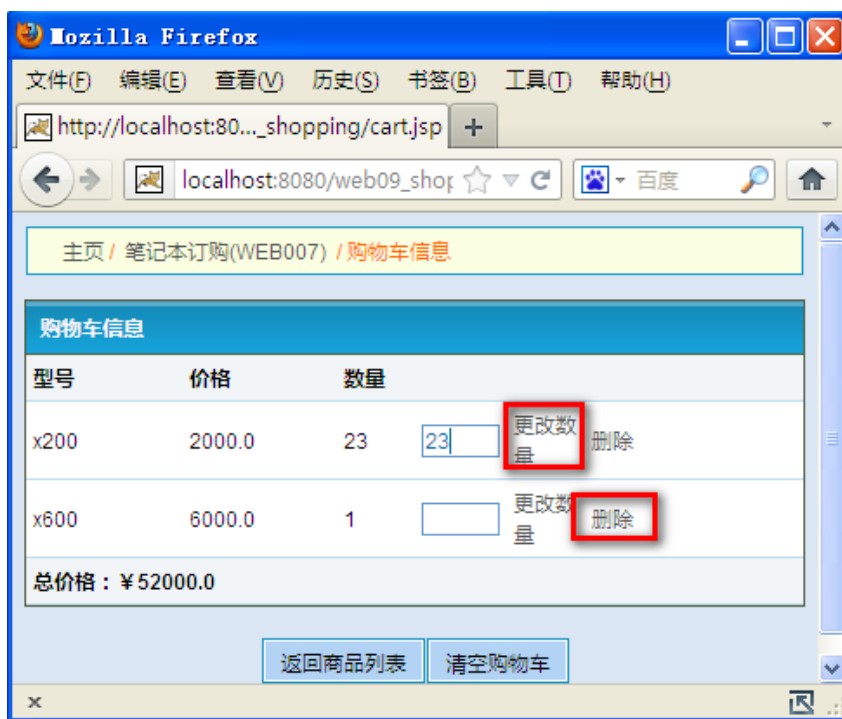
</html>
```

8) 测试

a. 点击“购买” && “查看购物车”



b. 在“购物车”页面继续做“更改数量”、“删除”等操作



c. 重启浏览器

d. 访问 http://localhost:8080/web09_shopping/cart.jsp
购物车中的用户的购物信息还在。

2. servlet 知识点小结 **

2.1. Servlet 基础 *

- 1) 什么是 Servlet? **
- 2) 如何开发一个 Servlet? **
- 3) 什么是 Servlet 容器? **

2.2. Servlet 核心 **

2.2.1. 核心的类与接口 *

- 1) Servlet 接口
- 2) GenericServlet 抽象类
- 3) HttpServlet 抽象类
- 4) ServletRequest,ServletResponse 接口
- 5) ServletConfig 接口
 - a. ServletContext getServletContext();
 - b. String getInitParameter(String paraName);
- 6) HttpServletRequest 接口
 - a. String getParameter(String name);
 - b. String[] getParameterValues(String name);
 - c. setCharacterEncoding(String code);
 - d. RequestDispatcher getRequestDispatcher(String url);
 - e. setAttribute(String name,Object obj);
 - f. Object getAttribute(String name);
 - g. removeAttribute(String name);
 - h. String getContextPath();
 - i. String getRequestURI();
 - j. HttpSession getSession()/getSession(boolean flag);
 - k. Cookie[] getCookies();
- 7) HttpServletResponse 接口
 - a. setContentType(String str);
 - b. PrintWriter getWriter();
 - c. sendRedirect(String url);
 - d. encodeURL(String url);
 - e. encodeRedirectURL(String url);
 - f. addCookie(Cookie cookie);
- 8) ServletContext 接口
 - a. String getRealPath(String str);
 - b. setAttribute(String name,Object obj);
 - c. Object getAttribute(String name);
 - d. removeAttribute(String name);
 - e. String getInitParameter(String name);

2.2.2. servlet 的生命周期 *

- 1) 什么是 servlet 生命周期?
- 2) servlet 生命周期的四个阶段?
<load-on-startup> 配置
- 3) 如何 override init 方法?

init()或者 init(ServletConfig config)

- 4) servlet 的初始化参数如何配置?

2.2.3. 表单处理 **

- 1) 如何读取表单中的参数
- 2) 如何处理中文

2.2.4. get 请求与 post 请求 **

- 1) 哪一些是 get 请求, 哪一些是 post 请求?
- 2) get 请求与 post 请求的区别?

2.2.5. 转发与重定向 **

- 1) 什么是重定向?
- 2) 如何重定向?
- 3) 重定向的特点?
- 4) 重定向编程需要注意的问题?
- 5) 什么是转发?
- 6) 如何转发?
- 7) 转发的特点?
- 8) 转发编程需要注意的问题?
- 9) 转发与重定向的区别

2.3. 状态管理 **

2.3.1. 什么是状态管理 *

2.3.2. Cookie **

- 1) 什么是 cookie?
- 2) 如何创建一个 cookie?
- 3) cookie 的生存时间
- 4) cookie 的路径问题
- 5) 编码问题
- 6) cookie 的限制

2.3.3. Session **

- 1) 什么是 session
- 2) 如何获得 session
- 3) session 的常用方法
 - a. `String getId();`
 - b. `String getRealPath(String str);`
 - c. `setAttribute(String name, Object obj);`
 - d. `Object getAttribute(String name);`
 - e. `setMaxInactiveInterval(int seconds);`
 - f. `invalidate();`
 - g. `ServletContext getServletContext();`
- 4) session 的超时
- 5) 删除 session

2.3.4. 禁止 cookie 以后，如何继续使用 session? **

- 1) url 重写是什么？
- 2) 如何实现 url 重写？
 - a. `encodeURL(String url);`
 - b. `encodeRedirectURL(String url);`

2.3.5. 案例 **

- 1) session 验证
- 2) 验证码
- 3) 购物车

2.4. 数据库访问 **

- 1) 什么是 dao?
- 2) 如何写一个 dao?

2.5. 过滤器与监听器 **

- 1) 什么是过滤器?
- 2) 如何写一个过滤器?
- 3) 配置初始化参数
- 4) 过滤器的优先级

- 5) 过滤器的优点
- 6) 什么是监听器？
- 7) 如何写一个监听器？

2.6. servlet 的线程安全问题 *

1) servlet 线程安全问题产生的原因

在默认情况下，容器只会为每一个 servlet 类创建唯一的一个实例，当有多个请求到达容器，就有可能有多个线程同时访问同一个实例。

2) 解决方式

a. 加锁(可以对整个 service 方法加锁，或者对代码块加锁，建议使用代码块加锁)。

b. 让 servlet 实现 SingleThreadModel 接口(不建议使用)

SingleThreadModel 接口是一个标识接口(没有定义任何的方法)。容器会为实现该接口的 servlet 创建多个实例，即一个线程分配一个。这种方式创建了过多的 servlet 实例，系统开销太多，不建议使用。

c. servlet 的属性尽量设置成可读的，不要去修改。

3. Jsp **

3.1. 什么是 jsp? *

sun 公司制订的一种服务器端动态页面生成技术规范。

3.2. jsp 的组成? *

1) html(html,css,javascript)

2) java 代码

✓ 第一种形式

java 代码片断 <% %>

✓ 第二种形式

jsp 表达式 <%= %>

✓ 第三种形式

jsp 声明 <%! %>

3) 指令

✓ **page 指令**

- ◆ import 属性
- ◆ pageEncoding 属性
- ◆ contentType 属性
- ◆ session 属性

true(缺省)/false。如果值为 false,则对应的 servlet 代码当中不会生成声明和创建 session 的代码。也就是说,不能够使用 session 隐含对象了。

- ◆ isELIgnored 属性

true(缺省)/false,是否忽略 el 表达式,如果是 true,忽略。

- ◆ isErrorPage 属性

true/false(缺省),当前 jsp 是否是一个错误处理页面,如果是 true,是错误处理页面。

- ◆ errorPage 属性

用于指定错误处理页面。

✓ **include 指令**

- ◆ file 属性
- ◆ taglib 指令

用于导入标签

- ◆ uri 属性

标签文件的命名空间

- ◆ prefix 属性

命名空间的前缀

4) **隐含对象 (9 个)**

- ✓ out
- ✓ request
- ✓ response
- ✓ session
- ✓ application
- ✓ exception

当一个页面设置了<%@page isErrorPage="true"%>,则可以 在该页面当中,使用该隐含对象读取错误信息。

- ✓ config

就是 ServletConfig,可以读取 jsp 的配置参数。

- ✓ pageContext

是 PageContext 类的实例,服务器会为每一个 jsp 实例(指的是 jsp 对应的那个 servlet 对象)创建唯一的一个 PageContext 实例。

作用主要有两个:

绑定数据:

setAttribute,getAttribute,removeAttribute

获得其它几个隐含对象:

即在获得了 pageContext 实例之后,可以通过该实例,
获得其它 8 个隐含对象。

✓ page

表示 jsp 实例本身。

5) 活动元素

在 jsp 实例已经运行了,告诉 jsp 引擎做一些处理。

✓ <jsp:forward page=""/>

转发,page 属性指定转发的地址。

✓ <jsp:include page=""/>

一个 jsp 在运行过程当中,调用另外一个 jsp。

✓ <jsp:param name="" value=""/>

设置参数 name 指定参数名 value 指定参数值

✓ <jsp:useBean id="" scope="" class=""/>

在指定的范围绑定一个对象。

范围指的是四个对象 pageContext,request,session,servletContext。

也就是说 scope 的值可以是"page","request","session","application"。

✓ <jsp:getProperty/>

✓ <jsp:setProperty name="" property="" value=""/>

✓ <jsp:setProperty name="" property="" param=""/>

依据请求参数给属性赋值。

✓ <jsp:setProperty name="" property="*">

使用"自省机制"给属性赋值。

6) 注释

<!-- <%=new Date()%> -->

注释中的代码会执行,但不会在页面上输出。

<!--xxxx-->

注释中的代码不会执行,也不会页面上输出。

7) jsp 源文件如何转换成.java 文件? *

✓ html ----> service(),使用 out.write()输出。

✓ <% %> ----> service(),照搬。

✓ <%= %> ----> service(),使用 out.print()输出。

✓ 指令 ----> 会影响源代码的生成,比如导包。

✓ <%! %> ----> jsp 声明中定义的变量会变为对应的 servlet 类的属性,
定义的方法会变成对应的 servlet 类的一个方法。

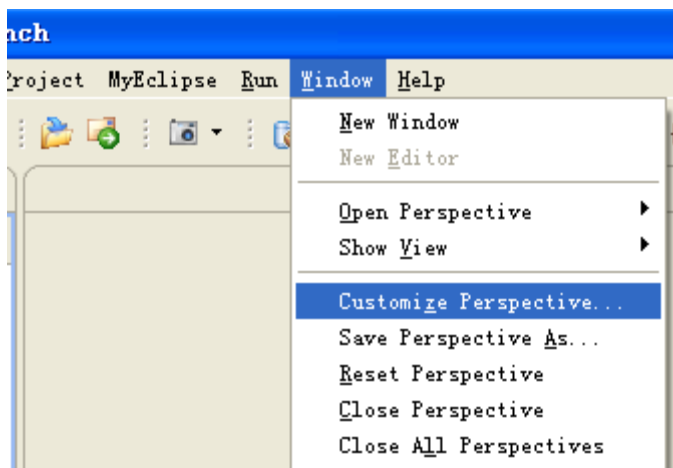
【案例 2】JSP **

1) 新建工程 web09_jsp

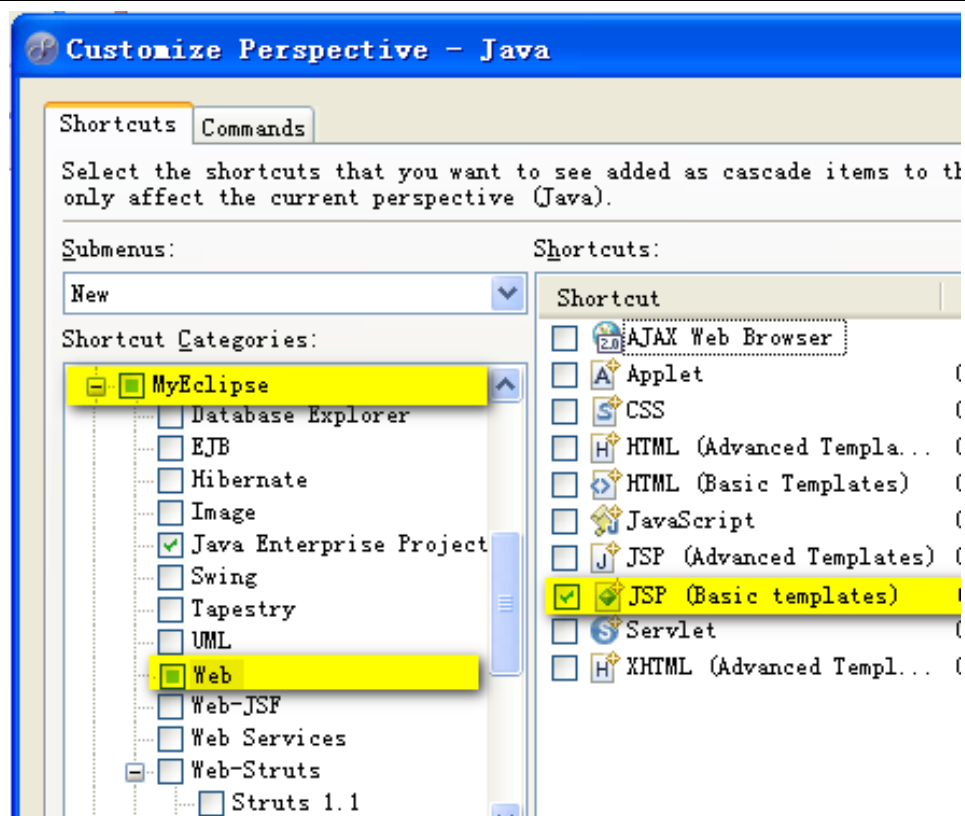
- 小知识

MyEclipse 工具定制 jsp 模板

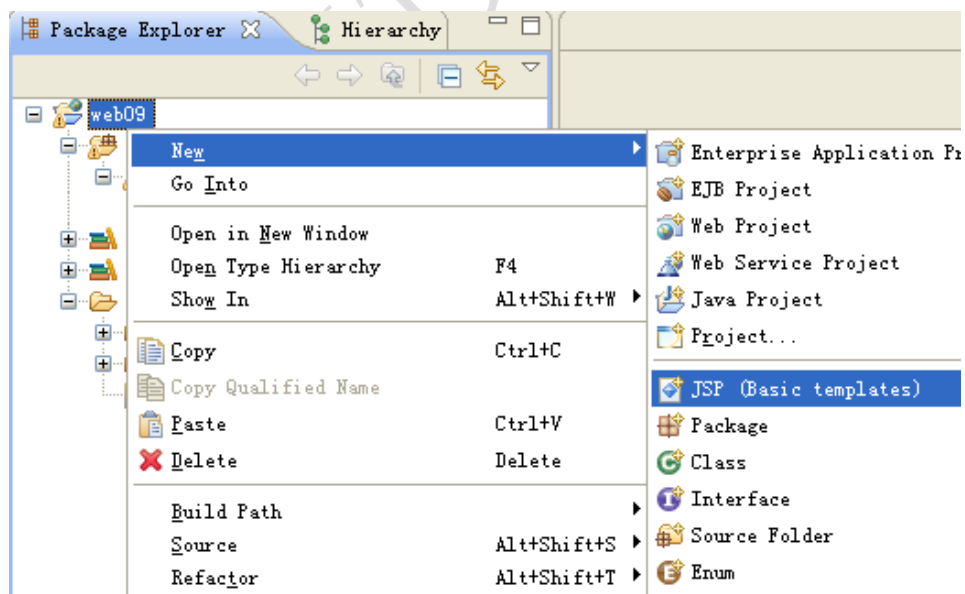
a. 打开 “Window” -- “Customize Perspective”



b. 选择 “MyEclipse” -- “Web” -- “JSP Basic templates” -- “OK”



c. 根据 JSP (Basic Template) 新建 JSP 文件jsp01.jsp



d. jsp01.jsp

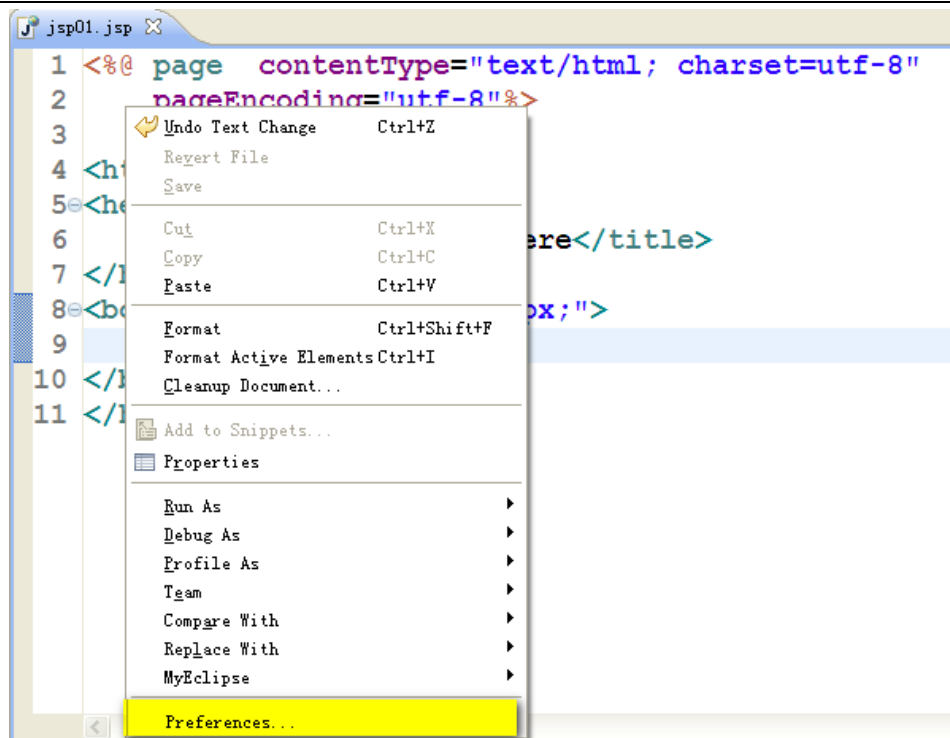
```
jsp01.jsp
1 <%@ page language="java"
2   contentType="text/html; charset=ISO-8859-1"
3   pageEncoding="ISO-8859-1"%>
4 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Trans:
5 "http://www.w3.org/TR/html4/loose.dtd">
6 <html>
7 <head>
8   <meta http-equiv="Content-Type"
9     content="text/html; charset=ISO-8859-1">
10  <title>Insert title here</title>
11 </head>
12 <body>
13
14 </body>
15 </html>
```

e. 修改 jsp01.jsp 为自定义的模板

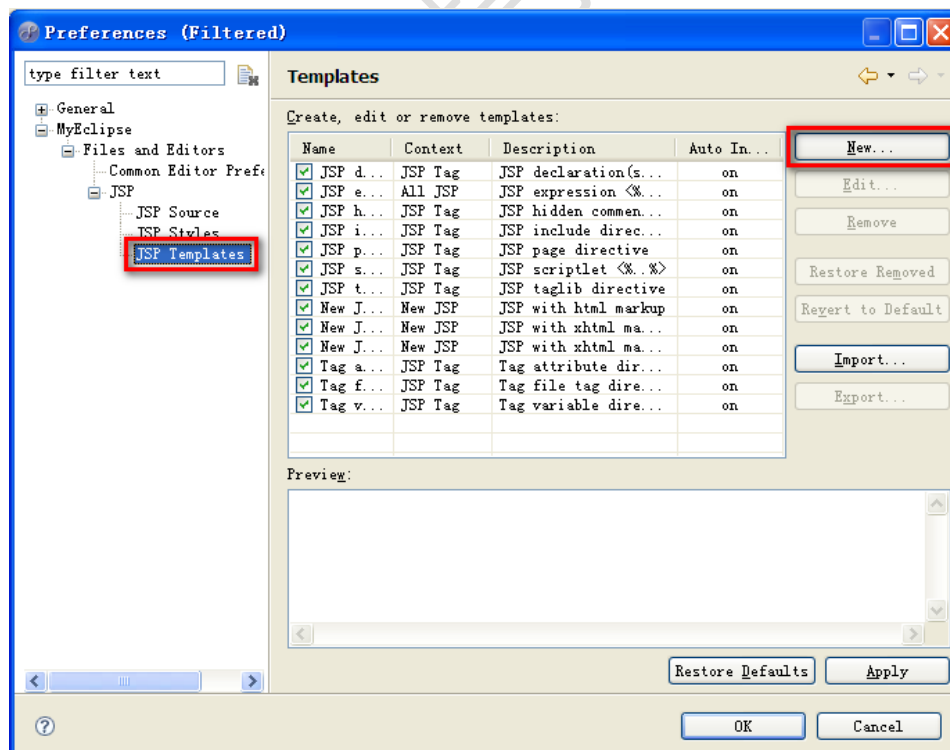
```
jsp01.jsp
1 <%@ page contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3   <!-- jsp声明 -->
4 <html>
5 <head>
6   <title>Insert title here</title>
7 </head>
8 <body style="font-size:30px;">
9
10 </body>
11 </html>
```

f. 将内容“全选” -- “复制”

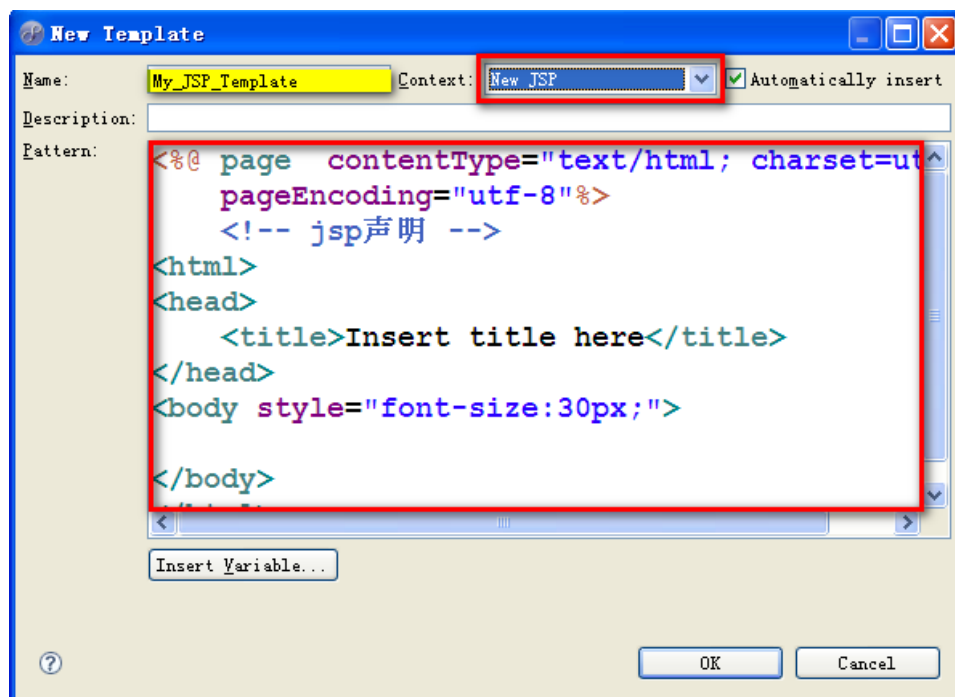
g. 在 jsp 文档上点击“右键”，选择“Properties”



h. 选择“JSP Template” -- “New”



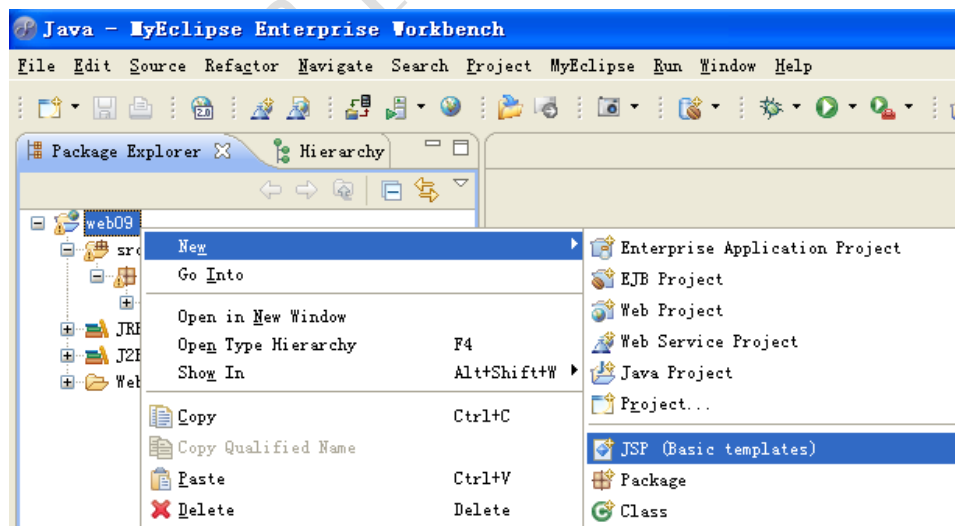
- i. 填写 Name 为 “My_Jsp_Template” , 选择 “Context” 为 “New JSP”
将内容 “粘贴” 到 Pattern 中, 点击 “OK” 确定



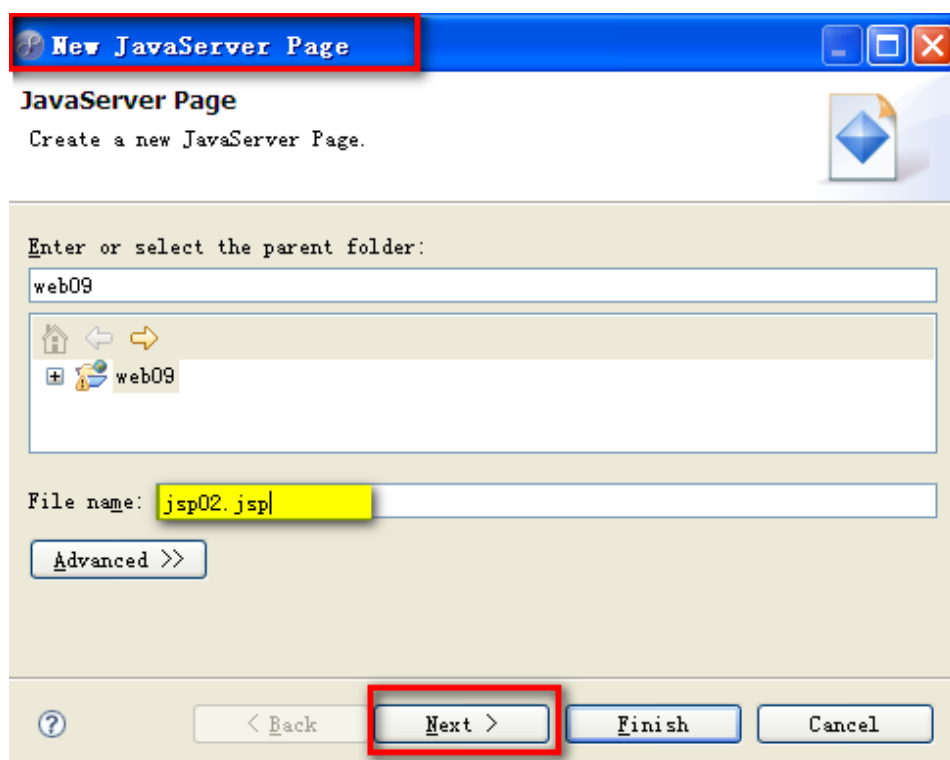
- j. OK, 自定义模板成功

当下一次新建 JSP 文件时

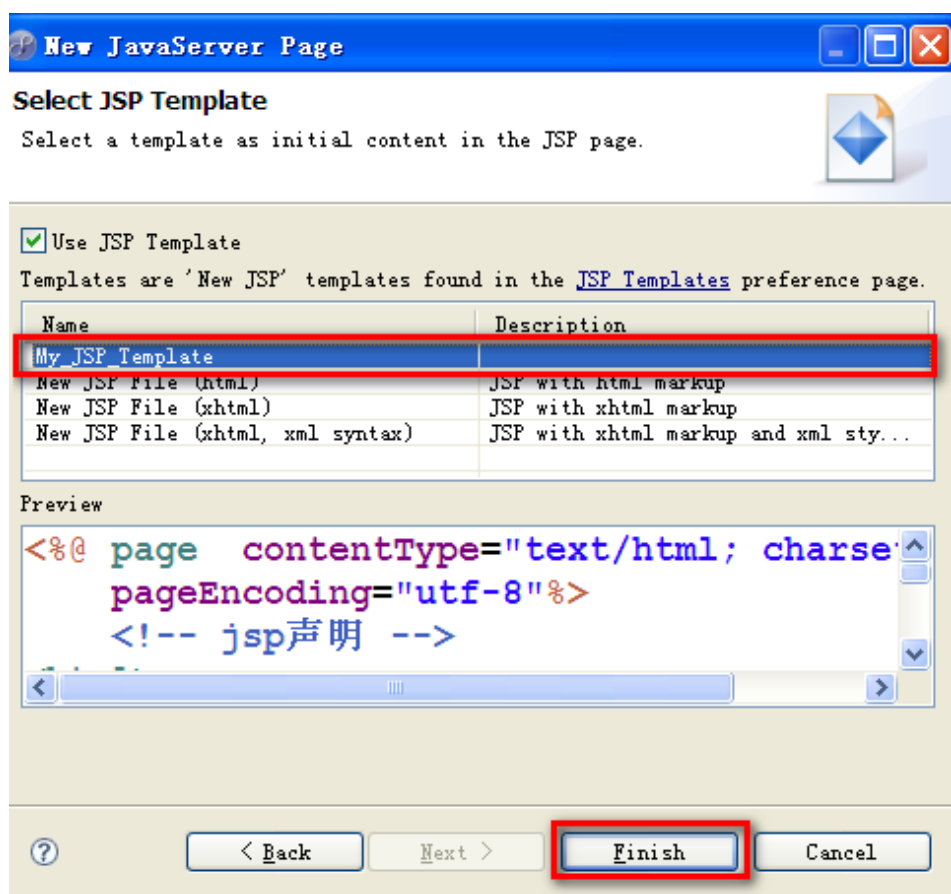
- k. 按 JSP (Basic Template) 新建 JSP 文档



- l. 命名为 “jsp02.jsp” , 点击 “Next” 按钮



m. 选择 “My_JSP_Template” , 点击 “Finish”



n. OK, 新建的即为自定义模板的 JSP 文件



- JSP 声明

2) JSP 声明的三种形式

```

1 <%@ page contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3   <!-- jsp声明 -->
4 <html>
5 <head>
6   <title>Insert title here</title>
7 </head>
8 <body style="font-size:30px;">
9   <%!
10       int i = 100;
11       int sum(int a1,int a2){
12           return a1 + a2;
13       }
14   %>
15   <%=i%><br/>
16   <%=sum(1,2)%>
17 </body>
18 </html>
19

```

- ✓ 第一种形式 java 代码片断 <% %>
- ✓ 第二种形式 jsp 表达式 <%= %>
- ✓ 第三种形式 jsp 声明 <%! %>

● page 指令的 session 属性

session 属性的取值为 true(缺省)/false。

如果值为 false,则对应的 servlet 代码当中不会生成声明和创建 session 的代码。

也就是说,不能够使用 session 隐含对象了。

3) 新建 jsp02.jsp

```

<%@ page session="false"
   contentType="text/html; charset=utf-8"
   pageEncoding="utf-8"%>
<html>
<head>
   <title>Insert title here</title>
</head>
<body style="font-size:30px;">
   <%
       session.setAttribute("username","zs");
   %>
</body>

```

```
</html>
```

4) 测试

访问 <http://localhost:8080/web09/jsp02.jsp> 将出现 500 错误

- page 指令的 `isErrorPage` 属性和 `errorPage` 属性

5) 新建 jsp03.jsp

```
jsp03.jsp x errorHandler.jsp
1 <%@ page errorPage="errorHandler.jsp"
2     contentType="text/html; charset=utf-8"
3     pageEncoding="utf-8"%>
4 <html>
5 <head>
6     <title>Insert title here</title>
7 </head>
8 <body style="font-size:30px;">
9     <%
10         String num = request.getParameter("num");
11         out.println(Integer.parseInt(num) + 100);
12     %>
13 </body>
14 </html>
15
```

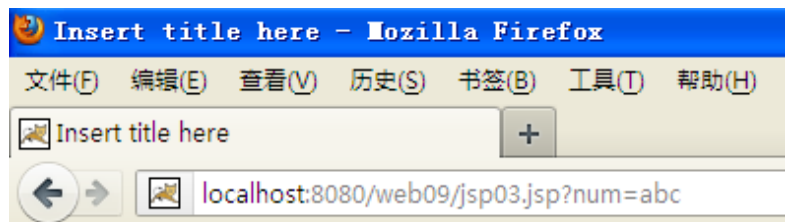
6) 新建 errorHandler.jsp

```
errorHandler.jsp x jsp03.jsp
1 <%@ page isErrorPage="true"
2     contentType="text/html; charset=utf-8"
3     pageEncoding="utf-8"%>
4 <!-- exception隐含对象 -->
5 <html>
6 <head>
7     <title>Insert title here</title>
8 </head>
9 <body style="font-size:30px;">
10     系统运行出错:<%=exception.getMessage() %>
11 </body>
12 </html>
13
```

7) 测试

访问 <http://localhost:8080/web09/jsp03.jsp?num=abc>

因为出异常，转到 errorHandler.jsp 页面



系统运行出错:For input string: "abc"

● 对 JSP 配置参数

8) 新建 jsp04.jsp

```

1  <%@ page contentType="text/html; charset=utf-8"
2     pageEncoding="utf-8"%>
3  <html>
4  <head>
5     <title>Insert title here</title>
6  </head>
7  <body style="font-size:30px;">
8     address:
9     <%=config.getInitParameter("address") %>
10 </body>
11 </html>
12

```

9) 配置 web.xml

```

5      xsi:schemaLocation="http://java.sun.com/xml/ns
6      http://java.sun.com/xml/ns/j2ee/web-app_2_4.xs
7      <servlet>
8          <servlet-name>jsp04</servlet-name>
9          <jsp-file>/jsp04.jsp</jsp-file>
10         <init-param>
11             <param-name>address</param-name>
12             <param-value>beijing</param-value>
13         </init-param>
14     </servlet>
15     <servlet-mapping>
16         <servlet-name>jsp04</servlet-name>
17         <url-pattern>/abc.html</url-pattern>
18     </servlet-mapping>
19 </web-app>
20

```

10) 测试

访问 <http://localhost:8080/web09/abc.html>



address: beijing

- 隐藏对象 pageContext

11) 新建 jsp05.jsp


```

jsp05.jsp x jsp06.jsp
1 <%@ page contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <html>
4 <head>
5   <title>Insert title here</title>
6 </head>
7 <body style="font-size:30px;">
8   <%
9     pageContext.setAttribute("username","zsf");
10    request.setAttribute("pwd","123");
11  %>
12    username:
13    <%=pageContext.getAttribute("username") %>
14    pwd:<%=request.getAttribute("pwd") %>
15    <a href="jsp06.jsp">访问jsp06</a>
16 </body>
17 </html>
18

```

12) 新建 jsp06.jsp

```

jsp06.jsp x
1 <%@ page contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <html>
4 <head>
5   <title>Insert title here</title>
6 </head>
7 <body style="font-size:30px;">
8   username:
9   <%=pageContext.getAttribute("username") %>
10  <br/>
11  pwd:<%=request.getAttribute("pwd") %>
12 </body>
13 </html>
14

```

13) 测试

一个 JSP 对应一个 pageContext 对象 所以在jsp06.jsp 中访问不到jsp05.jsp 中 pageContext 对象中的数据 username ;

同时，因为点击“链接”是一次新请求，所以jsp06.jsp 中也访问不到jsp05.jsp 中 request 对象中的数据 username ;

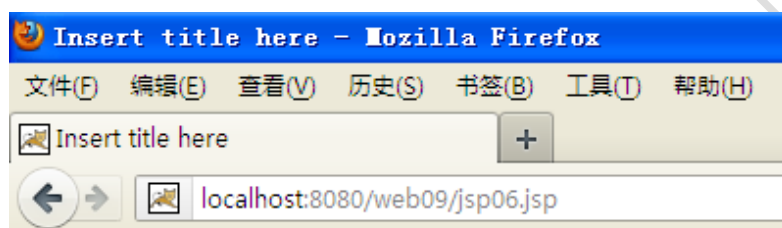
放入 session 中是可以的。

a. 访问 <http://localhost:8080/web09/jsp05.jsp>



username:zsf pwd:123 访问jsp06

b. 点击链接"访问 JSP06"



username:null
pwd:null

如下 jsp 隐藏对象访问范围从小到大

- ✓ pageContext 只有对应的 JSP 实例自己可以访问，生命周期从对应的 JSP 对象创建到 JSP 对象消亡。
- ✓ request 一次请求能访问，生命周期在一起请求和响应期间。
- ✓ session 一次会话期间能访问，多次请求和响应期间都存在。
- ✓ ServletContext 整个应用内部所有组件都能访问，除非服务器关闭，否则一直存在。

- 活动元素
 - `<jsp:forward page="" />` 转发,page 属性指定转发的地址
- 14) 新建 jsp07.jsp

```
jsp07.jsp x jsp08.jsp
1 <%@ page contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <html>
4 <head>
5   <title>Insert title here</title>
6 </head>
7 <body style="font-size:30px;">
8   <%
9     request.setAttribute("username", "zs");
10    %>
11   <jsp:forward page="jsp08.jsp"/>
12 </body>
13 </html>
```

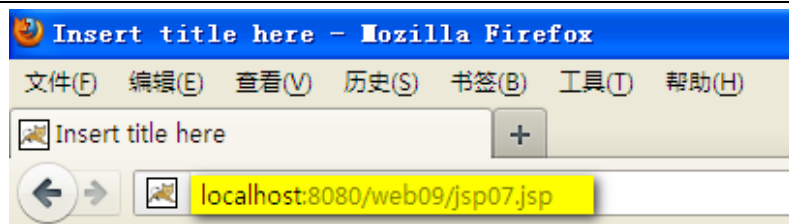
15) 新建jsp08.jsp

```
jsp08.jsp x
1 <%@ page contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <html>
4 <head>
5   <title>Insert title here</title>
6 </head>
7 <body style="font-size:30px;">
8   username:
9   <%=request.getAttribute("username") %>
10 </body>
11 </html>
```

16) 测试

访问 <http://localhost:8080/web09/jsp07.jsp>

地址仍然是 jsp07.jsp，实现了转发



username: zs

- `<jsp:include page=""/>` 一个 jsp 在运行过程当中，调用另外一个 jsp

17) 新建 jsp09.jsp



18) 新建 jsp10.jsp

```

jsp10.jsp
1 <%@ page contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <html>
4 <head>
5   <title>Insert title here</title>
6 </head>
7 <body style="font-size:30px;">
8   jsp10...<br/>
9   pwd:<%=request.getParameter("pwd") %>
10 </body>
11 </html>
12

```

19) 测试

访问 <http://localhost:8080/web09/jsp09.jsp>



jsp09...
jsp10...
pwd:123
 jsp09 other...

- `<jsp:useBean id="" scope="" class=""/>` 在指定的范围绑定一个对象。
 范围指的是四个对象 pageContext,request,session,servletContext。
 也就是说 scope 的值可以是"page","request","session","application"。
 两段代码是等价的

Tarena
达内科技


useBean Example


```


<jsp:useBean id="user"
              scope="request"
              class="com.tarena.entity.User" />

<%= user%><br>
<%= request.getAttribute("user") %>

```







```

com.tarena.entity.User user = null;
user = request.getAttribute("user");
if(user == null){
    user = new com.tarena.entity.User();
    request.setAttribute("user", user);
}
<%= user%><br>
<%= request.getAttribute("user") %>

```

- **<jsp:getProperty/>**

<jsp:setProperty name="" property="" value="" />

<jsp:setProperty name="" property="" param="" /> 依据请求参数给属性赋值。

<jsp:setProperty name="" property="" /> 使用"自省机制"给属性赋值。

20) 新建 bean.User.java

```
jsp10.jsp  User.java X
1 package bean;
2
3 public class User {
4     private String name;
5     private int age;
6     @Override
7     public String toString() {
8         return "name:" + name + "age:" + age;
9     }
10    public int getAge() {
11        return age;
12    }
13    public void setAge(int age) {
14        this.age = age;
15    }
16    public String getName() {
17        return name;
18    }
19    public void setName(String name) {
20        this.name = name;
21    }
22
23 }
```

21) 新建jsp11.jsp

```

jsp11.jsp
1 <%@ page contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <%@page import="bean.*" %>
4 <html>
5 <head>
6   <title>Insert title here</title>
7 </head>
8 <body style="font-size:30px;">
9   <jsp:useBean id="user" scope="request"
10     class="bean.User"/>
11   <jsp:setProperty name="user" property="name"
12     value="zs"/>
13   <jsp:setProperty name="user" property="age"
14     param="age"/>
15   <jsp:setProperty name="user" property="*/>
16   <%
17     User user2 =
18       (User) request.getAttribute("user");
19     out.println(user2);
20   %>
21 </body>

```

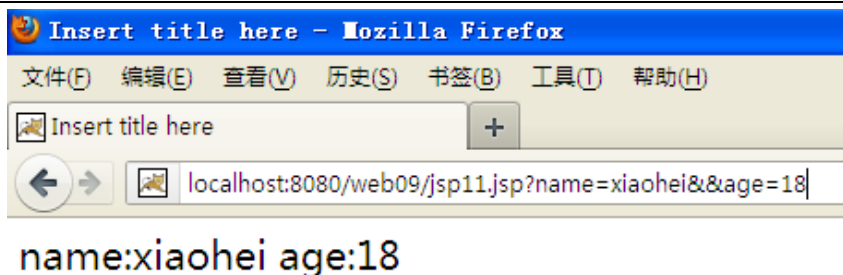
22) 测试

a. 访问 <http://localhost:8080/web09/jsp11.jsp?age=18>



name:zs age:18

b. <http://localhost:8080/web09/jsp11.jsp?name=xiaohei&&age=18>



● Jsp 中的注释

`<!-- <%=new Date()%> -->` 注释中的代码会执行，但不会在页面上输出。
`<%--xxxx--%>` 注释中的代码不会执行，也不会页面上输出。

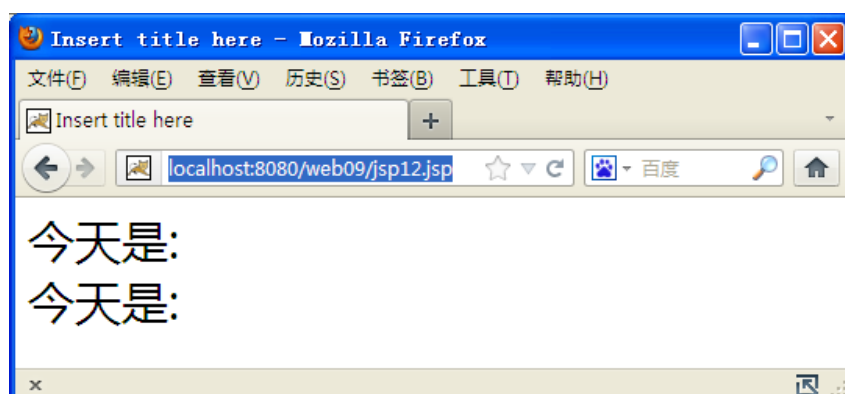
23) 新建 jsp12.jsp



24) 测试

`<!-- <%=new Date()%> -->` 注释中的代码会执行，但不会在页面上输出。
`<%--xxxx--%>` 注释中的代码不会执行，也不会页面上输出。

a. 访问 <http://localhost:8080/web09/jsp12.jsp>



b. 查看源代码

