

1. JSP是动态网页。

静态和动态不是指页面是否是动态的，而是指适当随着时间，地点，用户操作的改变而改变，动态网页需要用到服务端脚本语言（JSP）

2. 架构

BS：例如：网页版京东，客户端通过各类浏览器直接访问服务端

CS：Client Server 服务器，通过客户端（每一台机器）到服务器。例如：QQ

JSP基于BS

3. 访问的网页

在WIN-INF中的web.xml中看，查找welcome，就可以看访问的是那个页面

```
<!-- pattern: /services/ nonblocking/ number: 1 -->
</servlet-mapping>
- <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.xhtml</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
<!-- Websocket examples -->
```

先访问第一个index.html这个网页，如果没有找到就访问第二个依次查找，可以通过改变这个来改变访问顺序

bapps > examples		↕	↺	🔍 搜索"exa
名称				修改E
jsp				2020,
servlets				2020,
WEB-INF				2020,
websocket				2020,
index.html				2019,

3. 文件夹的作用

bin: 可执行文件 (startuo.bat shutdown.bat)
 cont: 配置文件 (server.xml)
 lib: 日志文件 (记录出错等信息)
 temp: 临时文件
 webapps: 可执行项目 (将我们开发的项目放入该目录)
 work: 存放jsp翻译成的java, 以及编辑成的class文件
 jsp->java->class

4. 常见的状态码

200: 一切正常
 300/301: 页面重定向 (页面跳转)
 404: 资源不存在
 403: 权限不足 (如果访问a目录, 但是a目录设置不可见)
 500: 服务器内部错误 (代码有误)

6. 遇到404的解决办法

当访问自己页面时报404的解决办法可以加入自己做的页面名称,

例如：可行：<http://localhost:8888/JspProject/Index.jsp>//这是写死的情况

可行：localhost:8888/JspProject

不可行：<http://localhost:8888/JspProject/>//为什么不可以呢

可能是因为冲突了

```
<welcome-file-list>
```

```
<welcome-file>Index2.jsp</welcome-file>
```

```
</welcome-file-list>
```

```
</web-app>
```

一、查看自己的路径是否正确，正确的路径访问是：<http://localhost:8080/xxx/yyy.html>（注：xxx表示你的项目名。yyy表示你的页面名称，如果你的页面是jsp，就把.html换成.jsp）。

二、查看自己的html或者jsp是否放在了WEB-INF下面了，WEB-INF是受保护的不可以访问里面的东西。所以只要把页面移除来放在WebContent下面就可以啦。

三、tomcat7绑定到了IDE工具下的时候要修改一个配置,笔者当初就是这个原因。(注：笔者用的是eclipse)

1、找到tomcat服务器的所在位置

2、双击该服务器，会出现以下界面

2、把红色的框框里面的内容修改成笔者现在的样子

1)第一个框框默认选择一，那个服务器是IDE自动集成的不方便查找。当选择二，下面的Server path，就会出现我们自己的tomcat的安装地址了。

2)第二个框框是tomcat访问的工程名，默认访问webapps。

注：如果打开以上界面，上面的两个框框无法修改的话，请检查。

1、你的tomcat是否启动着，关闭它。

2、请先remove掉里面的工程。

7. 在项目/WEB-INF/web.xml中设置初始页面

```
<welcome-file-list>
```

```
<welcome-file>Index2.jsp</welcome-file>
```

```
</welcome-file-list>
```

8. 错误码500:

HTTP Status 500 – Internal Server Error

Type 异常报告

消息 Unable to compile class for JSP:

描述 服务器遇到一个意外的情况，阻止它完成请求。

Exception

org.apache.jasper.JasperException: Unable to compile class for JSP:

JSP文件: [/Index.jsp] 的第 [9] 行发生了一个错误

Syntax error, insert ";" to complete Statement

```
6:      <body>
7:
8:                  hello jsp...
9:                  <%
10:                                     out.print("hello world....")
11:                                     %>
12:      </body>
```

6. 配置虚拟路径

用途: 当项目放在了webapps以外的文件夹的时候, 该怎么修改让项目可以被访问到

a. 方式一:

) > 各种软件的安装包 > apache-tomcat-8.5.50 > conf

名称	修改日期	类型	大小
Catalina	2019/12/16 21:09	文件夹	
catalina.policy	2019/12/7 19:21	POLICY 文件	14 KB
catalina.properties	2019/12/7 19:21	PROPERTIES 文件	8 KB
context.xml	2019/12/7 19:21	XML 文档	2 KB
jaspic-providers.xml	2019/12/7 19:21	XML 文档	2 KB
jaspic-providers.xsd	2019/12/7 19:21	XML Schema File	3 KB
logging.properties	2019/12/16 21:15	PROPERTIES 文件	4 KB
server.xml	2019/12/17 9:49	XML 文档	8 KB

在这里面配置找到一个<host...//这里里面配置>

```
</Realm>
```

```
<Host name="localhost" appBase="webapps"
      unpackWARs="true" autoDeploy="true">
```

```
    <!-- SingleSignOn valve, share authentication between web applications
         Documentation at: /docs/config/valve.html -->
```

```
    <!--
    <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
    -->
```

```
    <!-- Access log processes all example.
         Documentation at: /docs/config/valve.html
         Note: The pattern used is equivalent to using pattern="common" -->
```

```
    <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
          prefix="localhost_access_log" suffix=".txt"
          pattern="%h %l %u %t &quot;%r&quot; %s %b" />
```

```
</Host>
```

```

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs
    prefix="localhost_access_log" suffix=".txt"
    pattern="%h %l %u %t &quot;%r&quot; %s %b" />

<Context docBase="" path="/JspProject" />

</Host>

```

webapps就是默认的一个虚拟路径

换一个虚拟路径只需要添加一段代码：

```

<Context docBase="" path="/JspProject" /> //严格区分大小写，这样来再配一个其他的虚拟路径，
</Host>

```

docBase是指项目的实际路径；

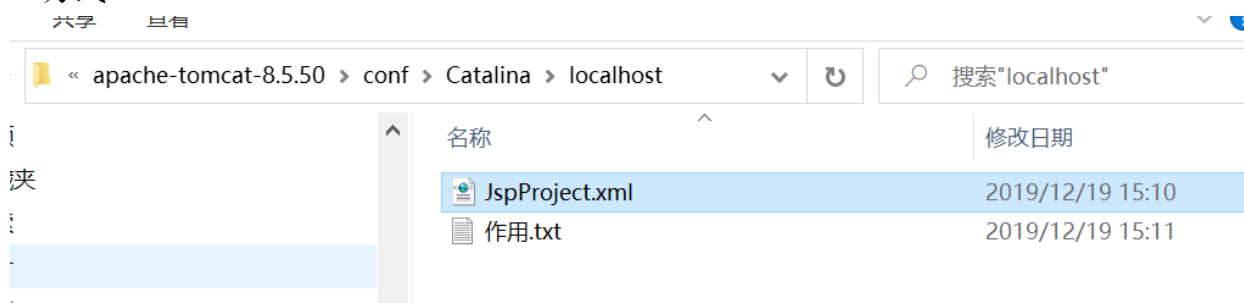
path是虚拟路径：要么写绝对路径，要么写相对路径【/JspProject】（相对于webapps）注意相对路径要写“/”

也可以写：【D:\各种软件的安装包\apache-tomcat-8.5.50\webapps\JspProject】

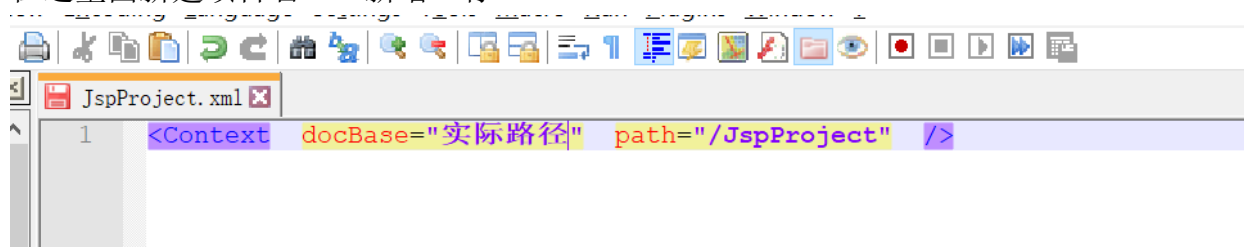
解析：当我访问path里面的路径时，实际访问的是docBase这里面的路径（项目的实际路径）

缺点：配置完以后需要重启

b. 方式二



在这里面新建项目名.xml新增一行



ROOT是默认项目，所以当项目名为ROOT的时候可以不用写项目名直接访问，eg:

localhost:8888/index.jsp

7. 虚拟主机

可以把项目变一个名字，把localhost改为www.test.com

通过www.test.com访问本机

应用原理：



a. 在conf/server.xml中找到Engine去配置

```
-->
<Engine name="Catalina" defaultHost="www.test.com">

  <!--For clustering, please take a look at documentation at:
  /docs/cluster-howto.html (simple how to)
  /docs/config/cluster.html (reference documentation) -->
  <!--
  <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
  -->

  <!-- Use the LockOutRealm to prevent attempts to guess user passwords
  via a brute-force attack -->
  <Realm className="org.apache.catalina.realm.LockOutRealm">
    <!-- This Realm uses the UserDatabase configured in the global JNDI
    resources under the key "UserDatabase". Any edits
    that are performed against this UserDatabase are immediately
    available for use by the Realm. -->
    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
      resourceName="UserDatabase"/>
  </Realm>

  <Host appBase="项目实际地址" name="www.test.com">
    <Context docBase="实际路径" path="/" />
  </Host>

  <Host name="localhost" appBase="webapps"
    unpackWARs="true" autoDeploy="true">
```

a. path="/"当访问这个 "/" 的时候直接访问项目的实际路径

还要记得改引擎的默认地址//在第一排

b. 以上操作完成之后还要修改本机的hosts文件C:\Windows\System\drivers\etc

增加 127.0.0.1 www.test.com

//这个加上可以通过www.test.com来访问本机

然后访问的时候在末尾需要加上端口号：8888才可以正常访问，所以还是可以一眼看出来是假的网站，但把端口号改为80的时候就可以省略不写也可以正常访问了，因为端口号80是网站的默认端口号，还是在server文件中修改端口号

流程：

www.test.com->hosts找映射—》server.xml找Engine的defaultHost—》通过"/"映射到项目的实际路径

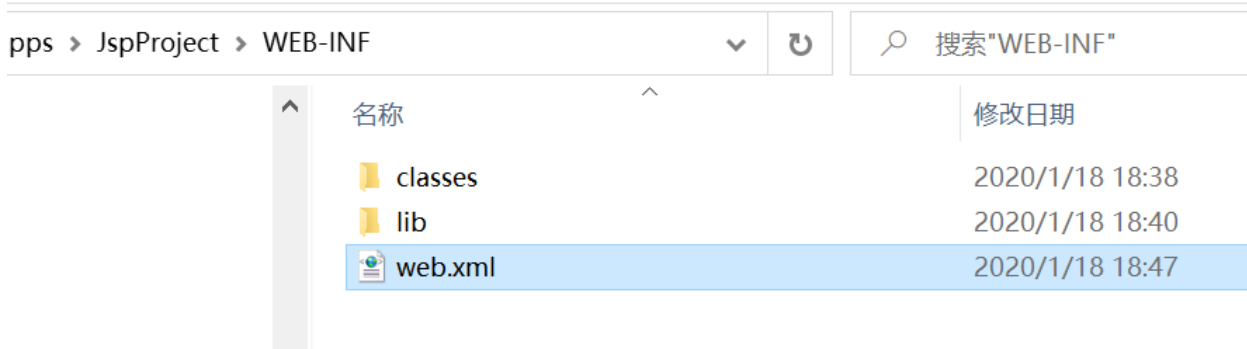
在访问的时候可以不用写/

7. 访问自己的项目



这时访问的是别人的项目，先shutdown.bat关掉，自己在webapps中创建一个项目，里面必须有WEB-INF这个文件夹

查看



名称		修改日期
	classes	2020/1/18 18:38
	lib	2020/1/18 18:40
	web.xml	2020/1/18 18:47

这是建立一个项目有的基本文件夹

作用：

classes：放置一些字节码文件，就是jsp文件最终被编译成class文件的存储的文件夹

lib：放置第三方库文件，只需要放在一个项目中的jre包

web.xml：必须要有以下这些图

```

<!-->
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1"
  metadata-complete="true">

  <display-name>Welcome to Tomcat这也可以改</display-name>
  <description>
    描述信息的部分: hello jspproject
  </description>

</web-app>

```

建完项目就自己做文件

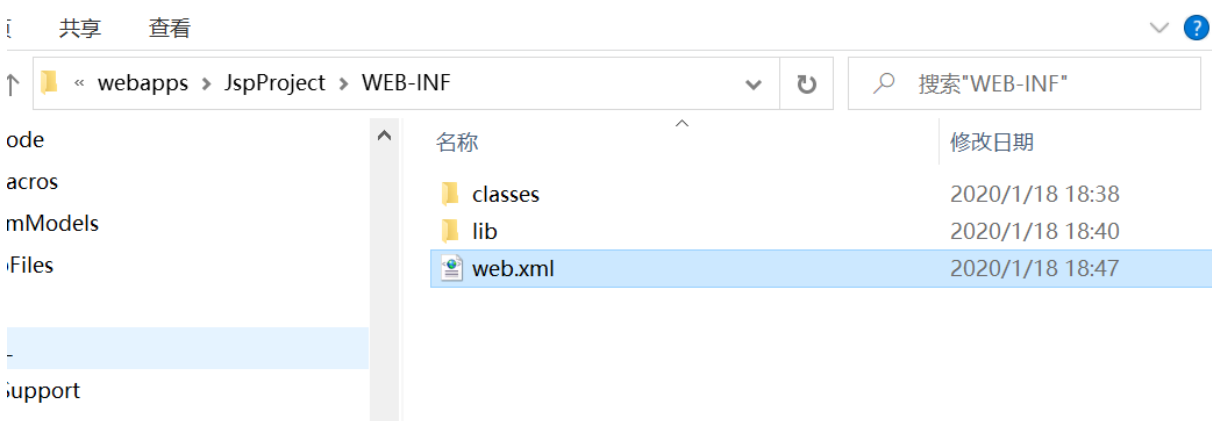
```

index.jsp
1 <html>
2   <head>
3     <title>my jsp project</title>
4   </head>
5
6   <body>
7     hello jsp...//这里写内容
8     <%
9       out.print("hello world...");
10      //这里写java脚本
11     %>
12   </body>
13 </html>

```

总结: jsp就是在html中嵌套的java代码

当有多个文件的时候, 指定访问的文件, 在项目的WEB-INF文件夹的web.xml文件里的welcome指定的初始页面



执行效果

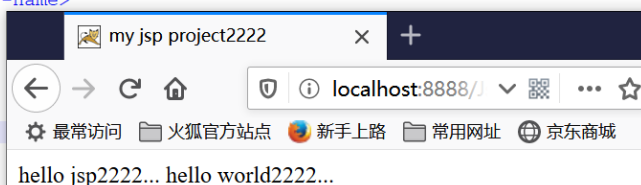
```

metadata-complete="true">

<display-name>Welcome to Tomcat这也可以改</display-name>
<description>
  描述信息的部分: hello jspproject
</description>
<welcome-file-list>
  <welcome-file>index2.jsp</welcome-file>
</welcome-file-list>

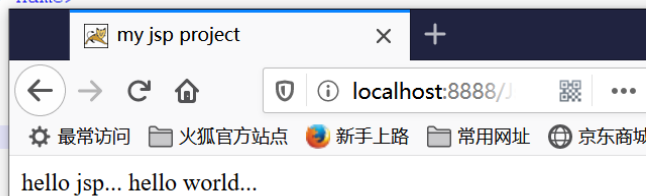
</web-app>

```




```
dex.jsp x index2.jsp x web.xml x
<html>
  <head>
    <title>my jsp project2222</title>
  </head>
  <body>
    hello jsp2222...
    <%
      out.print("hello world2222...");
    %>
  </body>
</html>
```

```
display-name>Welcome to Tomcat这也可以改</display-name>
description>
  描述信息的部分: hello jspproject
</description>
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```



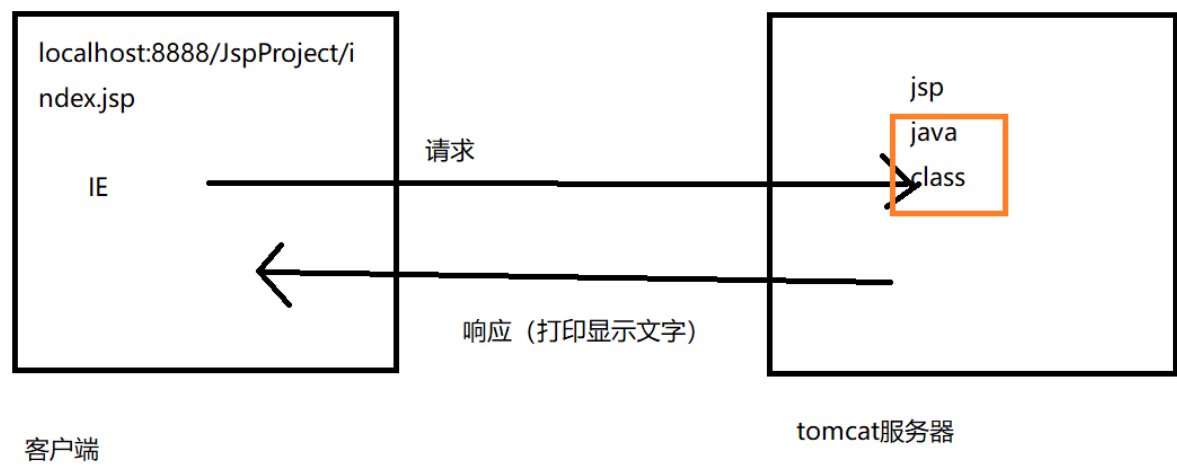
```
index.jsp x index2.jsp x web.xml x
1 <html>
2   <head>
3     <title>my jsp project</title>
4   </head>
5
6   <body>
7     hello jsp...
8     <%
9       out.print("hello world...");
10    %>
11   </body>
12
13 </html>
```

也可以直接写死，如下图



8. JSP执行流程

a. 大致的图解



第一次访问：服务端将JSP翻译成java，再将java编译成class文件

第二次访问：直接访问class文件（如果服务端修改了代码，将会再访问时重新翻译、编译）

b. 将jsp->java（Servlet文件）->class最终储存在下图

各种软件的安装包 > apache-tomcat-8.5.50 > work > Catalina > localhost > JspProject > org > apache > jsp					▼	🔄
名称	修改日期	类型	大小			
index_jsp.class	2020/1/18 19:15	CLASS 文件	6 KB			
index_jsp.java	2020/1/18 19:15	JAVA 文件	5 KB			
index2_jsp.class	2020/1/18 19:17	CLASS 文件	6 KB			
index2_jsp.java	2020/1/18 19:17	JAVA 文件	6 KB			

Jsp和Servlet文件可以相互转换

get和post都是http请求方式。

它们的区别如下：

区别一，get请求重点在从服务器上获取资源，而post请求重点在向服务器发送数据。

区别二，get传输数据是通过URL请求，以字段=value的形式，用？连接置于URL之后，多个请求数据之间用&连接，这个过程用户可见，因此是不安全的。

post传输数据是通过http的post机制，将字段与对应值封存在请求实体中发送给服务器，这个过程用户不可见，因此是安全的。

区别三，get传输的数据量小，受URL长度的限制，但是效率高。

post传输数据量不受限制，可以传输大量数据，所以传输文件时只能使用post，但是效率较低。请求较多时可能形成一个请求队列。

区别四，get方式只支持ASCII字符，因此向服务器传输中文有可能出现乱码。

post支持标准字符集，可以正确传递中文字符。