

JSP概念

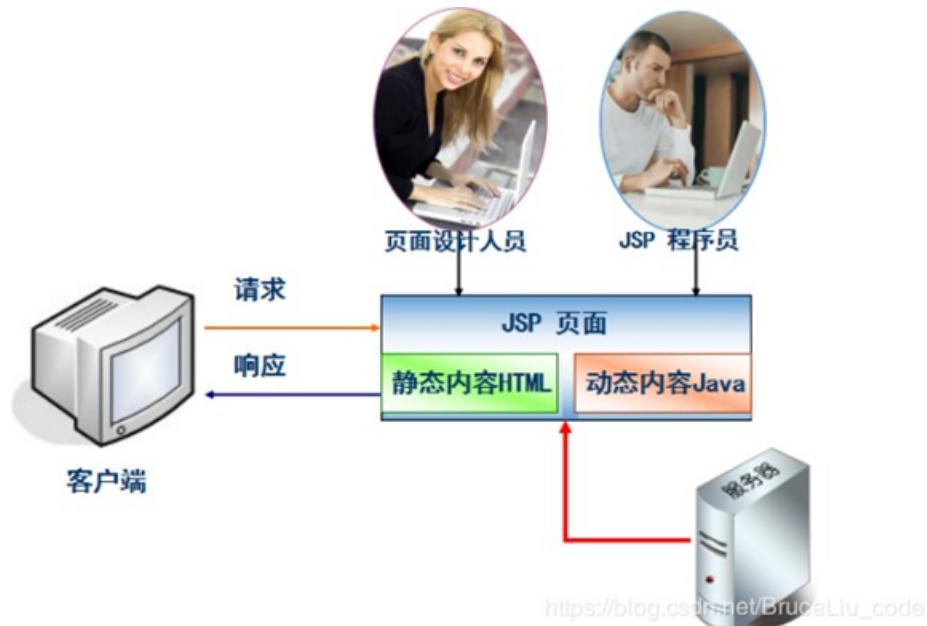
JSP (**java server pages**)，即java服务器页面，它是由Sun公司和其他很多个公司一起建立的一种动态网页技术；主要是用来代替Servlet来完成动态网页的输出。在JSP中可以书写html代码，还可以书写Java代码！JSP实质上是一个简化的servlet，是一种动态的网页技术的标准！

Servlet+JSP Servlet：获取数据、处理业务逻辑、查询数据等 JSP：页面展示

自我总结：servlet主要负责接收请求数据，，然后根据请求参数查询数据，最终通过响应对象向客户端响应动态的网页。Jsp**主要用来完成动态网页的输出**。

JSP引入目的

JSP性能好，可以在html页面中动态嵌入元素
服务器调用的是已经编译好的JSP文件
JSP基于Java Servlet Api，有很多强大企业的支持。
JSP可以与处理业务逻辑的Servlet 一起使用，该模式被Java Servlet 模版引擎所支持。

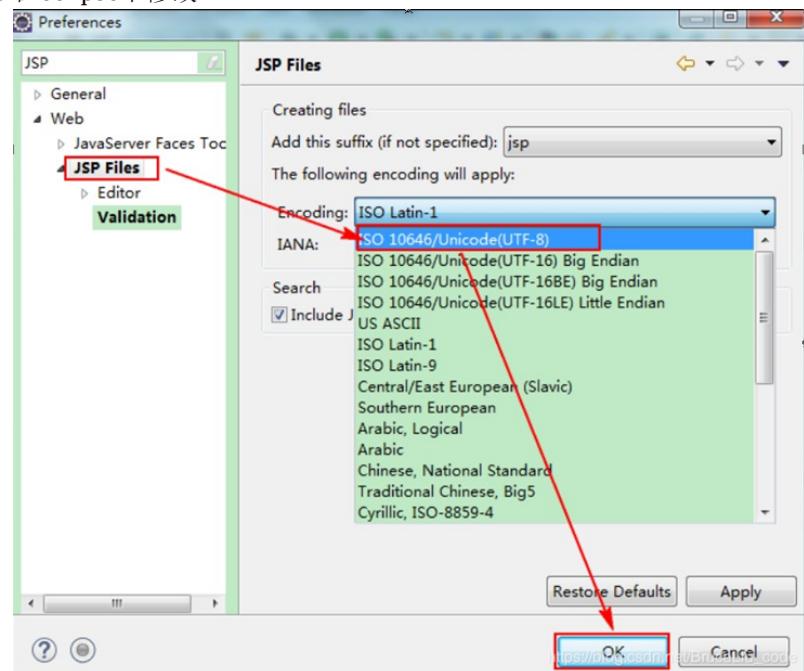


修改默认的JSP编码格式

JSP默认使用的是ISO-8859-1编码方式，无法书写中文，保存时报错！

修改方式：

1. <%@ page language="java" contentType="text/html; **charset=UTF-8**" pageEncoding="**UTF-8**"%>
在JSP页面中修改JSP的编码为"UTF-8"
2. 在eclipse中修改



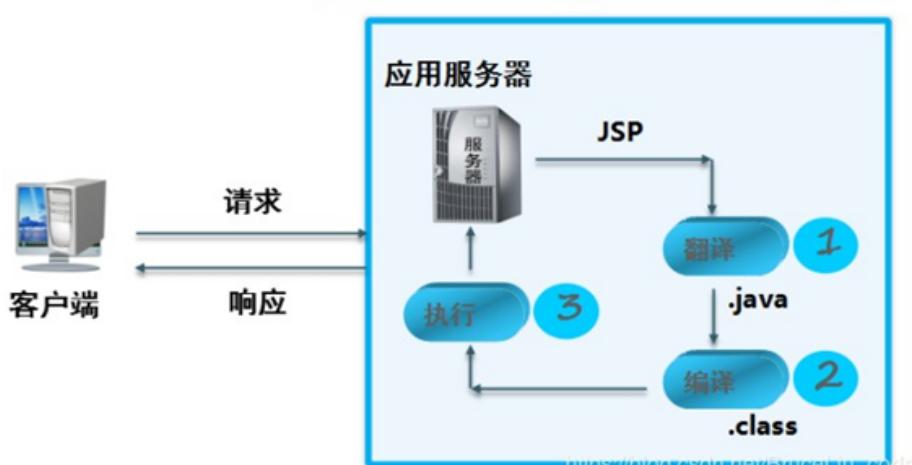
JSP中的运行原理（面试题）

Web容器处理JSP文件请求需要经过3个阶段

1. 翻译阶段

2. 编译阶段

3. 执行阶段



运行过程：

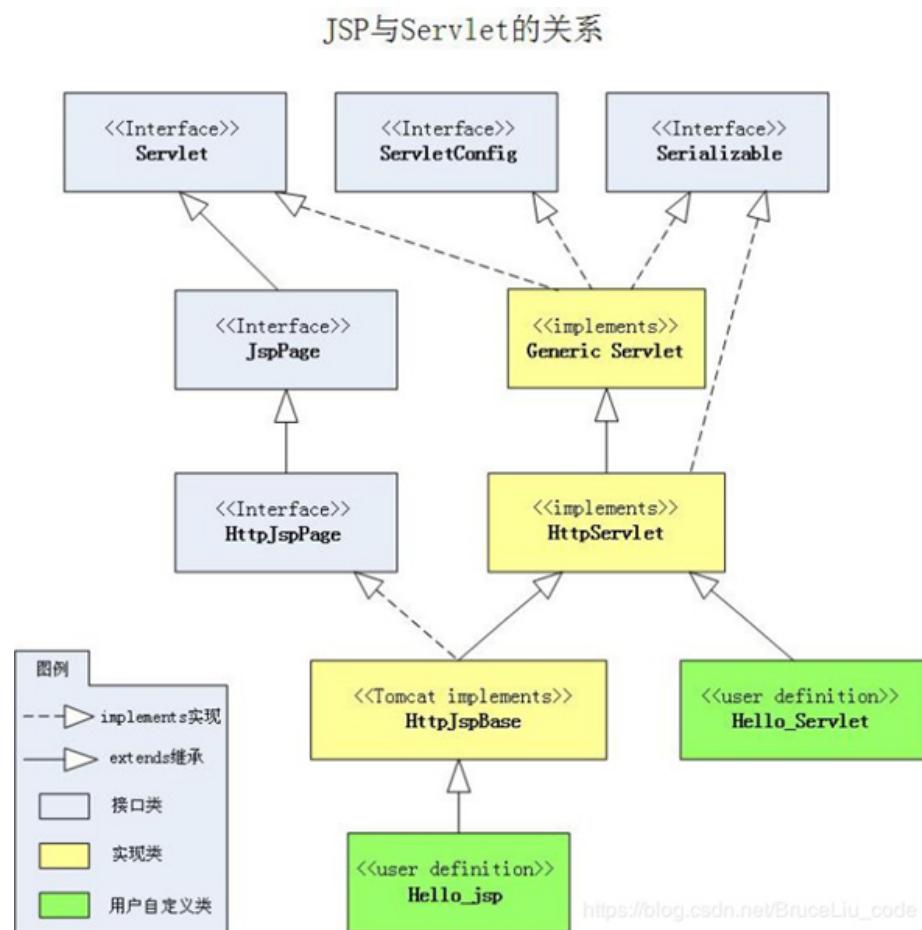
第一次访问JSP的时候，JSP会被翻译成.java的源文件，然后再被编译成.class的字节码文件，最后执行字节码文件，呈现运行结果。

第二次访问该JSP的时候，先去检测这个JSP内容有没有发生改变，如果内容有改变，那么将会执行翻译→编译→执行过程。如果JSP没有发生改变，那么将直接运行字节码文件，返回结果。

在照片中的目录：

> apache-tomcat-8.0.53 > work > Catalina > localhost > web0826 > org > apache > jsp
这个web0826是被部署到Tomcat中的项目
每个被部署到Tomcat中的项目都有对应这样一个路径下的jsp文件夹，在该文件夹中存放的是：jsp文件被翻译成的java源文件和对应的字节码文件
下有项目中jsp被翻译成java的源文件以及对应的字节码文件

JSP和Servlet的关系



JSP本质就是一个Servlet！只不过JSP侧重于显示数据，Servlet侧重于接收和响应数据！

JSP 迎宾服务员

Servlet 点菜服务员

JSP是Servlet技术的扩展，本质上是Servlet的简易方式，更强调应用的外表表达。JSP编译后是“类servlet”。Servlet和JSP最主要的不同点在于，Servlet的应用逻辑是在Java文件中，并且完全从表示层中的HTML里分离开来。JSP的情况是Java和HTML可以组合成一个扩展名为.jsp的文件。JSP侧重于页面显示，Servlet主要用于控制逻辑（功能）。

JSP中的3种脚本（了解）

JSP中可以书写java代码，来完成动态的html的拼接，相对于在servlet中拼接网页要简单很多！在JSP中书写java代码，称为java脚本！

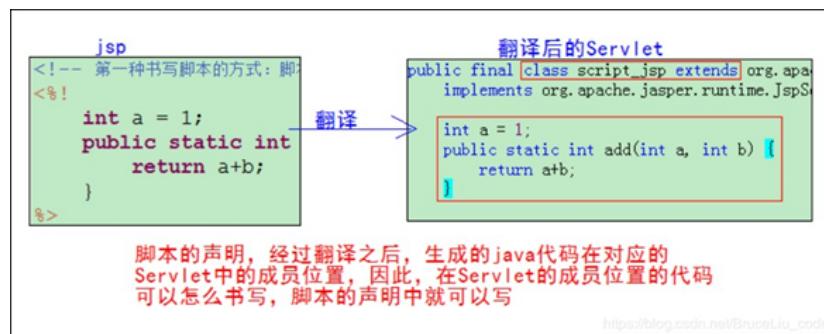
脚本的声明

语法格式：`<%! java代码 %>`

这个标签可以用来声明变量、方法、类、变量和方法是成员变量和成员方法。类是成员内部类。

用这种方式声明的方法是类的成员方法。

示例：



脚本的表达式

语法格式：`<%= 表达式 %>`

在这个标签中写的java代码将会出现在out()中向页面输出该标签中的代码一定不要加分号！

在脚本表达式中可以引用脚本声明中声明的东西。

脚本表达式相当于在Servlet的doGet或者是doPost方法中使用的代码response.getWriter().print()方法中的参数，所以脚本表达式最后不可以写分号。

示例：

JSP

```
<!-- 第二种脚本方式：脚本表达式 -->
a:<%=a %><br/>
2+1=<%=add(1,2) %>
body>
```

翻译 →

翻译后的Servlet

```
out.write("<head>\r\n");
out.write("<body>\r\n");
out.write("\t<!-- 第二种脚本方式 -->\r\n");
out.print(a);
out.print(add(1,2));
out.write("</body>\r\n");
out.write("</html>");
} catch (java.lang.Throwable e) {
    if (!e instanceof javax.servlet.ServletException) {
        e.printStackTrace();
    }
}
```

在脚本表达式中书写的表达式，翻译之后，对应Servlet的Service方法中out.print()方法中的参数
也就是说，方法中的参数可以怎么写，表达式就可以怎么写

https://blog.csdn.net/BruceLiu_000

脚本片段

语法格式：`<% java 代码片段1 %> <% java 代码片段2 %> <% java 代码片段3 %>`

最终这些片段组合在一起，必须是一段完整的java代码！在这个标签中编写的java代码、局部的JAVA代码，JSP翻译之后代码是在service方法中！
示例：

jsp

```
<!-- 脚本片段 -->
<%
    for (int i = 0; i < 10; i++) {
%>
<h3><%=i %></h3>
<% } %>
```

翻译 →

Servlet

```
out.write("\t");
for (int i = 0; i < 10; i++) {
    out.write("<h3>");
    out.print(i);
    out.write("</h3>");
}
out.write("\r\n");
```

jsp中的脚本片段，可以完成网页的动态的输出效果，可以加一些循环、判断等语句，来控制输出html代码

https://blog.csdn.net/BruceLiu_000

JSP中的3大指令

page指令

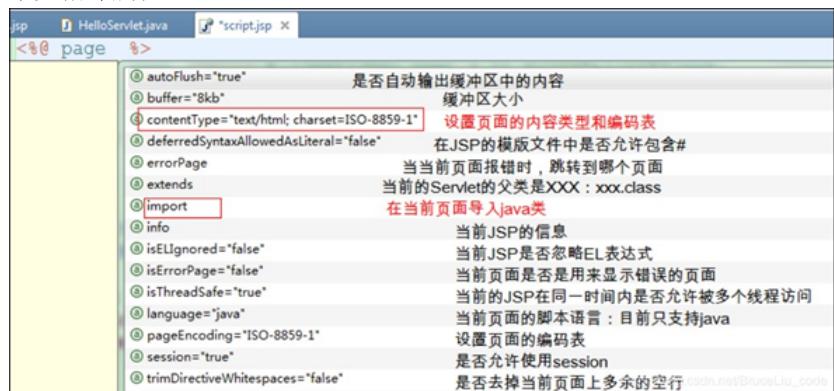
语法：`<%@ 指令名称 属性名称1="属性值1" 属性名称2="属性值2" %>`

作用：告诉JSP引擎如何使用jsp文件中的内容

示例：

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

常见指令属性:

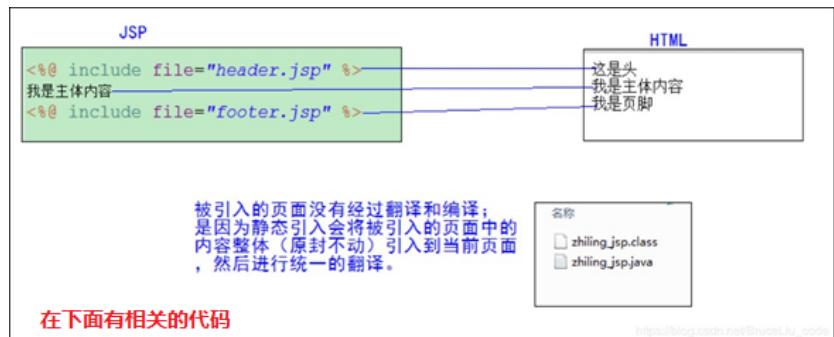


includ指令

语法格式: <%@ include file="header.jsp" %>

静态包含: 把其它资源包含到当前页面中

示例:



注意: 这种反收割hi, 称之为静态引入, 这种引入方法, 会将引入的页面中的所有内容原封不动的导入到当前的页面, 然后对整体进行统一的分析; 在引入的页面和被引入的页面中, 不能包含相同的变量名, 否则会报变量名重复的错误。

特点: 先合并, 后翻译。

示例代码:

1. index.jsp

```
<%@ page contentType="text/html; charset=UTF-8"
language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <%@ include file="top.jsp"%>
    <div>
        <h1>我是首页 <%=name%></h1>
    </div>
    <%@ include file="bottom.jsp"%>
</body>
</html>
```

2. top.jsp

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

<%
    String name="王二小";
%>
<div style="background-color: yellow">
    <h1>我是头部</h1>
</div>
```

3. bottom.jsp

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

<div style="background-color: green">
    <h1>我是尾部</h1>
</div>
```

taglib指令(会在JSTL标签库讲解到这指令！)

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

作用: 在JSP页面中导入JSTL标签库。替换jsp中的java代码片段。

JSTL标签库后面详细讲解，JSTL标签库，用来代替脚本片段完成循环和判断等语句