链滴

JSP 就是 Servlet

作者：howepeng

原文链接：https://ld246.com/article/1488332076000

来源网站：链滴

  自接触Java开发以来JSP在我的个人经验里一直扮演着重要的角色，一直以为能够熟练用JSP就是很好的掌握了它，
   但在一次面试过程当中被问及一个问题：“JSP 与 Servlet 的相同之处是什么”时，才发现自己对JSP工作的原理几乎一无所知。
   现在我将与大家分享经过学习和整理后我所理解的JSP和Servlet，
   本观点仅代表个人法，大家可在评论里就这个问题畅所欲言，以求大家共同进步。

 &nbs; **首先JSP是什么呢？**

   JSP是一种动页面技术 java server page，其根本是一个简化的Servlet设计。
   JSP诞生的目的就是了让前端代码和后端代码分离在不同的文件中。
   它实现了Html语法中的java扩展（以<%, %>形式），JSP与Servlet一样，在服务器端执行。
   通常返回给客户端的就一个HTML文本，因此客户端只要有浏览器就能浏览。
   我们使用JSP的主要目的就是表示逻辑从Servlet中分离出来，这样能够极大的简化开发。
   那么以上解释具体应该么理解呢？我们首先来看一个编译后的Servlet（.java）文件。
   jsp被变成了一个继承org.apache.jasper.runtime.HttpJspBase的类。
   查看jasper.jar里的HttpJspBase源就可以发现HttpJspBase继承自javax.servlet.http.HttpServlet，
   所以jsp就是一个servlet。

   Jsp文件

```
<%@ page language="java" import="java.util.*" pageEncoding="ISO-8859-1"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServe
Port()+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">

    <title>My JSP 'index.jsp' starting page</title>
    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->
  </head>

  <body>
    This is my JSP page.
  </body>
</html>
```

  Java文件

```
/*
 * Generated by the Jasper component of Apache Tomcat
 * Version: JspCServletContext/1.0
 * Generated at: 2017-02-24 03:08:52 UTC
 * Note: The last modified time of this file was set to
 *       the last modified time of the source file after
 *       generation to assist with modification tracking.
```

```
 */
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import java.util.*;

public final class index_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

  private static final javax.servlet.jsp.JspFactory _jspxFactory =
          javax.servlet.jsp.JspFactory.getDefaultFactory();

  private static java.util.Map&lt;java.lang.String,java.lang.Long&gt; _jspx_dependants;

  private javax.el.ExpressionFactory _el_expressionfactory;
  private org.apache.tomcat.InstanceManager _jsp_instancemanager;

  public java.util.Map&lt;java.lang.String,java.lang.Long&gt; getDependants() {
    return _jspx_dependants;
  }

  public void _jspInit() {
    _el_expressionfactory = _jspxFactory.getJspApplicationContext(getServletConfig().getServle
Context()).getExpressionFactory();
    _jsp_instancemanager = org.apache.jasper.runtime.InstanceManagerFactory.getInstanceMa
ager(getServletConfig());
  }

  public void _jspDestroy() {
  }

  public void _jspService(final javax.servlet.http.HttpServletRequest request, final javax.servlet.
ttp.HttpServletResponse response)
        throws java.io.IOException, javax.servlet.ServletException {

    final javax.servlet.jsp.PageContext pageContext;
    javax.servlet.http.HttpSession session = null;
    final javax.servlet.ServletContext application;
    final javax.servlet.ServletConfig config;
    javax.servlet.jsp.JspWriter out = null;
    final java.lang.Object page = this;
    javax.servlet.jsp.JspWriter _jspx_out = null;
    javax.servlet.jsp.PageContext _jspx_page_context = null;


    try {
      response.setContentType("text/html;charset=ISO-8859-1");
      pageContext = _jspxFactory.getPageContext(this, request, response,
                null, true, 8192, true);
      _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
      config = pageContext.getServletConfig();
```

```
      session = pageContext.getSession();
      out = pageContext.getOut();
      _jspx_out = out;

      out.write('\r');
      out.write('\n');

String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServe
Port()+path+"/";

      out.write("\r\n\r\n&lt;!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//E
\"&gt;\r\n&lt;html&gt;\r\n  &lt;head&gt;\r\n    &lt;base href=\"");
      out.print(basePath);
      out.write("\"&gt;\r\n    \r\n    &lt;title&gt;My JSP 'index.jsp' starting page&lt;/title&gt;\r\n\
&lt;meta http-equiv=\"pragma\" content=\"no-cache\"&gt;\r\n\t&lt;meta http-equiv=\"cach
-control\" content=\"no-cache\"&gt;\r\n\t&lt;meta http-equiv=\"expires\" content=\"0\"&gt;
  \r\n\t&lt;meta http-equiv=\"keywords\" content=\"keyword1,keyword2,keyword3\"&gt;\r\n\
&lt;meta http-equiv=\"description\" content=\"This is my page\"&gt;\r\n\t&lt;!--\r\n\t&lt;link
rel=\"stylesheet\" type=\"text/css\" href=\"styles.css\"&gt;\r\n\t--&gt;\r\n  &lt;/head&gt;\r\n
\r\n  &lt;body&gt;\r\n    This is my JSP page. &lt;br&gt;\r\n  &lt;/body&gt;\r\n&lt;/html&gt;\r
n");
    } catch (java.lang.Throwable t) {
      if (!(t instanceof javax.servlet.jsp.SkipPageException)){
        out = _jspx_out;
        if (out != null &amp;amp;&amp;amp; out.getBufferSize() != 0)
          try {
            if (response.isCommitted()) {
              out.flush();
            } else {
              out.clearBuffer();
            }
          } catch (java.io.IOException e) {}
        if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
        else throw new ServletException(t);
      }
    } finally {
      _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
```

<p> 大家可以看到，在这个文件里面有html代码，但这个文件是从.jsp文件转译成的.java文
，并不是手动编写的。
   也就是说我们写的JSP文件并不是直接运行的，而是先会转译成
java文件。
   做这个工作的正是JSP Container 即Tomcat服务器，它会将.jsp文件转译
.java文件，
   然后再将.java文件编译成.class文件，.class文件会处理请求，返回响应
最后生成页面返回给客户端显示。
   所以也可以理解为JSP文件就是Servlet文件，因为J
P文件是转换成Servlet文件才运行的。
   没有JSP的话，就需要把静态页面的代码手动
到Servlet文件里，这时候其实就是将网页逻辑和网页设计写在了一起，
   这样的开发
杂且效率比较低，所以JSP的出现就是将网页设计分离出来，简化开发。
   <br /
   既然是tomcat做的把jsp变为java的动作，那么java文件在哪呢?
   可以看看to
cat的以下路径,就会发现我们发布在tomcat上的jsp被tomcat转化后java文件了。
   to
cat\work\Catalina\localhost\工程名\org\apache\jsp\

   当然
omcat的jasper.jar里有个org.apache.jasper.JspC类。
   我们可以使用这个类手动来做j

p2java的动作。
   借助ant，我们就可以调用JspC这个类。
   执行jsp2jav
在outputDir设定的文件夹里就可以看到自己工程里jsp变成的java文件了。

```
<?xml version="1.0" encoding="GBK"?>
<project name="WebApp Precompilation JSP to Java to Class to Jar" basedir="." default="he
p">
 <target name="jsp2java">
  <taskdef classname="org.apache.jasper.JspC" name="jsp2java">
   <classpath id="jsp2java.classpath">
    <fileset dir="D:/tomcat/bin">
     <include name="*.jar"/>
    </fileset>
    <fileset dir="D:/tomcat/lib">
     <include name="*.jar"/>
    </fileset>
    <fileset dir="D:/workspace/SSMv1.0/WebRoot/WEB-INF/lib">
     <include name="*.jar"/>
    </fileset>
   </classpath>
  </taskdef>
  <!-- 注意JSP文件要设置为UTF-8编码 -->
  <jsp2java classpath="jsp2java.classpath" javaEncoding="UTF-8" validateXml="false"
  uriroot="D:/workspace/SSMv1.0/WebRoot"
  webXmlFragment="D:/workspace/SSMv1.0/WebRoot/WEB-INF/webJSP.xml"
  outputDir="D:/workspace/SSMv1.0/WebRoot/WEB-INF/JspC/src"/>
 </target>
</project>
```

  希望以上解释能对大家有所帮助！