



Java Server Page

# JSP程序设计(下)

isszym sysu.edu.cn

2017.6.20



# 目录

- 标签库概述
- 标准标签库
- 自定义标签库
- 附录1、字符编码
- 附录2、转换编码
- 附录3、JSP程序的编码问题
- 附录4、参考资料

# 标签库概述

- JSP标准标记库（JSTL）封装了许多JSP应用程序通用的核心功能，它还提供了自定义标签功能。
- `<jsp:forward>`、`<jsp:include>`等标签都是预先定义好的。标签编程的第一个优点是灵活性，例如：`<jsp:forward page="url">`，可以通过属性赋任意值。另一个优点是封装性。标签采用Java类实现，可以包含任意复杂的动作。使用标签也可以减少Scriptlet的使用。
- JSTL标签进行分类：
  - 核心标签(Core Tags)
  - XML标签(XML tags)
  - SQL标签(SQL tags)
  - 格式化标签(Formatting tags)
  - JSTL函数（JSTL Functions）

# 标准标签库

## 例1、核心标签-[for Each](#)标签

forEach.jsp

```
<%@ page language="java" import="java.util.*"
      contentType="text/html; charset=utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>c:forEach 标签实例</title>
</head>
<body>
<c:forEach var="i" begin="1" end="5">
    Item <c:out value="${i}"/><p>
</c:forEach>
</body>
</html>
```

`<c:forEach>`标签的属性



库函数:

jstl-1.2.jar

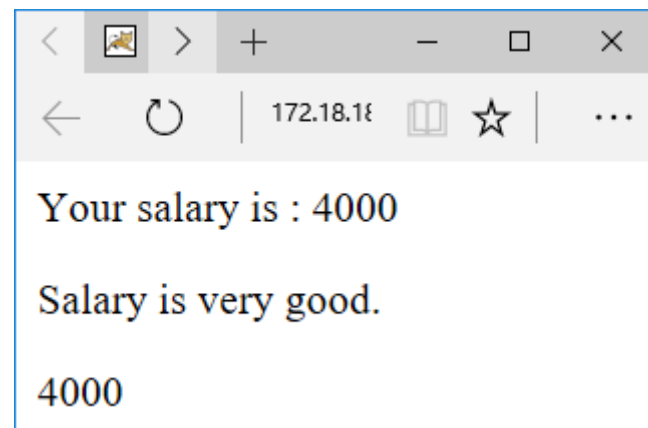
standard-1.1.2.jar

[参考](#)

## 例 2、核心标签- choose标签

choose.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>c:choose Tag Example</title>
</head>
<body>
<c:set var="count" value="${2000}"/>
<c:set var="salary" scope="session" value="${count*2}"/>
<p>Your salary is : <c:out value="${salary}"/></p>
<c:choose>
<c:when test="${salary <= 0}">
Salary is very low to survive.
</c:when>
<c:when test="${salary > 1000}">
Salary is very good.
</c:when>
<c:otherwise>
No comment sir...
</c:otherwise>
</c:choose>
<c:if test="${salary > 2000}" var = "ex">
<p> ${salary} <br />
</c:if>
</body>
</html>
```

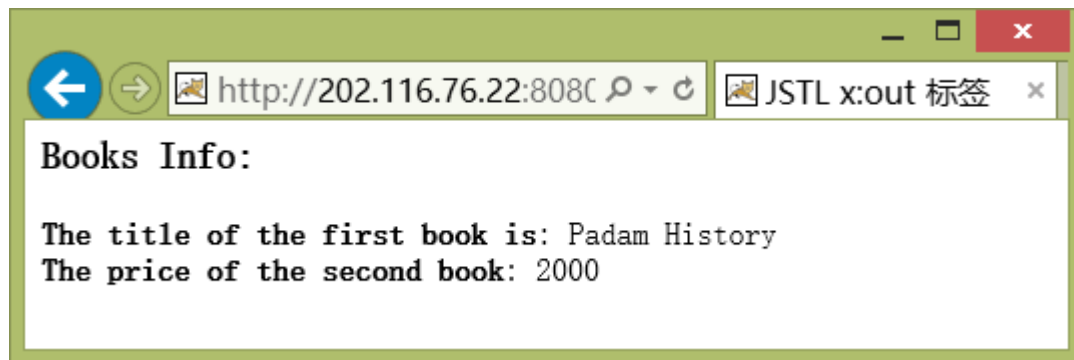


c:if的变量ex取值为true或false，  
是test的计算结果。

### 例 3、[xml](#)标签

xml.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<html>
<head> <title>JSTL x:out 标签</title></head>
<body>
<h3>Books Info:</h3>
<c:set var="xmltext">
<books>
<book>
<name>Padam History</name>
<author>ZARA</author>
<price>100</price>
</book>
<book>
<name>Great Mistry</name>
<author>NUHA</author>
<price>2000</price>
</book>
</books>
</c:set>
<x:parse xml="${xmltext}" var="output"/>
<b>The title of the first book is</b>:
<x:out select="$output/books/book[1]/name" />
<br>
<b>The price of the second book</b>:
<x:out select="$output/books/book[2]/price" />
</body>
</html>
```




tagTest2.jsp

## 例4、sql标签-insert

sqlInsert.jsp

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<html><head><title>JSTL sql:update Tag (insert) </title></head><body>
  <form action="sqlInsert.jsp" method="GET">
    学号: <input type="text" name="num" value="${fn:escapeXml(param["num"])}">
    姓名: <input type="text" name="name" value="${fn:escapeXml(param["name"])}">
    年龄: <input type="text" name="age" value="${fn:escapeXml(param["age"])}">
    <input type="submit" name="submit" value="保存">
  </form>
  <sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/test" user="user" password="123"/>
  <c:catch var="catchException">
    <c:if test="${param.submit!= null}">
      <sql:update dataSource="${snapshot}" var="count">
        INSERT INTO stu(num,name,age) VALUES ('${param.num}', '${param.name}', ${param.age});
      </sql:update>
    </c:if>
  </c:catch>
  <c:if test="${catchException != null}">
    <p>The exception is : ${catchException} <br />
    There is an exception: ${catchException.message}</p>
  </c:if>
  <c:if test="${count>0}">
    <p>Success!</p>
  </c:if>
  <a href="sqlQuery.jsp">浏览</a>
</body>
</html>
```



JSTL sqlInsert.jsp

202.116.100.100

学号: 13040506

姓名: 王二小

年龄: 16

保存

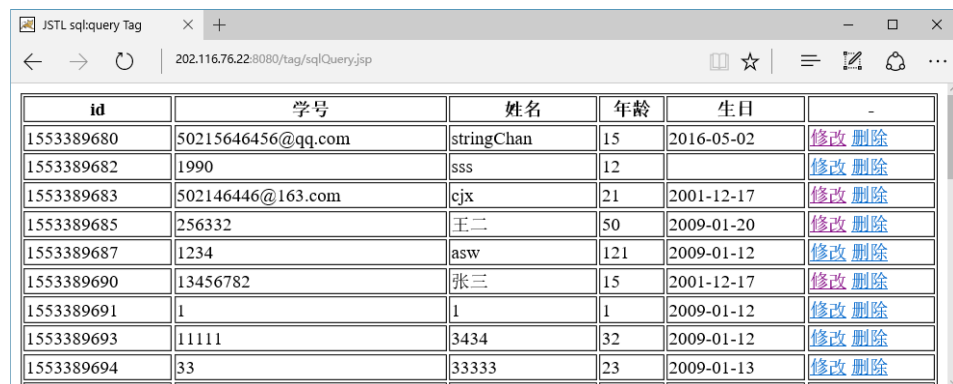
[浏览](#)

## 例5、sql标签-query

## sqlQuery.jsp

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html><head><title>JSTL sql:query Tag</title></head>
<body>
  <sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/test" user="user" password="123"/>
  <sql:query dataSource="${snapshot}" var="result">
    SELECT * from stu;
  </sql:query>

  <table border="1" width="100%">
    <tr><th>id</th><th>学号</th><th>姓名</th>
      <th>年龄</th><th>生日</th><th>-</th>
    </tr>
    <c:forEach var="row" items="${result.rows}">
      <tr>
        <td><c:out value="${row.id}"/></td>
        <td><c:out value="${row.num}"/></td>
        <td><c:out value="${row.name}"/></td>
        <td><c:out value="${row.age}"/></td>
        <td><c:out value="${row.dob}"/></td>
        <td><a href="sqlUpdate2.jsp?id=${row.id}">修改</a>
          <a href="sqlDelete.jsp?id=${row.id}">删除</a>
        </td>
      </tr>
    </c:forEach>
  </table>
  <a href="sqlInsert.jsp">新增</a>
</body>
</html>
```



| id         | 学号                 | 姓名         | 年龄  | 生日         | -                                     |
|------------|--------------------|------------|-----|------------|---------------------------------------|
| 1553389680 | 50215646456@qq.com | stringChan | 15  | 2016-05-02 | <a href="#">修改</a> <a href="#">删除</a> |
| 1553389682 | 1990               | sss        | 12  |            | <a href="#">修改</a> <a href="#">删除</a> |
| 1553389683 | 502146446@163.com  | cjx        | 21  | 2001-12-17 | <a href="#">修改</a> <a href="#">删除</a> |
| 1553389685 | 256332             | 王二         | 50  | 2009-01-20 | <a href="#">修改</a> <a href="#">删除</a> |
| 1553389687 | 1234               | asw        | 121 | 2009-01-12 | <a href="#">修改</a> <a href="#">删除</a> |
| 1553389690 | 13456782           | 张三         | 15  | 2001-12-17 | <a href="#">修改</a> <a href="#">删除</a> |
| 1553389691 | 1                  | 1          | 1   | 2009-01-12 | <a href="#">修改</a> <a href="#">删除</a> |
| 1553389693 | 11111              | 3434       | 32  | 2009-01-12 | <a href="#">修改</a> <a href="#">删除</a> |
| 1553389694 | 33                 | 33333      | 23  | 2009-01-13 | <a href="#">修改</a> <a href="#">删除</a> |

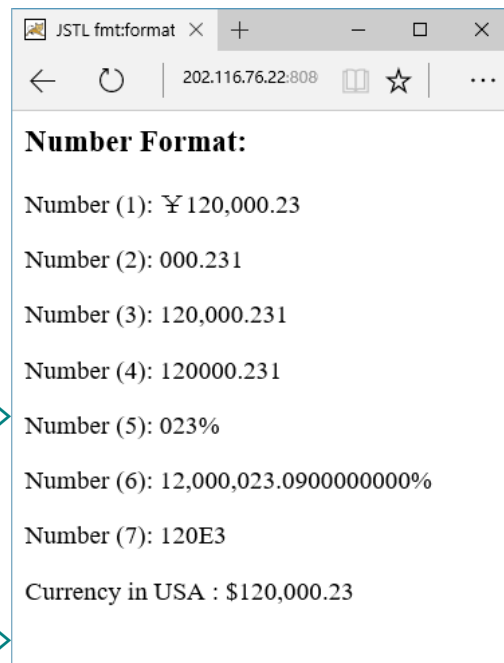


## 例6、[格式化标签](#)

fmtNumber.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<head><title>JSTL fmt:formatNumber Tag</title></head>
<body>
  <h3>Number Format:</h3>
  <c:set var="balance" value="120000.2309" />
  <p>Number (1): <fmt:formatNumber value="${balance}"
    type="currency"/></p>
  <p>Number (2): <fmt:formatNumber type="number"
    maxIntegerDigits="3" value="${balance}" /></p>
  <p>Number (3): <fmt:formatNumber type="number"
    maxFractionDigits="3" value="${balance}" /></p>
  <p>Number (4): <fmt:formatNumber type="number"
    groupingUsed="false" value="${balance}" /></p>
  <p>Number (5): <fmt:formatNumber type="percent"
    maxIntegerDigits="3" value="${balance}" /></p>
  <p>Number (6): <fmt:formatNumber type="percent"
    minFractionDigits="10" value="${balance}" /></p>
  <p>Number (7): <fmt:formatNumber type="number"
    pattern="###.###E0" value="${balance}" /></p>
  <p>Currency in USA :
    <fmt:setLocale value="en_US"/>
    <fmt:formatNumber value="${balance}" type="currency"/></p>
</body>
</html>
```

fmtNumber.jsp



## 例7、 JSTL函数

## functions.jsp

```
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html><head><title>JSTL Functions</title></head>
<body>
<p>xmlString: "I &lt;verb>am&lt;/verb> a test String" </p>
<p>theString: "I am a test String" </p>
<c:set var="theString" value="I am a test String"/>
<c:set var="xmlString" value="I <verb>am</verb> a test String"/>
<table border=1 cellspacing=0 >
<tr><td>函数</td><td>结果</td></tr>
<tr><td>fn:contains(theString, 'test')</td><td> <c:out value="\${fn:contains(theString, 'test')}}" /></td></tr> true
<tr><td>fn:endsWith(theString, 'ing')</td><td> <c:out value="\${fn:endsWith(theString, 'ing')}}" /></td></tr> true
<tr><td>fn:escapeXml(xmlString)</td><td> <c:out value="\${fn:escapeXml(xmlString)}}" /></td></tr> I &lt;verb>am...
<tr><td>fn:indexOf(theString, 'am')</td><td> <c:out value="\${fn:indexOf(theString, 'am')}}" /></td></tr> 2
<tr><td>fn:length(theString)</td><td> <c:out value="\${fn:length(theString)}}" /></td></tr> 18
<tr><td>fn:replace(theString, 'test', 'second')</td><td> <c:out value="\${fn:replace(theString, 'test', 'second')}}" /></td></tr> I am a second String
<tr><td>fn:startsWith(theString, 'ing')</td><td> <c:out value="\${fn:startsWith(theString, 'ing')}}" /></td></tr> false
<tr><td>fn:substring(theString, 2, 4)</td><td> <c:out value="\${fn:substring(theString, 2, 4)}}" /></td></tr> am
<tr><td>fn:substringAfter(theString, 'am')</td><td> <c:out value="\${fn:substringAfter(theString, 'am')}}" /></td></tr> a test String
<tr><td>fn:substringBefore(theString, 'am')</td><td> <c:out value="\${fn:substringBefore(theString, 'am')}}" /></td></tr> I
<tr><td>fn:toLowerCase(theString)</td><td> <c:out value="\${fn:toLowerCase(theString)}}" /></td></tr> i am a test string
<tr><td>fn:toUpperCase(theString)</td><td> <c:out value="\${fn:toUpperCase(theString)}}" /></td></tr> I AM A TEST STRING
<tr><td>fn:trim(' abc de ')</td><td> <c:out value="\${fn:trim(' abc de ')}}" /></td></tr> abc de
<c:set var="string1" value="\${fn:split(theString, ' ')}" />
<c:set var="string2" value="\${fn:join(string1, '-')} " />
<tr><td> &lt;c:set var="string1" value="\${fn:split(theString, ' ')}" /><br>
&lt;c:set var="string2" value="\${fn:join(string1, '-')} " /><br>
&lt;c:out value="\${fn:join(string2, ' ')}" /></td></tr> I-am-a-test-String
<td><c:out value="\${string2}" /></td></tr>
</table></body></html>
```

xmlString: "I <verb>am</verb> a test String"

theString: "I am a test String"

| 函数  | 结果  |
|---|---|
| fn:contains(theString, 'test')  | true  |
| fn:endsWith(theString, 'ing')   | true  |
| fn:escapeXml(xmlString)   | I &lt;verb&gt;am&lt;/verb&gt; a test String |
| fn:indexOf(theString, 'am')   | 2   |
| fn:length(theString)  | 18  |
| fn:replace(theString, 'test', 'second')   | I am a second String                        |
| fn:startsWith(theString, 'ing')   | false                                       |
| fn:substring(theString, 2, 4)   | am  |
| fn:substringAfter(theString, 'am')  | a test String                               |
| fn:substringBefore(theString, 'am')   | I   |
| fn:toLowerCase(theString)   | i am a test string                          |
| fn:toUpperCase(theString)   | I AM A TEST STRING                          |
| fn:trim(' abc de ')   | abc de                                      |
| <c:set var="string1" value="\${fn:split<br>(theString, ' ')}" /><br><c:set var="string2" value="\${fn:join<br>(string1, '-')} " /><br><c:out value="\${string2}" /> | I-am-a-test-String                          |

# 自定义标签

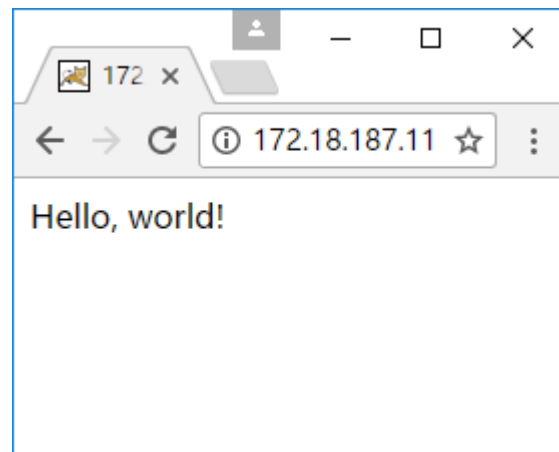
- 例子1、用标签定义替换标签

WEB-INF\tags\hello.tag

```
<table border="1">  
  <tr><td><%= "Hello! World!" %></td></tr>  
</table>
```

mytagTest.jsp

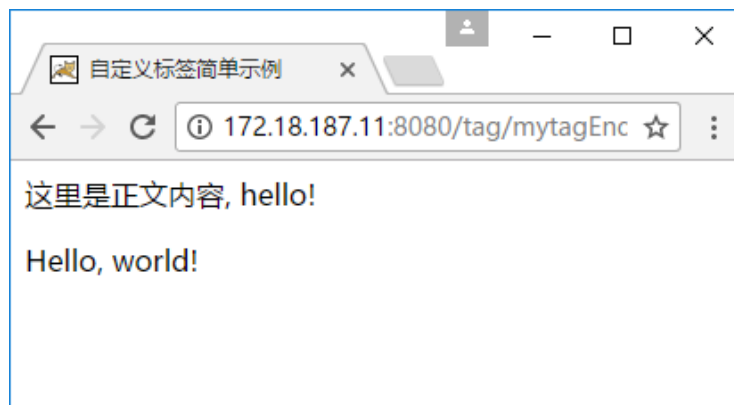
```
<%@ page language="java" import="java.util.*"  
      contentType="text/html; charset=utf-8"%>  
<%@ taglib prefix="mytagf" tagdir="/WEB-INF/tags"%>  
<html>  
  <body>  
    <mytagf:hello/>  
  </body>  
</html>
```



- 例子2—结束标签事件

mytagEnd.jsp

```
<%@ page language="java" contentType="text/html; charset=gb2312"%>
<%@ taglib uri="/WEB-INF/tlds/info.tld" prefix="ex" %>
<html>
<head>
    <title>自定义标签简单示例</title>
</head>
<body>
    <p>这里是正文内容, hello!</p>
    <ex:hello/>
</body>
</html>
```

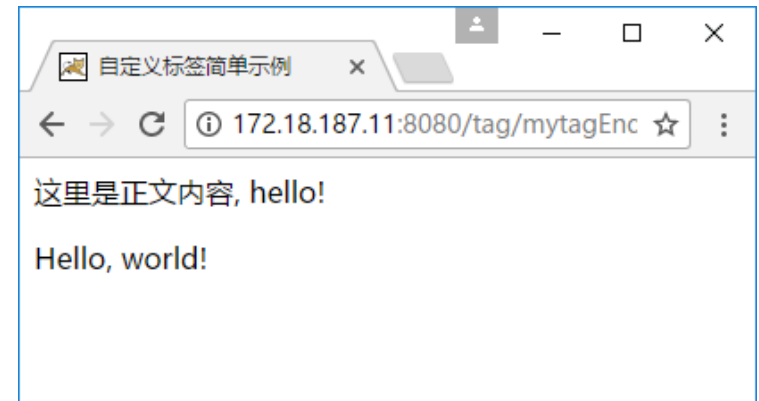


WEB-INF\tlds\info.tld (可以任意放置, 在引用程序中指出具体位置)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC
    "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
    "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<taglib>
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>
  <shortname>tld example</shortname>
  <tag>
    <name>hello</name>
    <tagclass>com.group.Info</tagclass>
    <bodycontent>empty</bodycontent>
    <attribute/>
  </tag>
</taglib>
```

WEB-INF\classes\com\group\info.java

```
package com.group;
import java.io.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
public class Info extends TagSupport{
    public int doEndTag(){
        try {
            String test = "Hello,world!";
            pageContext.getOut().println(test);
        }
        catch(IOException e){
        }
        return EVAL_PAGE;
    }
}
```

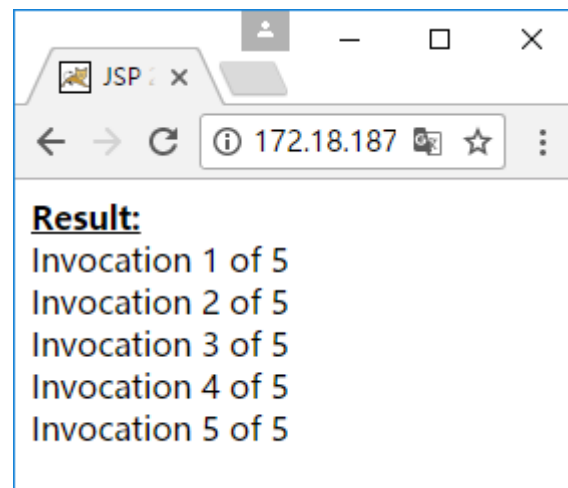


javac -classpath .\servlet-api.jar;.\jsp-api.jar Info.java

- 例子3—[重复标签](#)

tagRepeat.jsp

```
<%@ taglib prefix="mytag"
        uri="http://tomcat.apache.org/repeat-simple-taglib" %>
<html>
  <head>
    <title>JSP 2.0 Examples - Repeat SimpleTag Handler</title>
  </head>
  <body>
    <h1>JSP 2.0 Examples - Repeat SimpleTag Handler</h1>
    <b><u>Result:</u></b><br>
    <mytag:repeat num="5">
      Invocation ${count} of 5<br>
    </mytag:repeat>
  </body>
</html>
```





/WEB-INF/web.xml

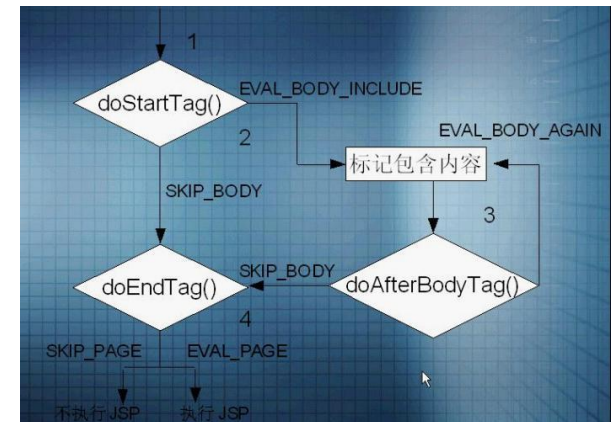
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1"
  metadata-complete="true">
  <description>
    Servlet and JSP Examples.
  </description>
  <display-name>Servlet and JSP Examples</display-name>
  <jsp-config>
    .....
    <taglib>
      <taglib-uri>
        http://tomcat.apache.org/repeat-simple-taglib
      </taglib-uri>
      <taglib-location>
        /WEB-INF/tags/simple-tags.tld
      </taglib-location>
    </taglib>
  </jsp-config>
</web-app>
```

/WEB-INF/tags/simple-tags.tld

```
<?xml version="1.0" encoding="UTF-8" ?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
  version="2.0">
  <description>A tag library exercising SimpleTag handlers.</description>
  <tlib-version>1.0</tlib-version>
  <short-name>SimpleTagLibrary</short-name>
  <uri>http://tomcat.apache.org/simple-taglib</uri>
  .....
  <tag>
    <description>Repeats the body of the tag 'num' times</description>
    <name>repeat</name>
    <tag-class>com.group.MyRepeatSimpleTag</tag-class>
    <body-content>scriptless</body-content>
    <variable>
      <description>Current invocation count (1 to num)</description>
      <name-given>count</name-given>
    </variable>
    <attribute>
      <name>num</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
    </attribute>
  </tag>
</taglib>
```

/WEB-INF/classes/com/group/MyRepeatSimpleTag.class

```
package com.group;
import java.io.IOException;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.SimpleTagSupport;
/**
 * SimpleTag handler that accepts a num attribute and
 * invokes its body 'num' times.
 */
public class MyRepeatSimpleTag extends SimpleTagSupport {
    private int num;
    @Override
    public void doTag() throws JspException, IOException {
        for (int i=0; i<num; i++) {
            getJspContext().setAttribute("count",
                String.valueOf( i + 1 ) );
            getJspBody().invoke(null);
        }
    }
    public void setNum(int num) {
        this.num = num;
    }
}
```



<http://blog.csdn.net/zljjava/article/details/17420809>

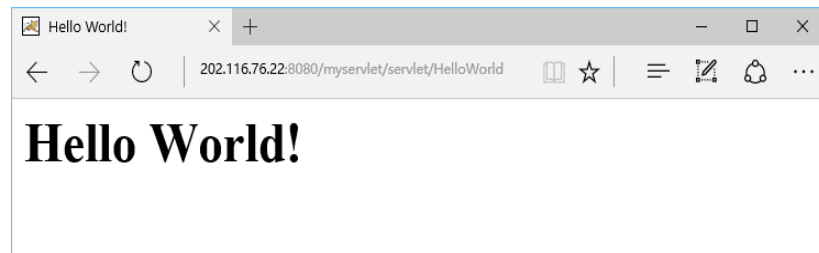
# Servlet程序

## 例子1

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

/WEB-INF/classes/HelloWorld.class

参考: tomcat examples



```
javac -classpath .\jar\servlet-api.jar HelloWorld.java
```

/WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app>
  <description>
    Servlet and JSP Examples.
  </description>
  <display-name>Servlet and JSP Examples</display-name>
  <servlet>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>HelloWorld</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/servlet/Hello</url-pattern>
  </servlet-mapping>
</web-app>
```

[Hello](#)

## 例子2

```
package serv
import java.io.*;                /WEB-INF/classes/serv/RequestInfo.class
import javax.servlet.*;
import javax.servlet.http.*;
public class RequestInfo extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Request Information Example</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h3>Request Information Example</h3>");
        out.println("Method: " + request.getMethod());
        out.println("Request URI: " + request.getRequestURI());
        out.println("Protocol: " + request.getProtocol());
        out.println("PathInfo: " + request.getPathInfo());
        out.println("Remote Address: " + request.getRemoteAddr());
        out.println("</body>");
        out.println("</html>");
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        doGet(request, response);
    }
}
```

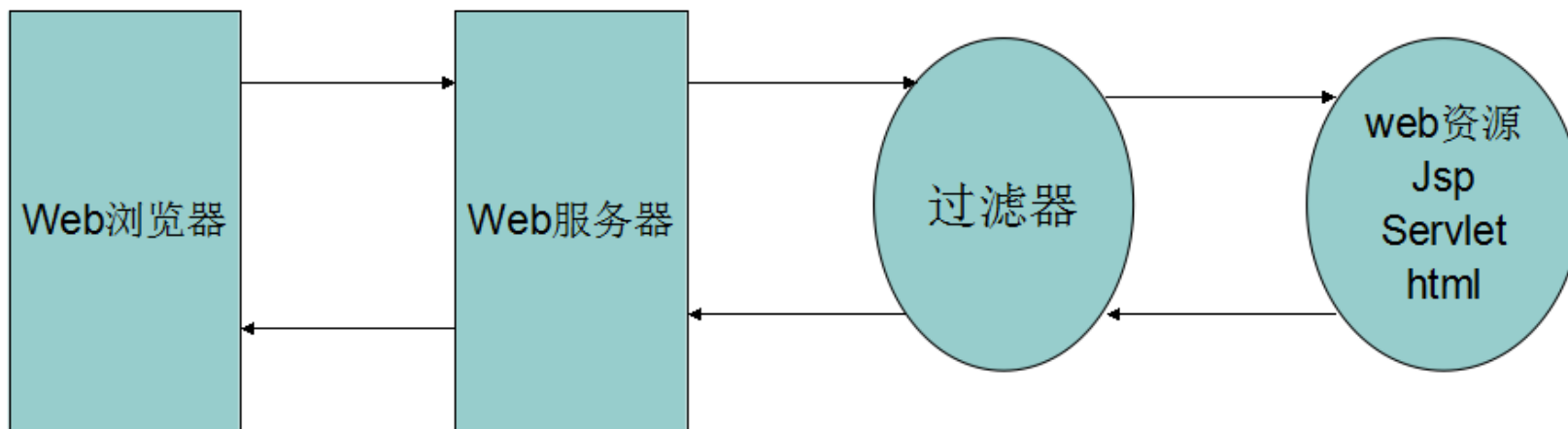
/WEB-INF/web.xml

```
<servlet>
  <servlet-name>Requ</servlet-name>
  <servlet-class>serv.RequestInfo</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Requ</servlet-name>
  <url-pattern>/servlet/request</url-pattern>
</servlet-mapping>
```

URL: <http://202.116.76.22:8080/myservlet/servlet/request>

# 过滤器

过滤器(Filter)技术可以在http请求被处理之前访问到该请求，并在http响应发出之前访问到该响应。通过链式过滤可以把http请求和http响应在多个过滤器之间传递。使用过滤器可以实现一些特殊的功能，例如：对JSP程序，Servlet程序、静态图片文件和静态html文件等进行拦截，并实现URL级别的权限访问控制、过滤敏感词汇、压缩响应信息等一些高级功能。



[参考 1 2](#)



/WEB-INF/classes/com/filter/FilterDemo01.class

```
package com.filter;
import java.io.IOException;          import javax.servlet.Filter;
import javax.servlet.FilterChain;    import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest; import javax.servlet.ServletResponse;
public class FilterDemo01 implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        this.URL = filterConfig.getInitParameter("URL");
        this.LocalURL = filterConfig.getInitParameter("LocalURL");
        System.out.println("URL:" + URL);
        System.out.println("LocalURL:" + LocalURL);
    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        //对request和response进行一些预处理
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        chain.doFilter(request, response); //让目标资源执行，放行
    }
    @Override
    public void destroy() {
        System.out.println("----过滤器销毁----");
    }
}
```

javac -classpath .\jar\servlet-api.jar FilterDemo01.java

/WEB-INF/web.xml

<!--配置过滤器-->

```
<filter>
    <filter-name>FilterDemo01</filter-name>
    <filter-class>com.FilterDemo01</filter-class>
    <init-param>
        <param-name>URL</param-name>
        <param-value>http://127.0.0.1:8081/uu/</param-value>
    </init-param>
    <init-param>
        <param-name>LocalURL</param-name>
        <param-value>http://127.0.0.1:8081/vv/</param-value>
    </init-param>
</filter>
```

<!--映射过滤器-->

```
<filter-mapping>
    <filter-name>FilterDemo01</filter-name>
    <!--“/*”表示拦截所有的请求 -->
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

# 附录1： 字符编码

- ASCII编码

ASCII(American Standard Code for Information )编码是单字节编码(0~255)主要用于显示现代英语和其他西欧语言。它等同于国际标准ISO/IEC 646。所有字符编码的0~127部分都和ASCII字符兼容。

| $D_3D_2D_1D_0$ \ $D_6D_5D_4$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0000                         | NUL | DLE | SP  | 0   | @   | P   | ^   | p   |
| 0001                         | SOH | DC1 | !   | 1   | A   | Q   | a   | q   |
| 0010                         | STX | DC2 | "   | 2   | B   | R   | b   | r   |
| 0011                         | ETX | DC3 | #   | 3   | C   | S   | c   | s   |
| 0100                         | EOT | DC4 | \$  | 4   | D   | T   | d   | t   |
| 0101                         | ENQ | NAK | %   | 5   | E   | U   | e   | u   |
| 0110                         | ACK | SYN | &   | 6   | F   | V   | f   | v   |
| 0111                         | BEL | ETB | '   | 7   | G   | W   | g   | w   |
| 1000                         | BS  | CAN | (   | 8   | H   | X   | h   | x   |
| 1001                         | HT  | EM  | )   | 9   | I   | Y   | i   | y   |
| 1010                         | LF  | SUB | *   | :   | J   | Z   | j   | z   |
| 1011                         | VT  | ESC | +   | ;   | K   | [   | k   | {   |
| 1100                         | FF  | FS  | ,   | <   | L   | \   | l   |     |
| 1101                         | CR  | GS  | -   | =   | M   | ]   | m   | }   |
| 1110                         | SO  | RS  | .   | >   | N   | ^   | n   | ~   |
| 1111                         | SI  | US  | /   | ?   | O   | _   | o   | DEL |

ASCII码 0~127

- ISO-8859-1编码

ISO-8859-1编码(Latin-1)是单字节编码，向下兼容ASCII，其编码范围是0x00~0xFF，其中0x00~0x7F完全和ASCII一致，0x80-0x9F之间是控制字符，0xA0-0xFF之间是文字符号。此字符集支持欧洲的二十二种语言，包括阿尔巴尼亚语、荷兰语、法罗语、德语、意大利语、拉丁语、卢森堡语、挪威语、葡萄牙语、西班牙语等。当转换一种未知编码时可以先按照这种编码读入，类似于读入字节流。

## • ANSI编码

ANSI(American National Standards Institute)编码采用1~2个字节进行对字符编码，并与ASCII编码兼容。

ANSI编码使用1个字节时，取值在0x00~0x7F之间，其表示的字符与ASCII码字符相同；采用2个字节时，第1个字节取值在0x80~0xFF之间。

在简体中文操作系统下ANSI编码是GB2312 编码，在繁体操作系统下ANSI编码是GBK码，在日文操作系统下 ANSI编码为JIS 编码。不同操作系统下的ANSI 编码会有重码，所以它们不能同时存在。

- 1981年发布的GB2312汉字编码国家标准，对汉字采用双字节编码，收录7445个图形字符，其中包括6763个汉字。
- 1984年实施的BIG5编码是台湾地区繁体中文标准字符集，采用双字节编码，共收录13053个中文字。
- 1995年发布的GBK编码是对GB2312编码的扩充，对汉字采用双字节编码，共收录21003个汉字，包含国家标准GB13000-1中的全部中日韩汉字，和BIG5编码中的所有汉字。
- 2000年发布的GB18030编码是对GBK编码的扩充，覆盖中文、日文、朝鲜语和中国少数民族文字，其中收录27484个汉字。GB18030字符集采用单字节、双字节和四字节三种方式对字符编码。兼容GBK和GB2312字符集。

GB2312采用区号和位号编码方式，下面的“答”字，区号为20（0x14），位号为80（0x50）。一个汉字的区号和位号分别加上0xA0，就是GB2312编码。  
“答”字的GB2312编码为0xB4F0。

20区

|    |   |   |   |   |   |   |   |   |   |   |
|----|---|---|---|---|---|---|---|---|---|---|
| 20 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0  |   | 础 | 储 | 矗 | 搐 | 触 | 处 | 揣 | 川 | 穿 |
| 1  | 橡 | 传 | 船 | 喘 | 串 | 疮 | 窗 | 幢 | 床 | 闯 |
| 2  | 创 | 吹 | 炊 | 捶 | 锤 | 垂 | 春 | 椿 | 醇 | 唇 |
| 3  | 淳 | 纯 | 蠢 | 戳 | 绰 | 疵 | 茨 | 磁 | 雌 | 辞 |
| 4  | 慈 | 瓷 | 词 | 此 | 刺 | 赐 | 次 | 聪 | 葱 | 囱 |
| 5  | 匆 | 从 | 丛 | 凑 | 粗 | 醋 | 簇 | 促 | 蹕 | 篡 |
| 6  | 窜 | 摧 | 崔 | 催 | 脆 | 瘁 | 粹 | 淬 | 翠 | 村 |
| 7  | 存 | 寸 | 磋 | 撮 | 搓 | 措 | 挫 | 错 | 搭 | 达 |
| 8  | 答 | 瘩 | 打 | 大 | 呆 | 歹 | 傣 | 戴 | 带 | 殆 |
| 9  | 代 | 贷 | 袋 | 待 | 逮 |   |   |   |   |   |

[GB2312编码参考](#)

- # Unicode编码

Unicode编码采用两个字节统一表示全世界的文字字符，其中，中、日、韩的三种文字占用了Unicode中0x3000到0x9FFF的部分，只能表示常用的七千多个汉字，不区分简体和繁体字。Windows和Java内部的字符串表示都是采用Unicode编码。包含'\0'的均为Unicode编码。

- # UTF-8编码

UTF-8（8-bit Unicode Transformation Format）是一种可变长度的字符编码，采用1到6个字节编码Unicode字符，可以同时表示世界上所有字符，简体汉字繁体汉字日文韩文等可以同时出现，国际标准为RFC 3629。所以，浏览器采用UTF-8编码时可以同时显示所有文字，包括简繁体字。

| Unicode符号范围 (十六进制)  | UTF-8编码方式（二进制）                      |
|---------------------|-------------------------------------|
| 0000 0000-0000 007F | 0xxxxxxx                            |
| 0000 0080-0000 07FF | 110xxxxx 10xxxxxx                   |
| 0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx          |
| 0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx |

Unicode码与UTF-8码的对应关系

# 附录2、编码转换

- 概述

转换编码主要是用来把字符编码转换成一种可以用ASCII可显示字符来表示的编码。URL编码主要用于在网页之间传递参数，Base64和Quoted-Printable主要用于邮件编码。

- URL编码

URL的参数和提交的数据都是键值对组成，每个键值对采用&连接：

$name_1=value_1 \& name_2=value_2 \& \dots \& name_n=value_n$

为了这些键值对可以显示出来，需要对它们进行URL编码。通过URL编码，可以把字符编码中大值字节(128~255)采用百分号加上两个十六进制字母表示。例如：汉字“胡”的GB2312编码为0xBAFA，它的URL编码是“%BA%FA”。

对于name和value中包括的=&\%等特殊字符，也要使用其ASCII码进行编码，比如“\”的ASCII码字节为0x5C，它的URL编码为三个字节“%5C”。

[js urlencode, encodeURIComponent](#)

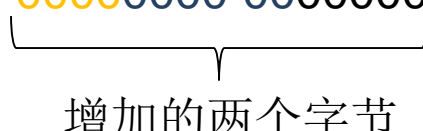
## • Base64编码

Base64编码要求把3个8位字节 ( $3 \times 8 = 24$ ) 转化为4个6位的字节 ( $4 \times 6 = 24$ )，之后在6位的前面补两个0，形成8位一个字节的形式。如果剩下的字符不足3个字节，则用'\0'填充，输出字符使用'='，因此编码后输出的文本末尾可能会出现1或2个'='。

字符串"1235" 需要补充两个字节"1235\0\0" (总字节数要求为3个字节的倍数)

ASCII的十六进制表示: 313233350000 (汉字编码为GB2312或UTF-8)

二进制表示:    00110001 00110010 00110011 00110101 00000000 00000000



增加的两个字节



|          |          |          |          |          |          |        |        |
|----------|----------|----------|----------|----------|----------|--------|--------|
| 1        | 2        | 3        | 5        | \0       | \0       |        |        |
| 00110001 | 00110010 | 00110011 | 00110101 | 00000000 | 00000000 |        |        |
| 001100   | 010011   | 001000   | 110011   | 001101   | 010000   | 000000 | 000000 |
| 12       | 19       | 8        | 51       | 13       | 16       | 0      | 0      |
| M        | T        | I        | z        | N        | Q        | =      | =      |

| 码值 | 字符 | 码值 | 字符 | 码值 | 字符 | 码值 | 字符 |
|----|----|----|----|----|----|----|----|
| 0  | A  | 16 | Q  | 32 | g  | 48 | w  |
| 1  | B  | 17 | R  | 33 | h  | 49 | x  |
| 2  | C  | 18 | S  | 34 | i  | 50 | y  |
| 3  | D  | 19 | T  | 35 | j  | 51 | z  |
| 4  | E  | 20 | U  | 36 | k  | 52 | 0  |
| 5  | F  | 21 | V  | 37 | l  | 53 | 1  |
| 6  | G  | 22 | W  | 38 | m  | 54 | 2  |
| 7  | H  | 23 | X  | 39 | n  | 55 | 3  |
| 8  | I  | 24 | Y  | 40 | o  | 56 | 4  |
| 9  | J  | 25 | Z  | 41 | p  | 57 | 5  |
| 10 | K  | 26 | a  | 42 | q  | 58 | 6  |
| 11 | L  | 27 | b  | 43 | r  | 59 | 7  |
| 12 | M  | 28 | c  | 44 | s  | 60 | 8  |
| 13 | N  | 29 | d  | 45 | t  | 61 | 9  |
| 14 | O  | 30 | e  | 46 | u  | 62 | +  |
| 15 | P  | 31 | f  | 47 | v  | 63 | /  |

"1235" => "1235\0\0"  
=> MTIzNQ==


参考


## • Quoted-Printable编码

如果输入数据可打印字符编码(33~60、62~126)，则直接输出；其它的需编码为“=”加两个字节的十六进制码(大写)。

Subject:=?gbk?**Q?**09374023=5E=AD=AF=B6=D8=B4=A8=5F=D7=F7=D2=B51?=

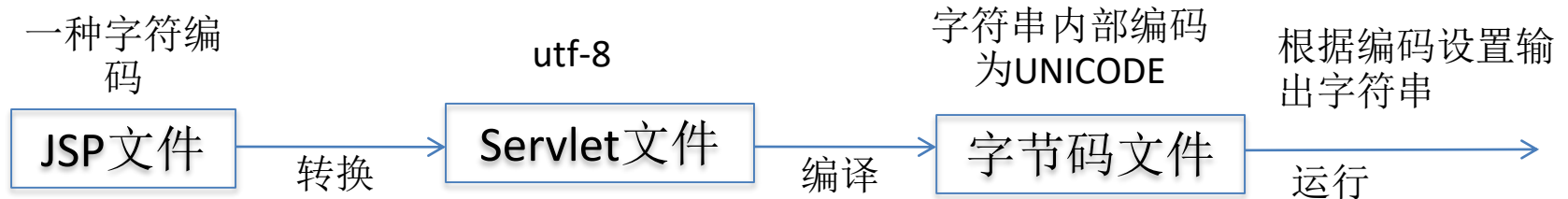
其中，09374023=5E=AD=AF=B6=D8=B4=A8=5F=D7=F7=D2=B51为Quoted-Printable编码。

10101011  =AD

张3  11010101 11000101 00110011 (GB2312编码)  
=D5 =C5 3

\* BASE64编码和Quoted-Printable编码主要用于邮件编码

# 附录3、 JSP程序的编码问题



```
<%@ page pageEncoding="GB2312"
        contentType="text/html; charset=GB2312"%>
```

- `pageEncoding`指出本JSP文件所用编码，`contentType`指出要生成响应的文件类型和格式以及所用编码，这里是网页文件及其编码。没有`pageEncoding`时使用`charset`指出JSP文件的编码。`pageEncoding` 和`charset`同时出现时必须一致，同时缺失时使用默认编码ISO-8859-1。
- 转换得到的`servlet`文件均采用`utf-8`编码，执行字节码文件时字符串的内部编码为UNICODE，输出到http响应时根据下面语句(由`page`指令得到)转换编码：  
`response.setContentType("text/html; charset=GB2312");`
- 提交的输入值采用与网页相同的编码，还要进行URL编码。用`request.getParameter()`取出提交值前系统会自动进行URL解码，得到的字符串默认采用ISO-8859-1编码，除非预先设置编码：  
`request.setCharacterEncoding("GB2312");`

[参考](#)

- 如果要把字符串转换到正确编码，方法如下：

```
String s = request.getParameter("name");  
byte[] bs = s.getBytes("iso-8859-1");  
String s2 = new String(bs, "utf-8");
```

- 如果自己生成URL参数（包含汉字），则要用如下方法：

```
<%@ page contentType="text/html; charset=gb2312" %>  
<a href="ds.jsp?url=<%=java.net.URLEncoder.encode("这里是参数值",  
    "GB2312")%>">点击这里</a>
```

```
<%  
    str=request.getParameter("url");  
    str=java.net.URLDecoder.decode(str, "GB2312");  
    out.print(str);  
%>
```

# 附录4、核心JSTL标签

| Tag                                      | Description  |
|--|--|
| <c:out>                                  | 显示表达式的结果，类似的方式<%=>效果，但是可以使用简单的“.”符号来访问属性。                        |
| <c:set>                                  | 定义并设置一个变量值。  |
| <c:remove>                               | 删除一个变量。 <c:remove var="salary"/>                                 |
| <c:catch>                                | 捕捉错误。  |
| <c:if>                                   | if语句标签   |
| <c:choose><br><c:when><br><c:otherwise > | switch语句标签   |
| <c:import><br><c:param>                  | 把一个URL的内容取回到一个网页、一个字符串变量，或者一个读者（Reader）。用<c:param>往URL中加入参数。     |
| <c:forEach >                             | forEach语句标签。   |
| <c:forTokens>                            | 逐令牌循环。   |
| <c:redirect>                             | 重定向到一个新的URL。 <c:redirect url="http://202.116.76.22:8080/jsp/" /> |
| <c:url>                                  | 创建一个URL，可以用<c:param>加入参数。  |

核心标签库：<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

# 附录5、 XML JSTL标签

| Tag                                       | Description  |
|---|--|
| <x:out>                                   | Like <%= ... >, but for XPath expressions.   |
| <x:parse>                                 | Use to parse XML data specified either via an attribute or in the tag body.  |
| <x:set >                                  | Sets a variable to the value of an XPath expression.   |
| <x:if >                                   | Evaluates a test XPath expression and if it is true, it processes its body. If the test condition is false, the body is ignored.                           |
| <x:forEach>                               | To loop over nodes in an XML document.   |
| <x:choose><br><x:when ><br><x:otherwise > | Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>                          |
| <x:transform ><br><x:param >              | <x:transform > applies an XSL transformation on a XML document。 <x:param > is used along with the transform tag to set a parameter in the XSLT stylesheet。 |

XML标签库： <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

# 附录6、SQL标签

| Tag                 | Description   |
|---------------------|---|
| <sql:setDataSource> | Creates a simple DataSource suitable only for prototyping   |
| <sql:query>         | Executes the SQL query defined in its body or through the sql attribute.  |
| <sql:update>        | Executes the SQL update defined in its body or through the sql attribute.   |
| <sql:param>         | Sets a parameter in an SQL statement to the specified value.  |
| <sql:dateParam>     | Sets a parameter in an SQL statement to the specified java.util.Date value.   |
| <sql:transaction >  | Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction. |

SQL标签库：<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>

# 附录7、格式化JSTL标签

| Tag                   | Description  |
|-----------------------|--|
| <fmt:formatNumber>    | To render numerical value with specific precision or format.   |
| <fmt:parseNumber>     | 把字符串解析为数字、百分比和货币。  |
| <fmt:formatDate>      | Formats a date and/or time using the supplied styles and pattern   |
| <fmt:parseDate>       | Parses the string representation of a date and/or time   |
| <fmt:bundle>          | Loads a resource bundle to be used by its tag body.  |
| <fmt:setLocale>       | Stores the given locale in the locale configuration variable.  |
| <fmt:setBundle>       | Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable. |
| <fmt:timeZone>        | Specifies the time zone for any time formatting or parsing actions nested in its body.                   |
| <fmt:setTimeZone>     | Stores the given time zone in the time zone configuration variable                                       |
| <fmt:message>         | 显示一条国际化的消息.  |
| <fmt:requestEncoding> | 设置http请求的字符编码  |

格式标签库: <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>



# 附录8、JSTL 函数

| Function                | Description   |
|-------------------------|---|
| fn:contains()           | Tests if an input string contains the specified substring.  |
| fn:containsIgnoreCase() | Tests if an input string contains the specified substring in a case insensitive way.              |
| fn:endsWith()           | Tests if an input string ends with the specified suffix.  |
| fn:escapeXml()          | Escapes characters that could be interpreted as XML markup.                                       |
| fn:indexOf()            | Returns the index withing a string of the first occurrence of a specified substring.              |
| fn:join()               | Joins all elements of an array into a string.   |
| fn:length()             | Returns the number of items in a collection, or the number of characters in a string.             |
| fn:replace()            | Returns a string resulting from replacing in an input string all occurrences with a given string. |
| fn:split()              | Splits a string into an array of substrings.  |
| fn:startsWith()         | Tests if an input string starts with the specified prefix.  |
| fn:substring()          | Returns a subset of a string.   |
| fn:substringAfter()     | Returns a subset of a string following a specific substring.                                      |
| fn:substringBefore()    | Returns a subset of a string before a specific substring.   |
| fn:toLowerCase()        | Converts all of the characters of a string to lower case.   |
| fn:toUpperCase()        | Converts all of the characters of a string to upper case.   |
| fn:trim()               | Removes white spaces from both ends of a string.  |

JSTL函数库：<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

# 附录9、参考资料

<http://blog.csdn.net/wfgeqgeq/article/details/6951878>

<http://www.yiibai.com/jstl>

<http://www.runoob.com/jsp/jsp-expression-language.html>

<http://www.runoob.com/jsp/jsp-jstl.html>

[http://www.yiibai.com/jsp/jstl\\_core\\_foreach\\_tag.html](http://www.yiibai.com/jsp/jstl_core_foreach_tag.html)

<http://blog.csdn.net/xiazdong/article/details/6982491>

<http://www.cnblogs.com/kristain/articles/2177728.html>

<http://little-bill.iteye.com/blog/757378>