
JSP

一、JSP概述
1、什么是JSP
JSP和Servlet都是由SUN公司提供的动态Web资源开发技术
JSP看起来像一个HTML，但是JSP中可以书写Java代码
可以通过Java代码获取动态的数据。

JSP本质上就是一个Servlet！！
(JSP在第一次被访问时,会翻译为一个Servlet文件,之后Servlet再执行,做出响应)

静态Web资源：不管什么时间访问,不管什么人访问,看到的效果都是相同的（html/css/js...）
动态Web资源：可以根据不同的访问时间、访问条件显示不同的内容(比如购物车页面、订单页面。。。)（Servlet/JSP）

2、JSP执行过程
JSP在第一次被访问时，会翻译为一个Servlet程序(xxx.java--> xxx.class)；
接着Servlet程序会执行,在执行的过程中，主要是将JSP中包含的html内容通过out.write一行一行输出到浏览器上并显示；
如果jsp中包含Java代码，在翻译后的Servlet程序中会执行这些Java代码，将执行的结果在一行一行输出到浏览器并显示。

二、JSP语法
1、模版元素
模版元素就是写在JSP中的html内容
或者是除了JSP特有的内容以外的其他内容都是模版元素
模版元素在翻译后的Servlet中，是被out.write一行一行输出到浏览器,经过浏览器解析并显示在网页上。

2、JSP表达式
格式：<%= 表达式内容 %>
表达式内容可以是：比如 字符串常量、变量、表达式
作用：执行表达式内容，将结果输出在当前位置，最后再发送给浏览器。
<%= "Hello JSP..." %>
<% String name = "张三疯"; %>
<%= name %>
<%= "张三疯" %>
<%= 364+785 %>

3、JSP脚本片段
格式：<% 若干Java语句 %>
作用：在翻译后的Servlet中，将脚本片段中的语句复制粘贴到对应的位置执行。

4、JSP注释
JSP注释的格式:
<%-- 注释内容 --%>
注释不能嵌套!!
JSP注释在翻译时，直接被丢弃，不予翻译!!

5、JSP指令
-- 指令不会产生输出
-- 指令的作用是用来通知JSP解析引擎如何将JSP翻译为一个Servlet。

5.1.page指令
page指令用于声明JSP页面的基本属性信息
(比如当前JSP使用的开发语言、JSP文件使用的编码、导入的jar包等。。)
<%@page 若干属性声明...%>
page指令是必须存在的指令
可以放在JSP文件的任意位置
只是为了可读性考虑,一般放在第一行!

(1)language="java"
-- 可以省略，用于指定当前JSP文件使用的开发语言是Java(了解)

(2)import="java.util.Date"
-- 用于在JSP中引入所需要的类或者包.

```
<%@page import="java.util.Date,java.io.File" ...%>

(3)pageEncoding="utf-8"
-- 通知JSP解析引擎, 使用哪一个编码解析当前JSP文件
-- Eclipse开发工具也会根据这个编码来保存JSP文件
-- 这个编码可以防止JSP在响应数据时出现乱码!!
-- 也可以防止JSP在翻译为Servlet时出现乱码!!
```

5.2.include指令

include指令用于实现页面的包含效果

```
<!-- 将头部包含进来 -->
<%@include file="/include/_header.jsp" %>

<!-- 将尾部包含进来 -->
<%@include file="/include/_footer.jsp" %>
```

5.3.taglib指令

在JSP中用于引入标签库文件

引入之后就可以使用标签库中的标签了

三、EL表达式

为什么要在JSP中使用标签??

JSP中如果写入了大量的Java代码, 会导致页面结构的混乱, 不利于后期的维护, 也无法实现代码的复用!!

因此SUN公司提出, 在JSP2.0的版本, 推荐不要在JSP中写任何一行Java代码!!

SUN推荐使用标签技术替代JSP中的Java代码

格式: ${ 常量/变量/表达式 }

主要作用是: 从域中获取数据(pageContext域、request域、session域、application域)

(1)获取 常量、变量（必先存入域中）、表达式的值

(2)获取域中的数组或集合中的数据

(3)获取域中的map集合中的数据

代码示例:

```
---------------------------------------
<h2>1.获取常量、变量（先存入域中）、表达式的值</h2>
<%-- 对于EL中的常量和表达式是直接计算结果,
        将结果输出到当前位置, 再发送给浏览器 --%>
${ "Hello EL" }
${ 384+496 }
<%
        //声明一个变量, 并且将变量存入到域中
        String name = "张无忌";
        request.setAttribute("name1", name);
%>
${ name1 }


<h2>2.获取域中数组或集合中的数据</h2>
<%
        //声明一个string数组,将数组存入域中
        String[] names =
                {"王海涛", "齐雷", "陈子枢"};
        pageContext.setAttribute("names", names);
%>
${ names[0] }
${ names[1] }
${ names[2] }


<h2>3.获取域中map集合中的数据</h2>
<%
        //声明一个map集合
        Map map = new HashMap();
        map.put("name", "阿凡达");
        map.put("age", "18");
        map.put("nickname", "小达达");
```

```
        //将map集合存入域中
        request.setAttribute("map1", map);
%>
${ map1.name }
${ map1.get("name") }
${ map1["name"] }

${ map1.age }
${ map1.nickname }
---------------------------------------
```

四、JSTL标签

1、JSTL标签介绍

JSTL标签是为JavaWeb开发人员准备的一套标准通用的标签库
和EL配合使用可以取代JSP中的Java代码
使用JSTL标签库中的标签之前，需要确认是否导入JSTL的jar包以及在JSP中要引入JSTL库
(1)导入JSTL的jar包
(2)在JSP中引入JSTL库

2、<c:set>标签

c:set标签用于往四大域中添加属性

```
<c:set var="name" value="张飞" scope="request"/>
${ name }

<c:set var="name" value="赵云" scope="request"/>
${ name }
```

3、<c:if>标签

```
c:if标签用于实现简单的if...else结构
<c:if test="${ 378+459 > 888 }">yes</c:if>
<c:if test="${ !(378+459 > 888) }">no</c:if>
```

4、<c:forEach>标签

```
c:forEach是循环标签, 用于遍历域中的数组
        或集合(包括map集合),或者执行指定次数的循环

<h3>(1)遍历域中的数组或集合</h3>
<%
        //声明一个list集合并往集合中添加元素
        List list = new ArrayList();
        list.add("张三");
        list.add("李四");
        list.add("王五");
        list.add("赵六");
        //将集合存入域中
        request.setAttribute("list", list);
%>
<c:forEach items="${ list }" var="obj">
        ${ obj }<br/>
</c:forEach>

<h3>(2)遍历域中的Map集合</h3>
<%
        //声明一个map集合
        Map map = new HashMap();
        map.put("name", "阿凡达");
        map.put("age", "18");
        map.put("nickname", "小达达");
        //将map集合存入域中
        request.setAttribute("map1", map);
%>
<c:forEach items="${ map1 }" var="entry">
        <%--
        ${ entry.getKey() } : ${ entry.getValue() }<br/>
         --%>
        ${ entry.key } : ${ entry.value }
```

```
</c:forEach>


<h3>(3)遍历1~100之间的整数,
        将3的倍数的数值设置颜色为红色</h3>
<%-- for(int i=1;i<=100;i++){}
<c:forEach begin="1" end="100" var="i">
        <span
                ${ i%3==0 ? "style='color:red'" : "" }
        >${ i }</span>
</c:forEach>
 --%>
<c:forEach begin="1" end="100" var="i">
        <span
                <c:if test="${ i%3==0 }">style='color:red'</c:if>
        >${ i }</span>
</c:forEach>


<h3>(4)遍历1~100之间的整数,
        将3的倍数的数值设置颜色为红色,
        数值之间使用逗号分隔</h3>
<%--
        varStatus表示循环遍历信息的对象,
        上面常用属性有:
        (1)first:boolean值表示当前元素是否为第一个
        (2)last:boolean值表示当前元素是否为最后一个
        (3)count:int值, 表示当前元素是第几个
        (4)index:下标
 --%>
<c:forEach begin="1" end="100" var="i"
                        step="1" varStatus="status">
        <span
                ${ i%3==0 ? "style='color:red'" : "" }
        >${ i }</span>
        <c:if test="${ !status.last }">,</c:if>
</c:forEach>

<hr/><hr/>

<c:forEach begin="0" end="100" var="i"
                        varStatus="status">
        [${ status.index }] ${ i }

</c:forEach>
<hr/><hr/>
<c:forEach begin="1" end="10" var="i"
                        varStatus="status">
        [${ status.count-1 }] Hello JSP...<br/>
</c:forEach>
```