出处不明：

JSP是Servlet的扩展，在没有JSP之前，就已经出现了Servlet技术。Servlet是利用输出流动态生成HTML页面，包括每一个HTML标签和每个在HTML页面中出现的内容。

由于包括大量的HTML标签、大量的静态文本及格式等，导致Servlet的开发效率极为低下。所有的表现逻辑，包括布局、色彩及图像等，都必须耦合在Java代码中，这的确让人不胜其烦。

JSP的出现弥补了这种不足，JSP通过在标准的HTML页面中插入Java代码，其静态的部分无须Java程序控制，只有那些需要从数据库读取并根据程序动态生成信息时，才使用Java脚本控制。

从表面上看，JSP页面已经不再需要Java类，似乎完全脱离了Java面向对象的特征。

事实上，JSP是Servlet的一种特殊形式，每个JSP页面就是一个Servlet实例——JSP页面由系统编译成Servlet，Servlet再负责响应用户请求。

JSP其实也是Servlet的一种简化，使用JSP时，其实还是使用Servlet，因为Web应用中的每个JSP页面都会由Servlet容器生成对应的Servlet。对于Tomcat而言，JSP页面生成的Servlet放在work路径对应的Web应用下。

看下面一个简单的JSP页面：

```
<!-- 表明此为一个JSP页面 -->
<%@ page contentType="text/html; charset=gb2312" language="java" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
    <HEAD>
        <TITLE>第一个JSP页面</TITLE>
    </HEAD>

    <BODY>
    <!-- 下面是Java脚本-->
        <%for(int i = 0 ; i < 10; i++)
```

```
            {
              out.println(i);
          %>
         <br>
          <%}
          %>
    </BODY>
</HTML>
```

当启动Tomcat之后，可以在Tomcat的Catalina/localhost/jsp/test/org/apache/jsp目录下找到如下文件(假如Web应用名为jsptest，上面JSP页的名为test1.jsp）：test1_jsp.java和test1_jsp.class。

这两个文件都是Tomcat生成的，Tomcat根据JSP页面生成对应Servlet的Java文件及class文件。

下面是test1_jsp.java文件的源代码，这是一个特殊的Java类，是一个Servlet类：

//JSP页面经过Tomcat编译后默认的包(不同的servlet容器提供商生成的servlet文件是不同的)

```
package org.apache.jsp;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

//继承HttpJspBase类，该类其实是个HttpServlet的子类(jasper是tomcat的jsp engine)
public final class test1_jsp extends org.apache.jasper.runtime.HttpJspBase
          implements org.apache.jasper.runtime.JspSourceDependent
    {
          private static java.util.Vector _jspx_dependants;
          public java.util.List getDependants() {
              return _jspx_dependants;
          }
          //用于响应用户的方法
```

```
public void _jspService(HttpServletRequest request,
   HttpServletResponse response)
   throws java.io.IOException, ServletException
{   //built-in objects(variavles) are created here.
    //获得页面输出流
    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    //获得页面输出流
    JspWriter out = null; //not PrintWriter. JspWriter is buffered
defautly.
    Object page = this;
    JspWriter _jspx_out = null;
    PageContext _jspx_page_context = null;
    //开始生成响应
    try
    {
        _jspxFactory = JspFactory.getDefaultFactory();
       //设置输出的页面格式
   response.setContentType("text/html; charset=gb2312");
   pageContext = _jspxFactory.getPageContext(this, request,
   response, null, true, 8192, true);
   _jspx_page_context = pageContext;
   application = pageContext.getServletContext();
   config = pageContext.getServletConfig();
   session = pageContext.getSession();
   //页面输出流

  out = pageContext.getOut();
   _jspx_out = out;
   //输出流，开始输出页面文档
   out.write("rn");
   //下面输出HTML标签
```

```
    out.write("<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
    Transitional//EN">rn");
    out.write("<HTML>rn");
    out.write("<HEAD>rn");
    out.write("<TITLE>first Jsp</TITLE>rn");
    out.write("</HEAD>rn");
    out.write("<BODY>rn");
    //页面中的循环，在此处循环输出
    for(int i = 0 ; i < 10; i++)
    {
     out.println(i);
     out.write("rn");
     out.write("<br>rn");
    }
    out.write("rn");
    out.write("</BODY>rn");
    out.write("</HTML>rn");
    out.write("rn");
   }
   catch (Throwable t)
   {
    if (!(t instanceof SkipPageException))
    {
       out = _jspx_out;
       if (out != null && out.getBufferSize() != 0)
         out.clearBuffer();
       if (_jspx_page_context != null) _jspx_page_context.handle
       PageException(t);
    }
  }
   finally
 {
   if (_jspxFactory != null) _jspxFactory.releasePageContext(_jspx_
   page_context);
}
```

```
        }
}
```

对比test1.jsp和test1_jsp.java文件，可得到一个结论：该JSP页面中的每个字符都由test1_jsp.java文件的输出流生成。

根据上面的JSP页面工作原理图，可以得到如下四个结论：

— JSP文件必须在JSP服务器内运行。

— JSP文件必须生成Servlet才能执行。

— 每个JSP页面的第一个访问者速度很慢，因为必须等待JSP编译成Servlet。

— JSP页面的访问者无须安装任何客户端，甚至不需要可以运行Java的运行环境，因为JSP页面输送到客户端的是标准HTML页面。

JSP和Servlet会有如下转换：

- JSP页面的静态内容、JSP脚本都会转换成Servlet的xxxService()方法，类似于自行创建Servlet时service()方法。

- JSP声明部分，转换成Servlet的成员部分。所有JSP声明部分可以使用private,protected,public,static等修饰符，其他地方则不行。

- JSP的输出表达式(<%= ..%>部分)，输出表达式会转换成Servlet的xxxService()方法里的输出语句。

- 九个内置对象要么是xxxService()方法的形参，要么是该方法的局部变量，所以九个内置对象只能在JSP脚本和输出表达式中使用。// 不能在jsp Declaration中使用

<TOMCAT_HOME>/conf/web.xml这个文件，里面有这样一段

```
    <servlet>
        <servlet-name>jsp</servlet-name>
        <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
```

```
        <init-param>
            <param-name>fork</param-name>
            <param-value>false</param-value>
        </init-param>
        <init-param>
            <param-name>xpoweredBy</param-name>
            <param-value>false</param-value>
        </init-param>
        <load-on-startup>3</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>jsp</servlet-name>
        <url-pattern>*.jsp</url-pattern>
    </servlet-mapping>
```

然后再去看看org.apache.jasper.servlet.JspServlet这个类，跟着就会看到org.apache.jasper.servlet.JspServletWrapper这个类jsaper包下的类就是解析JSP代码的类了

有机会自己找一些例子来分析下