JSP

JSP 教程 | 菜鸟教程 (runoob.com)

JSP的介绍及作用

介绍

> *JSP的全称是Java Service Pages。即Java的服务器页面。*

作用

> *JSP的主要作用是代替Servlet程序回传Html页面的数据。因为Servlet程序回传Html页面数据是一件非常繁琐的事情。开发成本和维护成本都极高。*

创建与访问

1. JSP页面在web目录下创建。
2. JSP页面和Html页面一样，都是存放在web目录下。访问JSP页面也跟访问也跟访问Html页面一样。`http://ip:port/工程路径/b.jsp`。

JSP的本质

> *JSP页面本质上是一个Servlet 程序。*

当第一次访问 JSP页面的时候。Tomcat 服务器会把JSP页面翻译成为一个 java 源文件。并且对它进行编译成 为.class 字节码程序。当打开 java 源文件可以发现其里面的内容有 `public final calss 工程名.jsp extends org.apache.jasper.runtime.HttpJspBase`。

跟踪原代码发现，HttpJspBase 类，它是直接地继承了 HttpServlet 类。也就是说，JSP 翻译出来的 java 类，它间接了继 承了 HttpServlet 类。也就是说，翻译出来的是一个 Servlet 程序。`public abstract class HttpJspBase extends HttpServlet implements HttpJspPage{}`

总结可知，通过翻译的 java 源代码可以得到结果——JSP就是 Servlet 程序。也可以去观察翻译出来的 Servlet 程序的源代码，不难发现。其底层实现也是通过输出流把 html 页面数据回传给客户端。

JSP的三种语法

JSP头部的page指令

> *JSP的page指令可以修改JSP页面中一些重要的属性，或者行为。*

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

1. language 属性表示 JSP翻译后是什么语言文件。暂时只支持 java。
2. contentType 属性表示 JSP返回的数据类型是什么。也是源码中 response.setContentType()参数值 。
3. pageEncoding 属性表示当前 JSP页面文件本身的字符集。
4. import 属性跟 java 源代码中一样。用于导包，导类。

以下两个属性是给 out 输出流使用

autoFlush 属性设置当 out 输出流缓冲区满了之后，是否自动刷新冲级区。默认值是 true。

buffer 属性设置 out 缓冲区的大小。默认是 8k。

1. errorPage属性设置当JSP页面运行时出错，自动跳转去的错误页面路径。路径一般是以斜杠打头，它表示请求地址为**http://ip:port/工程路径/**，映射到代码的web目录
2. isErrorPage属性设置当前JSP页面是否是错误信息页面。默认是false。如果是true可以获取异常信息。
3. session属性设置访问当前JSP页面，是否会创建HttpSession对象。默认是true。
4. extends属性设置JSP翻译出来的Java类默认继承谁。

JSP中常用脚本

声明脚本（极少使用）

格式：`<%! 声明Java代码 %>`

作用：可以给JSP翻译出来的Java类定义属性和方法甚至是静态代码块、内部类等。

练习

1.声明类属性

```
<%!
    private Integer id;
    private String name;
    private static Map<String,Object> map;
%>
```

2.声明static静态代码块

```
<%!
    static{
        map = new HashMap<String,Object>();
        map.put("key1", "value1");
    }
%>
```

3.声明类方法

```
<%!
    public int abc(){
        return 12;
    }
%>
```

4.声明内部类

```
<%!
    public static class A {
    private Integer id = 12;
    private String abc = "abc";
    }
%>
```

表达式脚本（常用）

格式：`<%=表达式%>`

作用：JSP页面上输出数据。

特点

1. 所有的表达式脚本都会被翻译到_jspService() 方法中
2. 表达式脚本都会被翻译成为 out.print()输出到页面上
3. 由于表达式脚本翻译的内容都在*jspService() 方法中,所以*jspService()方法中的对象都可以直接使用。
4. 表达式脚本中的表达式不能以分号结束。

练习

输出整型

```
<%=12 %>
```

输出浮点型

```
<%=12.12 %>
```

输出字符串

```
<%="我是字符串" %>
```

输出对象

```
<%=map%>
<%=request.getParameter("username")%>
```

代码脚本

格式：`<% java 语句 %>`

作用：可以在 jsp 页面中，编写我们自己需要的功能（写的是 java 语句）。

特点

1. 代码脚本翻译之后都在_jspService 方法中
2. 代码脚本由于翻译到*jspService()方法中，所以在*jspService()方法中的现有对象都可以直接使用。
3. 还可以由多个代码脚本块组合完成一个完整的 java 语句。
4. 代码脚本还可以和表达式脚本一起组合使用，在 jsp 页面上输出数据

练习：

代码脚本----if 语句

```
<%
    int i = 13;
    if( i == 12){
%>
    <h1>true</h1>
<%
    }else{
%>
    <h1>false</h1>
<%
    }
%>
<br>
```

代码脚本----for 循环语句

```jsp
<table>
<%
    for(int j = 0; j < 10; j++){
%>
    <tr>
        <td>第<%=j + 1%>行</td>
    </tr>
<%
    }
%>
</table>
```

翻译后 java 文件中_jspService 方法内的代码都可以写

```jsp
<%
    String username = requset.getParameter("username");
    System.out.println("用户名的请求参数值是: " + username);
%>

```

JSP中的三种注释

html注释

`<!--这是html注释-->`

html 注释会被翻译到 java 源代码中。在_jspService 方法里，以 out.writer 输出到客户端。

java 注释

`// 单行 java 注释`

`/* 多行 java 注释 */`

java 注释会被翻译到 java 源代码中。

jsp 注释

```
<%-- 这是 jsp 注释 --%>
```

jsp 注释可以注掉，jsp 页面中所有代码

JSP中out.print()、out.println()以及out.write()的区别

out.print()和out.write()

print()和println()是JspWriter类中定义的方法，write()则是Writer类中定义的。

print()和println()方法可将各种类型的数据转换成字符串的形式输出，而write()方法只能输出字符、字符数组和字符串等与字符相关的数据。

如果字符串对象的值为null，print()和println()方法将输出内容为“null”的字符串，而write()方法则是抛出NullPointerException异常。

out.print()和out.println()

println()虽然看似是换行，但转成网页之后，这种换行被认为是空格，所以输出的仍然是一行，用空格分隔，但右键点击页面查看源代码时，能看出换行起作用了。

所以在页面上需要换行的话，需要用`<br/>`。

JSP九大内置对象

request	请求对象
response	响应对象
pageContext	JSP的上下文对象
session	会话对象
application	ServletContext对象
config	ServletConfig对象
out	JSP输出流对象
page	指向当前JSP的对象

exception	异常对象

JSP四个域对象演示

四个域对象分别是：

1. pageContext (PageContextImpl 类)当前 jsp 页面范围内有效
2. request (HttpServletRequest 类) 一次请求内有效
3. session (HttpSession 类)一个会话范围内有效（打开浏览器访问服务器，直到关闭浏览器）
4. application (ServletContext 类)整个web工程范围内都有效（只要 web 工程不停止，数据都在）

域对象是可以像 Map 一样存取数据的对象。四个域对象功能一样。不同的是它们对数据的存取范围。 虽然四个域对象都可以存取数据。在使用上它们是有优先顺序的。 四个域在使用的时候，优先顺序分别是，他们从小到大的范围的顺序。 pageContext ---> request ---> session ---> application

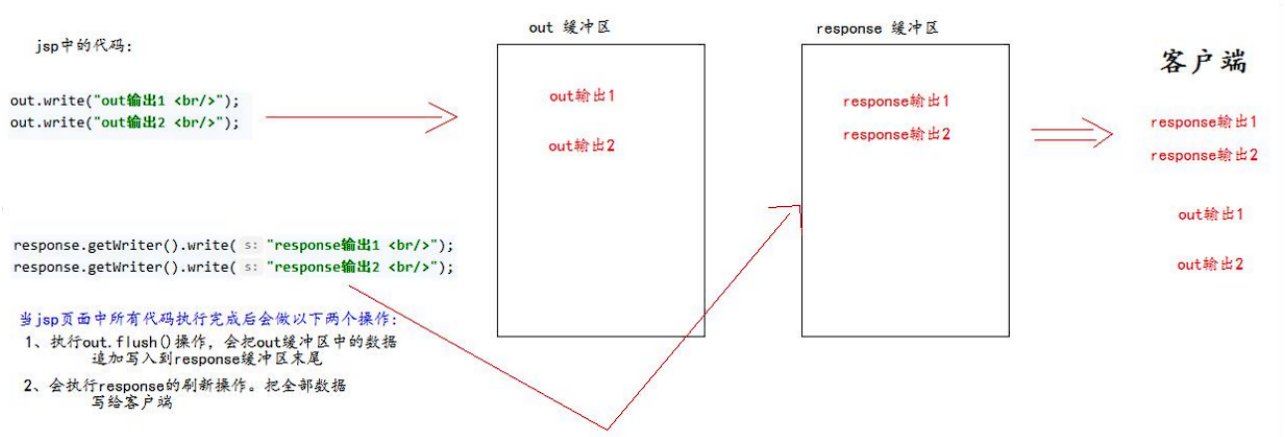
```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>scope.jsp页面</h1>
    <%
        //往四个域中都分别保存数据
        pageContext.setAttribute("key", "pageContext");
        request.setAttribute("key", "request");
        session.setAttribute("key", "session");
        application.setAttribute("key", "application");
    %>

    pageContext 域是否有值：<%=pageContext.getAttribute("key")%>
<br>
```

```
    request 域是否有值：<%=request.getAttribute("key")%> <br>
    session 域是否有值：<%=session.getAttribute("key")%> <br>
    application 域是否有值：<%=application.getAttribute("key")%>
<br>

    <%

 request.getRequestDispatcher("/scope2.jsp").forward(request,res
ponse);
    %>
</body>
</html>
```

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<html>
<head>
    <title>scope2.jsp</title>
</head>
<body>
    <h1>scope2.jsp 页面</h1>
    pageContext 域是否有值：<%=pageContext.getAttribute("key")%>
<br>
    request 域是否有值：<%=request.getAttribute("key")%> <br>
    session 域是否有值：<%=session.getAttribute("key")%> <br>
    application 域是否有值：<%=application.getAttribute("key")%>
<br>
</body>
</html>
```

JSP中out输出和response.getWriter输出的区别

response 中表示响应，我们经常用于设置返回给客户端的内容（输出） out 也是给用户做输出使用的。



由于 jsp 翻译之后，底层源代码都是使用 out 来进行输出，所以一般情况下。我们在 jsp 页面中统一使用 out 来进行输出。避免打乱页面输出内容的顺序。

out.write() 输出字符串没有问题

out.print() 输出任意数据都没有问题（都转换成为字符串后调用的 write 输出）

深入源码，浅出结论：在 jsp 页面中，可以统一使用 out.print()来进行输出。

JSP的常用标签

JSP静态包含

<%@ include file=""%> 就是静态包含

file 属性指定你要包含的 jsp 页面的路径

地址中第一个斜杠 / 表示为 http://ip:port/工程路径/ 映射到代码的 web 目录

静态包含的特点：

1、静态包含不会翻译被包含的 jsp 页面。

2、静态包含其实是把被包含的 jsp 页面的代码拷贝到包含的位置执行输出。

```
<%@ include file="/include/footer.jsp"%>
```

JSP动态包含

`<jsp:include page=""></jsp:include>` 这是动态包含

page 属性是指定你要包含的 jsp 页面的路径

动态包含也可以像静态包含一样。把被包含的内容执行输出到包含位置 动

态包含的特点：

1、动态包含会把包含的 jsp 页面也翻译成为 java 代码 2、动态包含底层代码使用如下代码去调用被包含的 jsp 页面执行输出。

`JspRuntimeLibrary.include(request, response, "/include/footer.jsp", out, false);`

3、动态包含，还可以传递参数.

JSP请求转发

`<jsp:forward page=""></jsp:forward>` 是请求转发标签，它的功能就是请求转发 page 属性设置请求转发的路径

`<jsp:forward page="/scope2.jsp"></jsp:forward>`

JSP练习题

在JSP页面中输出九九乘法表

```
<%@ page contectType="text/html";charset="UTF-8" language="java" %>
<html>
    <head>
        <title>Title</title>
        <style type="text/css">
            table{
                width:650px;
            }
```

```
        </style>
    </head>
    <body>
        <h1 align="center">九九乘法口诀表</h1>
        <table align="center">
            <% for(int i = 0; i <= 9; i++){ %>
                <tr>
                    <% for(int j = 1; j <= i; j++){
    <td><%=j + "x" + i + "=" + (i*j)%></td>
                <%  }%>
                </tr>
            <%  }%>
        </table>
    </body>
</html>
```

遍历输出**10**个学生信息到表格中

```
<%@ page import="java.util.List" %>
<%@ page import="com.atguigu.pojo.Student" %>
<%@ page import="java.util.ArrayList" %>
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<html>
<head>
    <title>Title</title>
    <style>
    table{
        border: 1px blue solid;
        width: 600px;
        border-collapse: collapse;
        }
        td,th{
        border: 1px blue solid;
        }
    </style>
</head>
<body>
<%--练习二：jsp 输出一个表格，里面有 10 个学生信息。--%>
```

```
21      <%
22      List<Student> studentList = (List<Student>)
    request.getAttribute("stuList");
23      %>
24      <table>
25          <tr>
26              <td>编号</td>
27              <td>姓名</td>
28              <td>年龄</td>
29              <td>电话</td>
30              <td>操作</td>
31          </tr>
32      <% for (Student student : studentList) { %>
33          <tr>
34              <td><%=student.getId()%></td>
35              <td><%=student.getName()%></td>
36              <td><%=student.getAge()%></td>
37              <td><%=student.getPhone()%></td>
38              <td>删除、修改</td>
39          </tr>
40      <% } %>
41      </table>
42  </body>
43  </html>
```

什么是Listener监听器

1、Listener 监听器它是 JavaWeb 的三大组件之一。JavaWeb 的三大组件分别是：Servlet 程序、Filter 过滤器、Listener 监 听器。

2、Listener 它是 JavaEE 的规范，就是接口

3、监听器的作用是，监听某种事物的变化。然后通过回调函数，反馈给客户（程序）去做一些相应的处理。

ServletContextListener监听器

ServletContextListener 它可以监听 ServletContext 对象的创建和销毁。

ServletContext 对象在 web 工程启动的时候创建，在 web 工程停止的时候销毁。

监听到创建和销毁之后都会分别调用 ServletContextListener 监听器的方法反馈。

两个方法分别是:

```
public interface ServletContextListener extends EventListener {
/**
* 在 ServletContext 对象创建之后马上调用，做初始化
*/
    public void contextInitialized(ServletContextEvent sce);
/**
* 在 ServletContext 对象销毁之后调用
*/
    public void contextDestroyed(ServletContextEvent sce);
}
```

如何使用 ServletContextListener 监听器监听 ServletContext 对象。

使用步骤如下：

1、编写一个类去实现 ServletContextListener

2、实现其两个回调方法

3、到 web.xml 中去配置监听

监听器实现类

```java
public class MyServletContextListenerImpl implements ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        System.out.println("ServletContext 对象被创建了");
    }
    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        System.out.println("ServletContext 对象被销毁了");
    }
}
```

web.xml中的配置

```xml
<!--配置监听器-->
<listener>
    <listener-class>com.atguigu.listener.MyServletContextListenerImpl</listener-class>
</listener>
```

返回文首