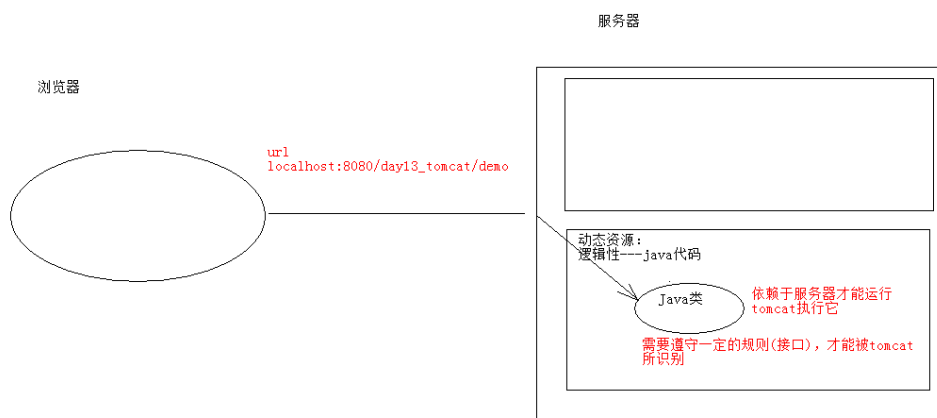


认识Servlet

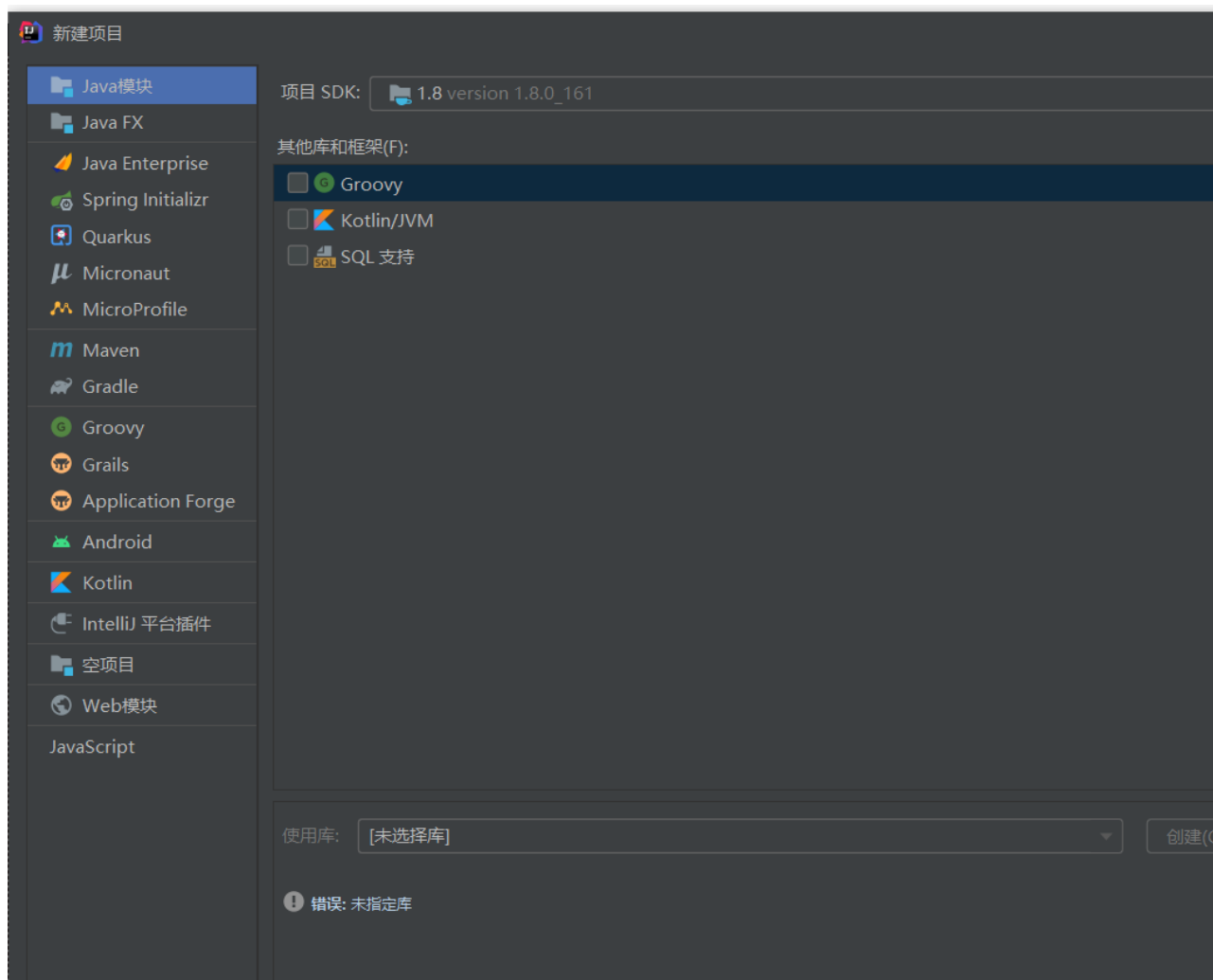
- 服务器里的静态资源与动态资源可以被访问到
- 每一个用户访问动态资源时看到的都是不一样的，即动态资源有逻辑性，用java代码来体现，java代码是封装在类当中的
- 浏览器请求动态资源时找的是java类，这种java类与普通的java类是不一样的，这种Java类不是自己去运行的，要依赖于服务器才能运行，主方法没有main方法，这个类需要tomcat去执行它
- 并不是所有的java类tomcat都可以去执行的，tomcat可以执行的类需要遵守一定的规则，tomcat才能识别到这种类、执行它、并且创建它的对象、调用它的方法等
- 在java里只要遇见规则这两个字，那么他就是接口；这个接口这个规则就是servlet；这个Java类就是sevlet的是实现
- servlet就是一个接口，定义了Java类被浏览器访问到(tomcat识别)的规则
- servlet是运行在服务器上的一个的小应用程序
- 将来我们自定义一个类，实现Servlet接口，复写方法。
- 在javaEE，里面搜索servlet
注：在jdk的api里搜索不到，必须得在jdk ee里才能搜到
- servet里有5个方法要去实现
- 比如网站访问项目的某个资源，即访问demo资源，搜索localhost:8080/day13_tomcat/demo,需要通过这个资源名称去映射这个类的访问，demo就是这个资源的名称，demo要去找找到这个Java类，所以我们还需要把这个Java类去映射成路径



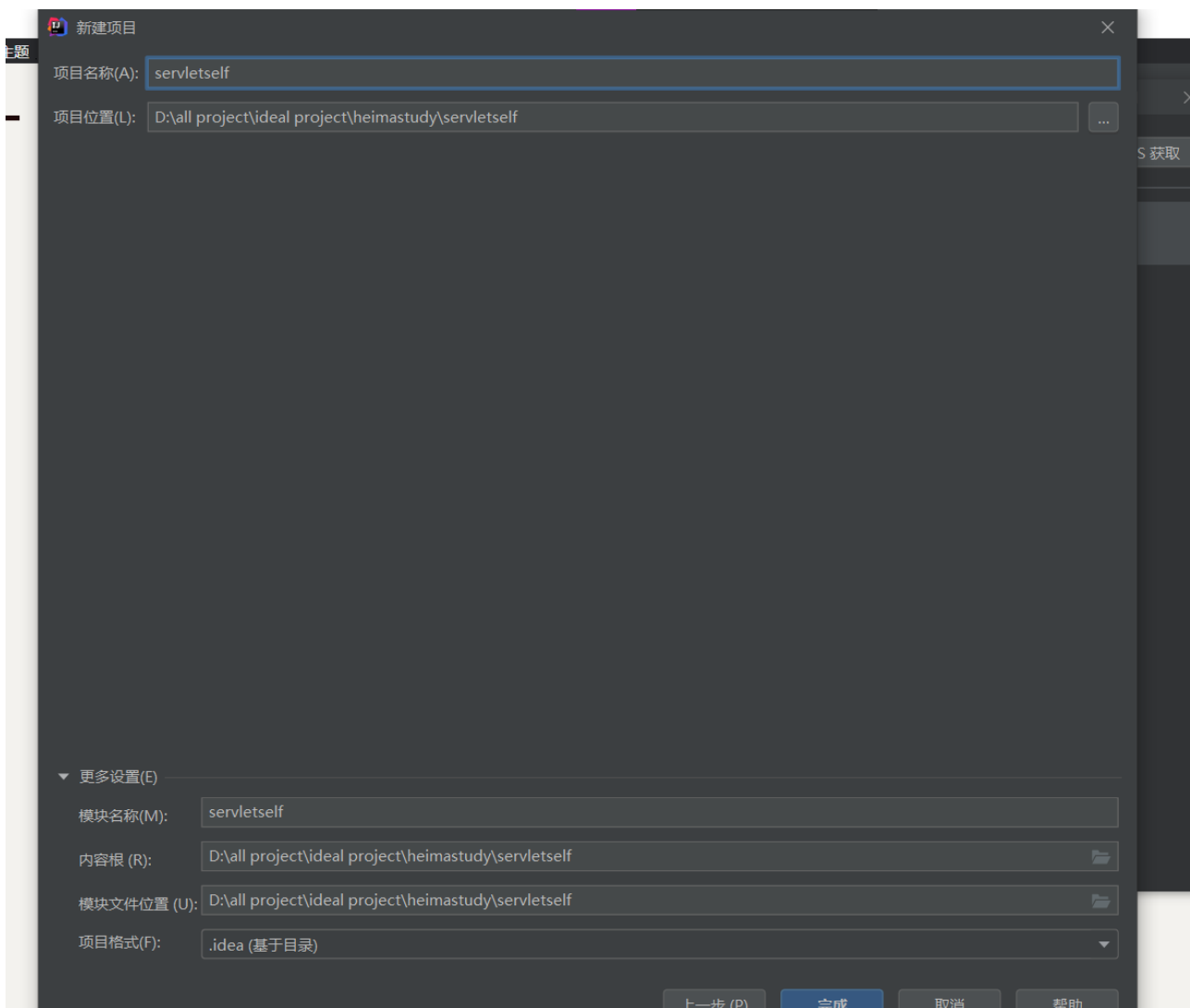
Method Summary	
Methods	
Modifier and Type	Method and Description
void	destroy () Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
ServletConfig	getServletConfig () Returns a ServletConfig object, which contains initialization and startup parameters for this servlet.
String	getServletInfo () Returns information about the servlet, such as author, version, and copyright.
void	init (ServletConfig config) Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
void	service (ServletRequest req, ServletResponse res) Called by the servlet container to allow the servlet to respond to a request.

Servlet在IDEA中的配置即入门

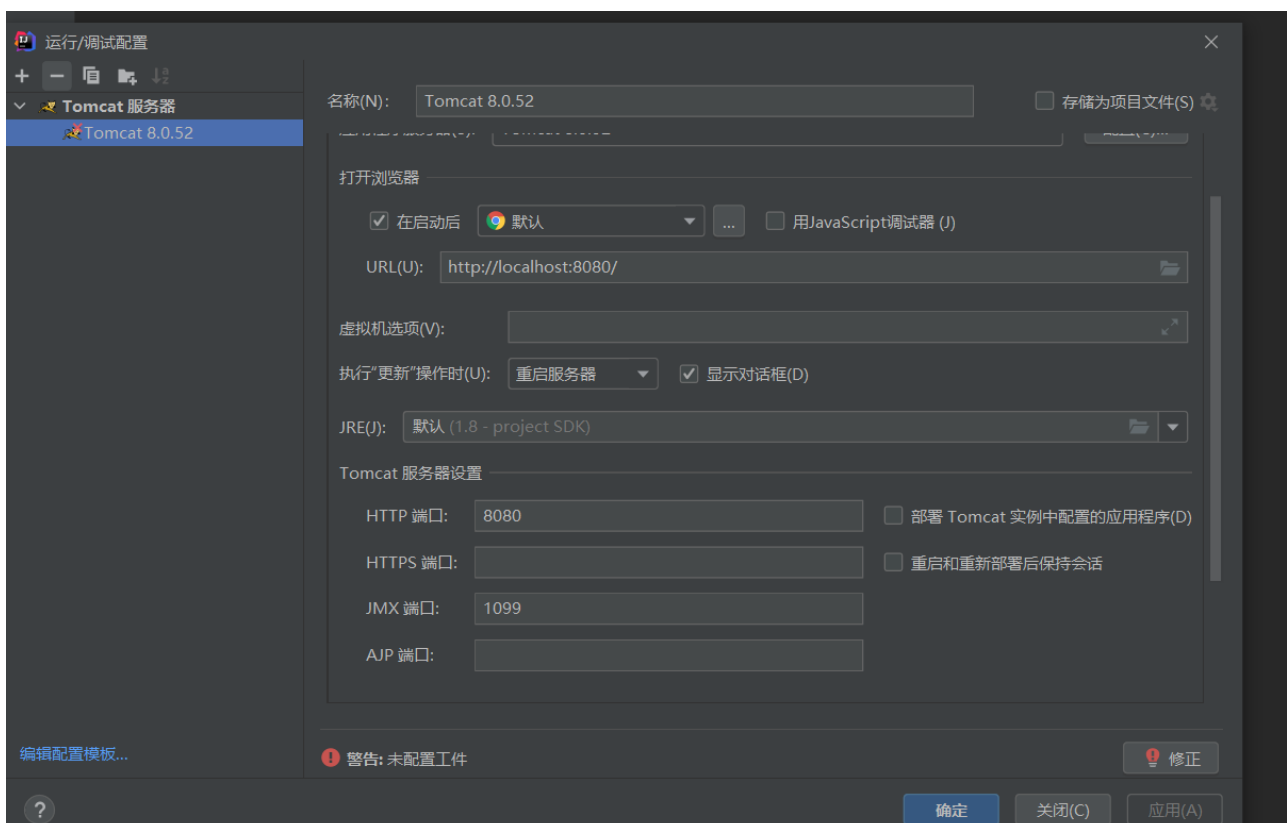
- 创建“Java模块”

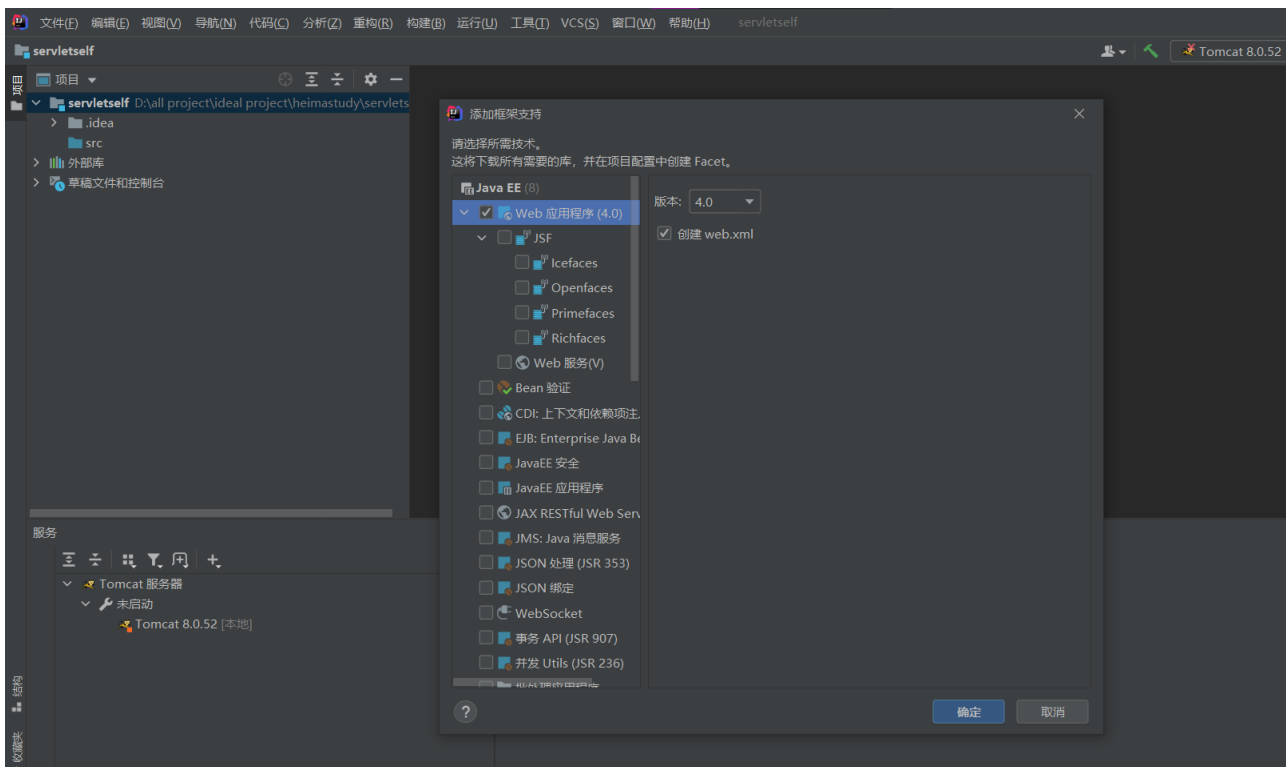


- 命名

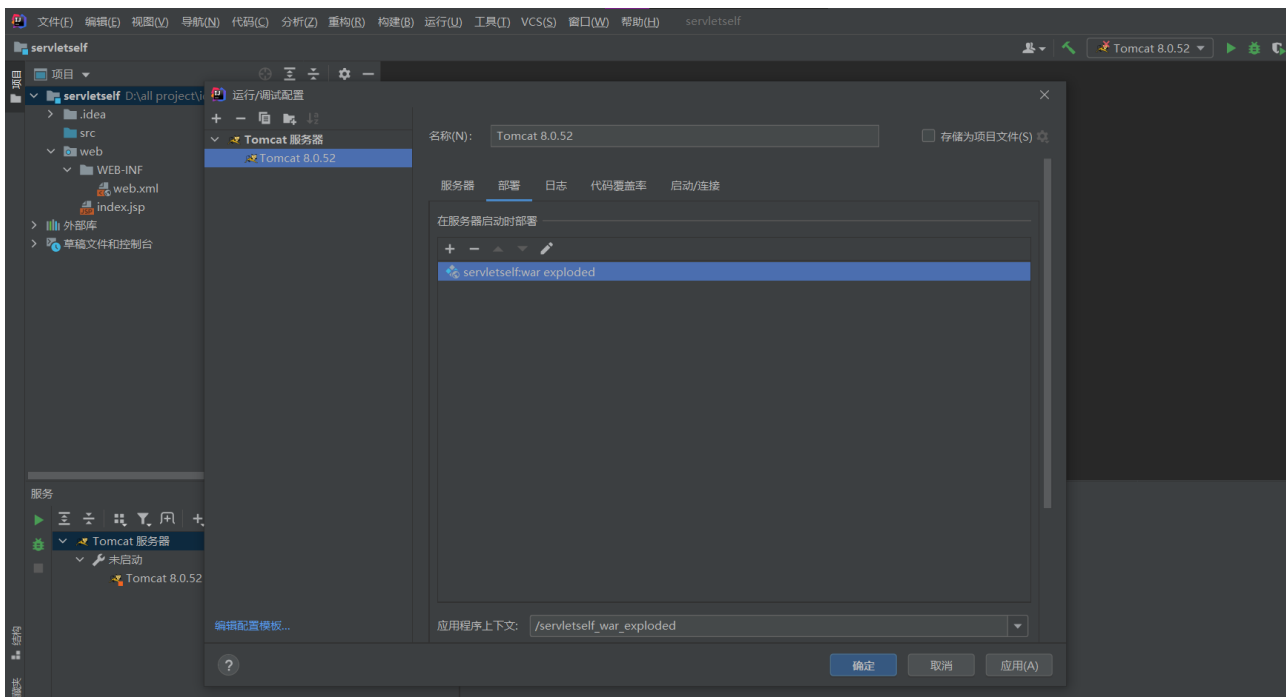


- 配置tomcat、增添加web框架





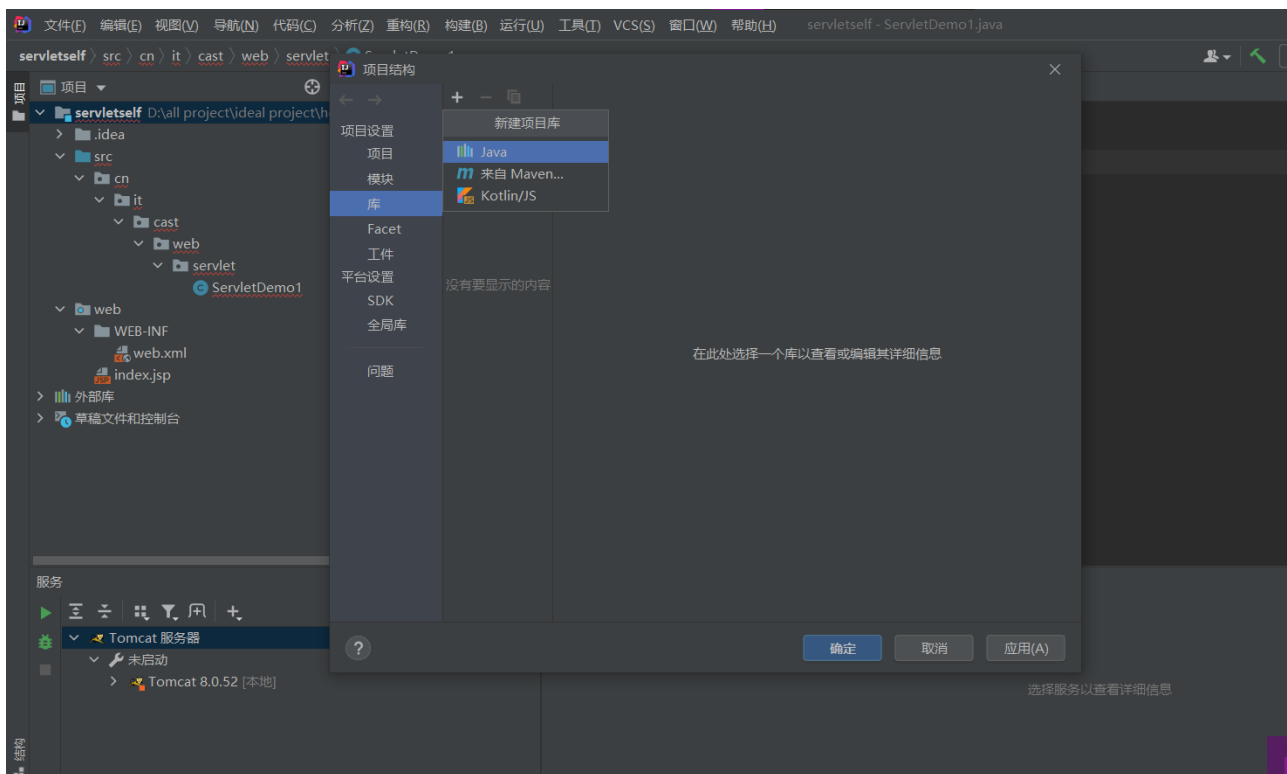
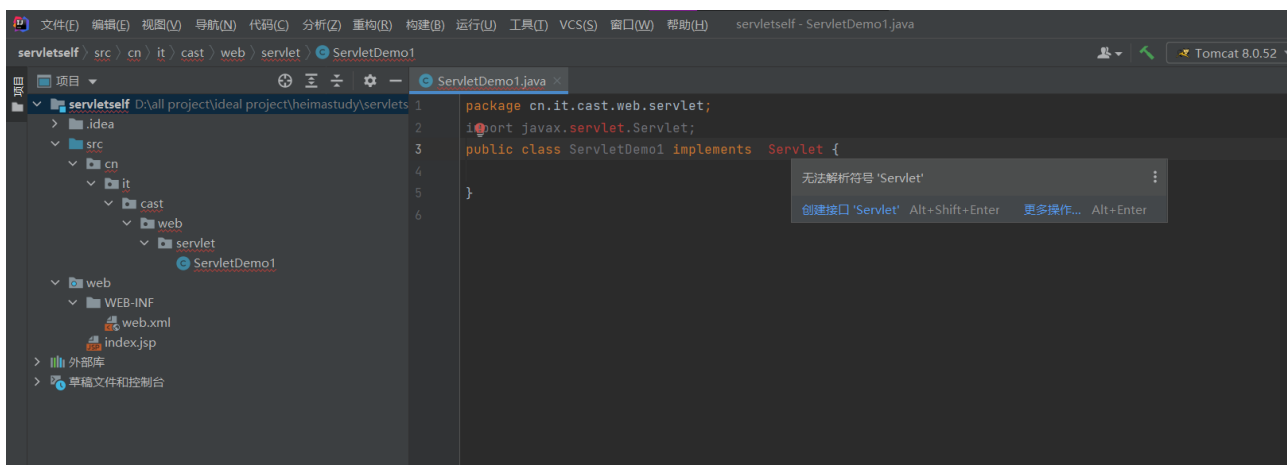
- 集成模块到tomcat

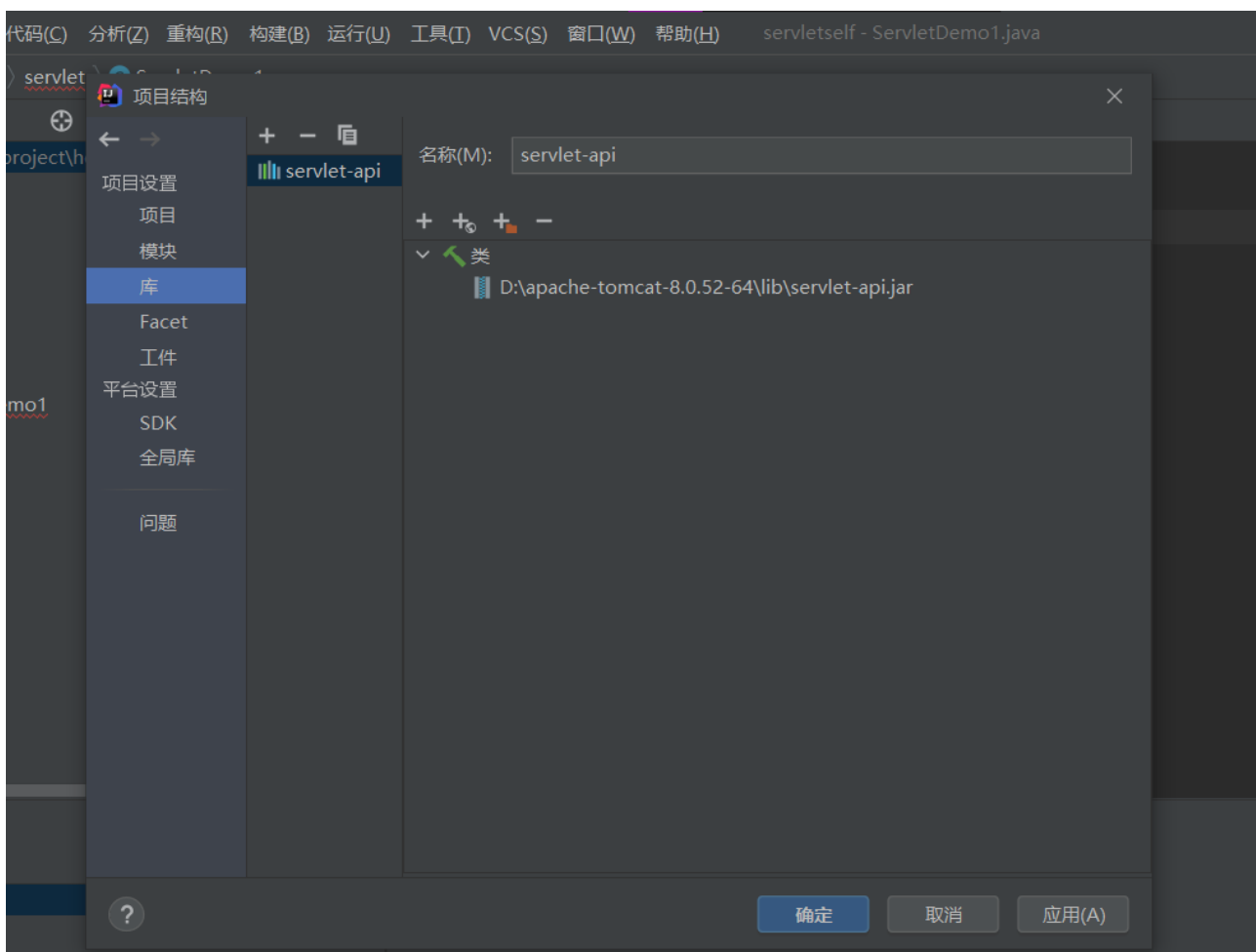
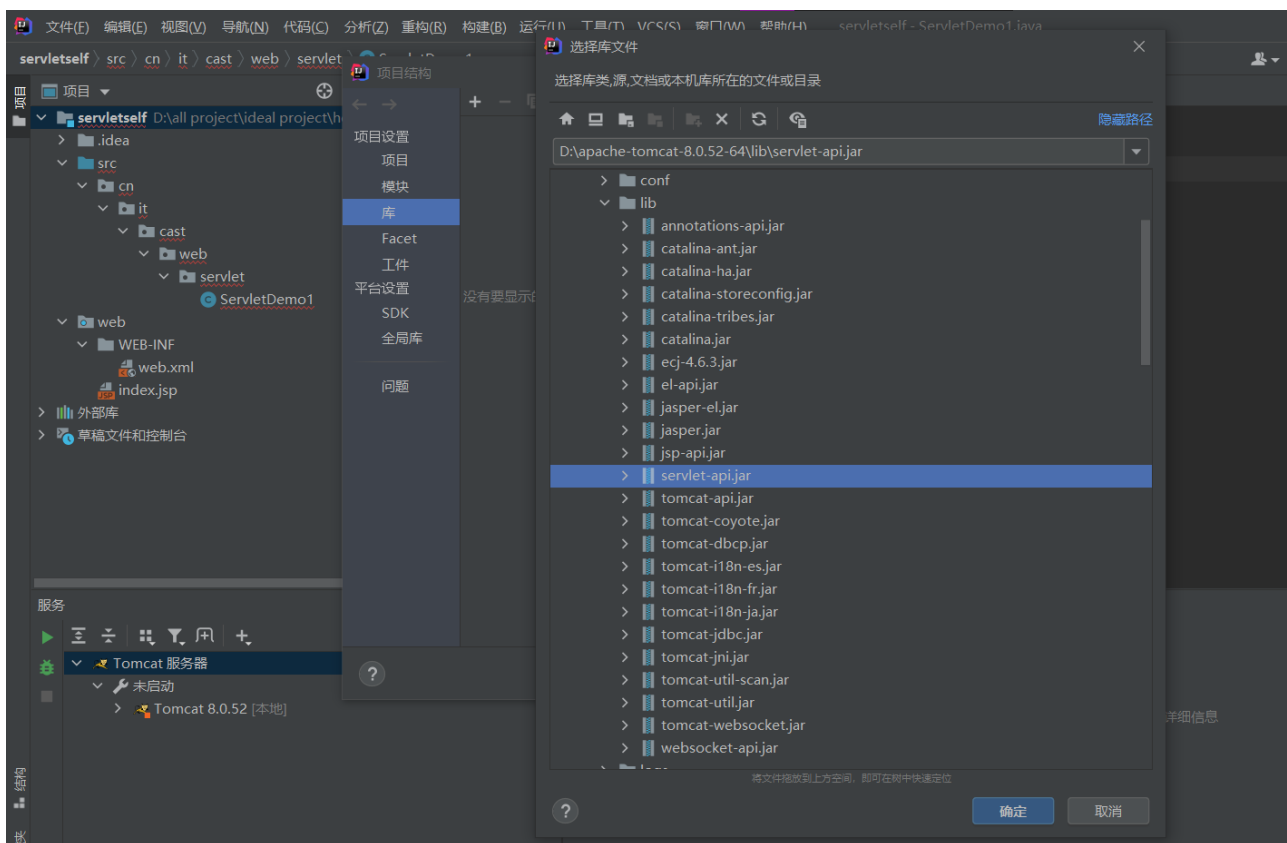


- src下创建cn.it.cast.web.servlet.ServletDemo1,把这个Java类实现servlet接口，但系统提示发现没有可找到的包，需要按以下方法去导入包：

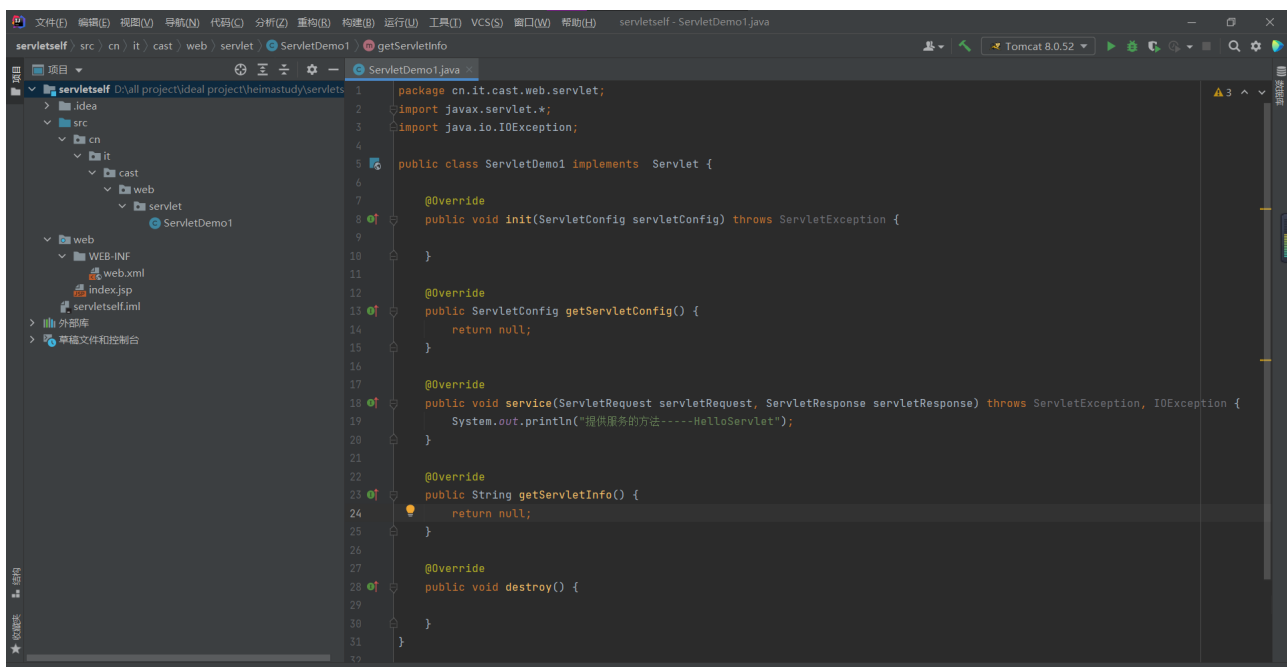
项目结构-----库-----+号-----Java-----选择servlet-api.jar----应用

然后即可成功导入包

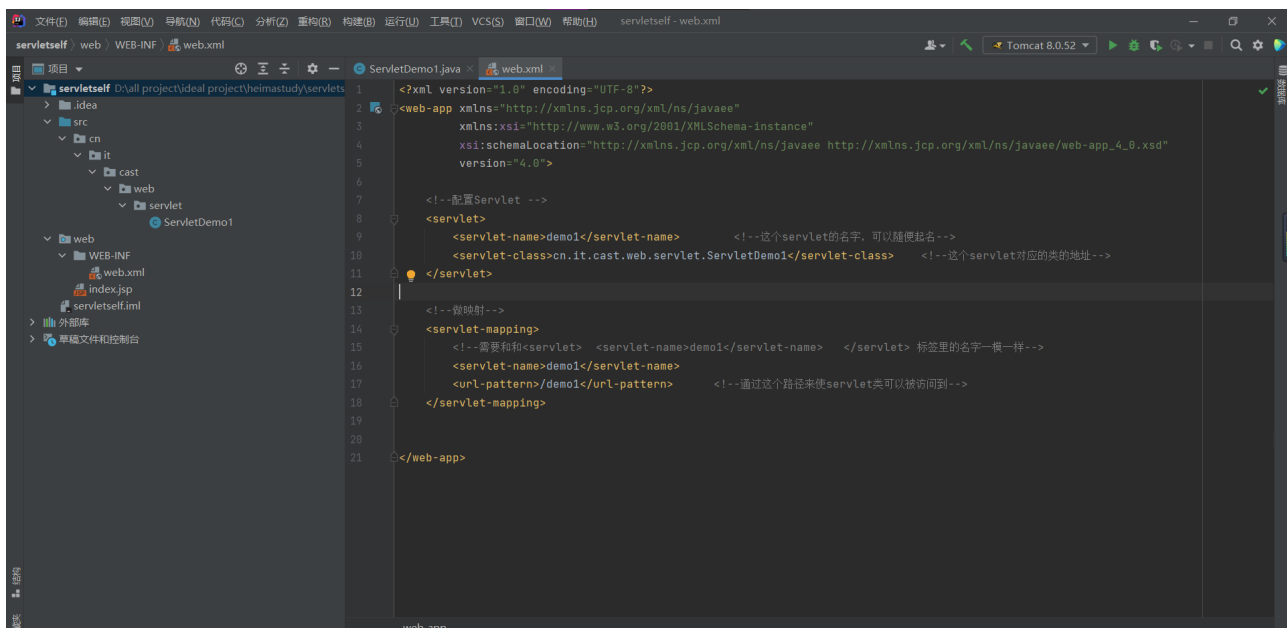




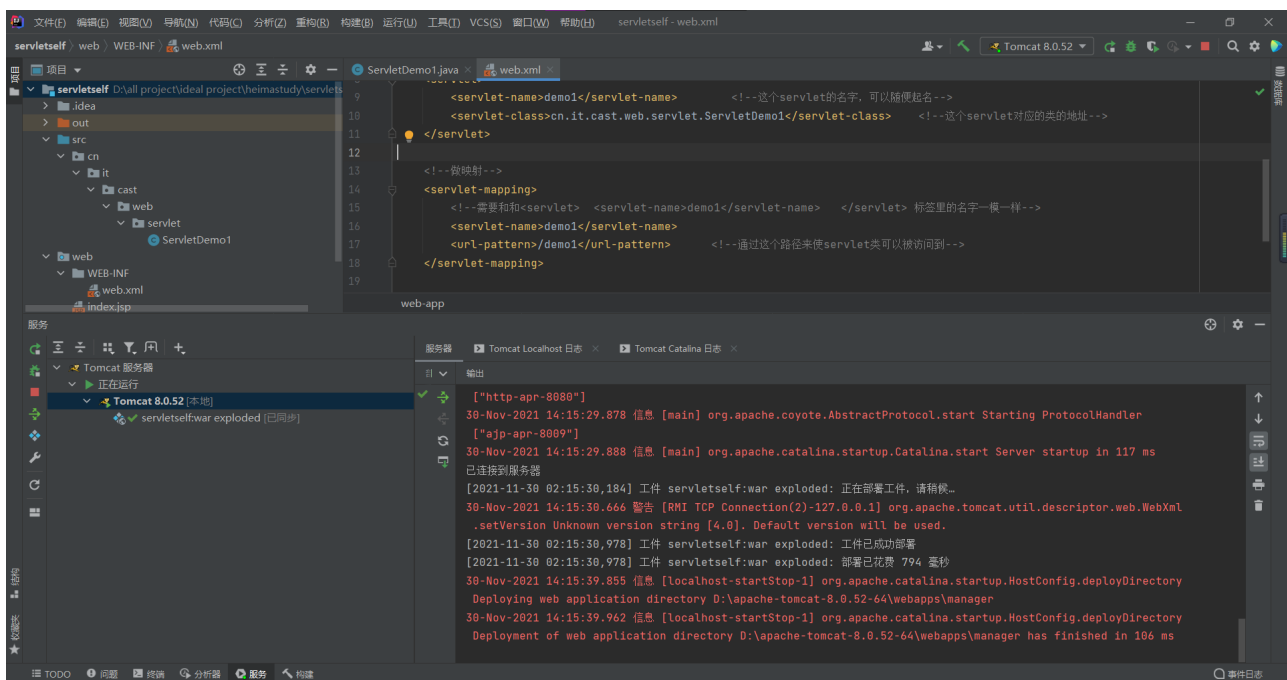
- 在ServletDemo1里继承Servlet接口后，快捷键实现5个方法，再在service方法里输出一句话



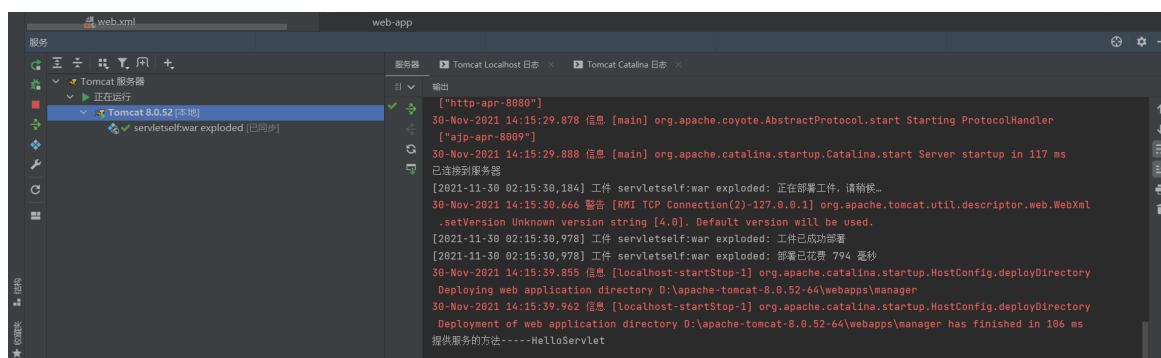
- 配置Servlet,在web.xml中配置增加以下代码



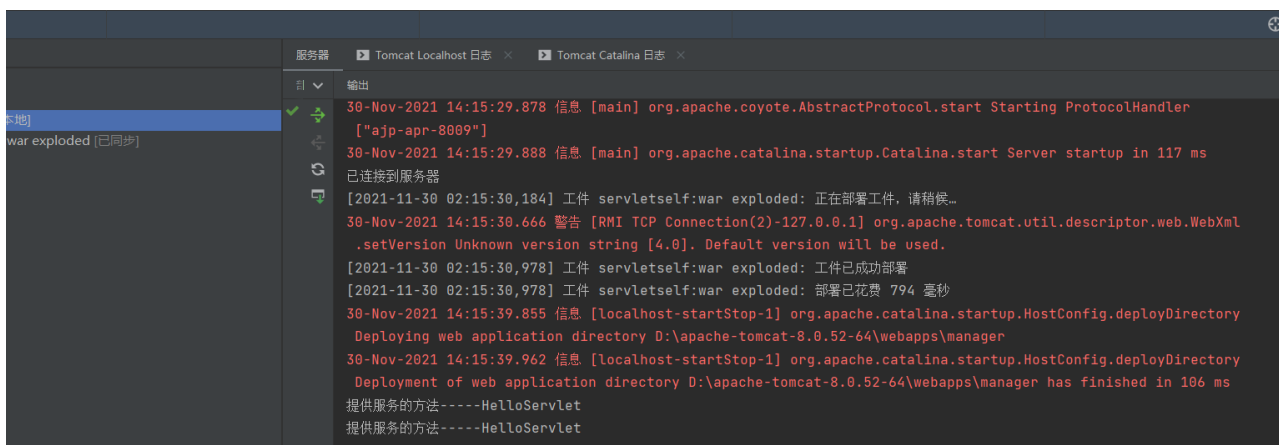
- 启动tomcat，主界面输出以下信息，且项目目录结构会出现out目录



- 网页访问（http://localhost:8080/servletself_war_explode/demo1）,网页永远是白色的；
控制台会输出"提供服务的方法-----HelloServlet";



- 如果再一次访问（http://localhost:8080/servletself_war_explode/demo1），控制台会再一次输出"提供服务的方法-----HelloServlet";



Servlet入门步骤总结

1. 新版idea创建“Java模块”，并配置tomcat，添加web框架（如上）
2. 定义一个类，并导入servlet包，实现Servlet接口

```
public class ServletDemo1 implements Servlet
```

3. 实现接口中的抽象方法(点击下面红色下划线会出现实现方法，请点击，实现5个方法)，在service里写入

```
System.out.println("提供服务的方法-----HelloServlet");
```

4. 配置Servlet,在web.xml中配置增加以下代码

```
<!--配置Servlet -->
<servlet>
    <servlet-name>demo1</servlet-name>          <!--这个
servlet的名字，可以随便起名-->
    <servlet-
class>cn.it.cast.web.servlet.ServletDemo1</servlet-class>
<!--这个servlet对应的类的地址-->
</servlet>

<!--做映射-->
<servlet-mapping>
    <!--需要和<servlet> <servlet-name>demo1</servlet-
name>    </servlet> 标签里的名字一模一样-->
    <servlet-name>demo1</servlet-name>
    <url-pattern>/demo1</url-pattern>          <!--通过这个路
径来使servlet类可以被访问到-->
</servlet-mapping>
```

5. 网页访问 (http://localhost:8080/servletself_war_explode/demo1), 网页永远是白色的;

控制台会输出"提供服务的方法-----HelloServlet";

Servlet执行原理步骤

1. 当服务器接受到客户端浏览器的请求后, 会解析请求URL路径 (即用户搜索的http://localhost:8080/servletself_war_explode/demo1)

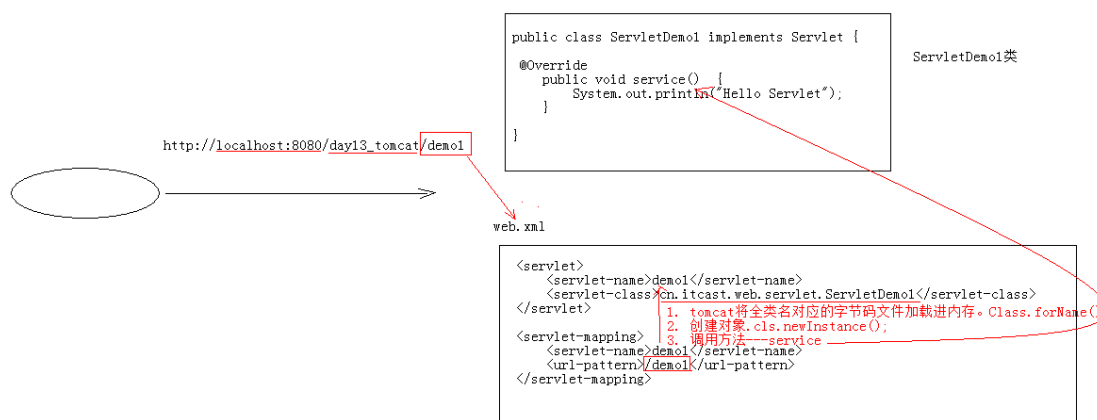
2. 查找web.xml文件, 逐一查找是否有对应的标签体内容是demo01。

注: 如果是访问web下创建的普通html文件的话, 是不需要去web.xml里访问的, 因为web.xml里面是没有它的url-pattern标签的

3. 如果有, 则在找到对应的全类名

4. tomcat会将字节码文件加载进内存, 并且创建其对象

5. 调用其方法



Servlet中的生命周期方法及详解:

1. init方法: 初始化方法, 用于加载资源, 特点是在servlet被创建时执行init方法, 都只会执行一次

• Servlet的创建

* 第一种, 默认情况下, 第一次被访问http://localhost:8080/servletself_war_explode/demo2时Servlet被创建, 此时init方法被调用;

即在web.xml里它所属的标签下是默认-1

*第二种，只要tomcat服务器一启动，Servlet就会被创建且init方法就会被调用

在web.xml里demo03的标签下写上5

```
<servlet>
  <servlet-name>demo2</servlet-name>
  <servlet-class>cn.it.cast.web.servlet.ServletDemo2</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>demo2</servlet-name>
  <url-pattern>/demo2</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>demo3</servlet-name>
  <servlet-class>cn.it.cast.web.servlet.ServletDemo3</servlet-class>
  <!--指定servlet的创建时期;load-on-startup翻译是当服务器启动时加载;
  共有两种, 1.第一次被访问时创建:配 负数 (如果不写这个标签的话时是默认为-1的)
  2.在服务器启动时创建:配 0或正整数 此时只要服务器一点击绿色三角形init方法就会被调用-->
  <load-on-startup>5</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>demo3</servlet-name>
  <url-pattern>/demo3</url-pattern>
</servlet-mapping>
```

- Servlet的init方法，只执行一次，说明一个Servlet在内存中只存在一个对象，Servlet是单例的
- 多个用户同时访问一个对象时，可能在线程安全问题（Servlet是共享的）
- 解决：尽量不要在Servlet中定义成员变量。即使定义了成员变量，也不要对修改值

2.service方法：提供服务的方法，特点是每一次servlet被访问网页时都会执行，会执行多次

3.destroy方法：销毁方法，在servlet被杀死时执行一次（精确来说destroy方法执行是时servlet被销毁之前），一般用于释放资源

- 即点击右上角红色方形服务器被正常关闭时执行一次(当服务器被关闭时servlet就被销毁了)
- 只有服务器正常关闭时，才会执行destroy方法。（给窗口点击叉那种属于非正常关闭）

4.ServletConfig: 获取servletConfig对象，即获取servlet的配置对象（这个方法一般情况下是不用它的）

5.getServletInfo: 用来获取servlet的一些信息：版本作者等等（这个方法一般情况下是不用它的）

Servlet3.0版本以上注解配置

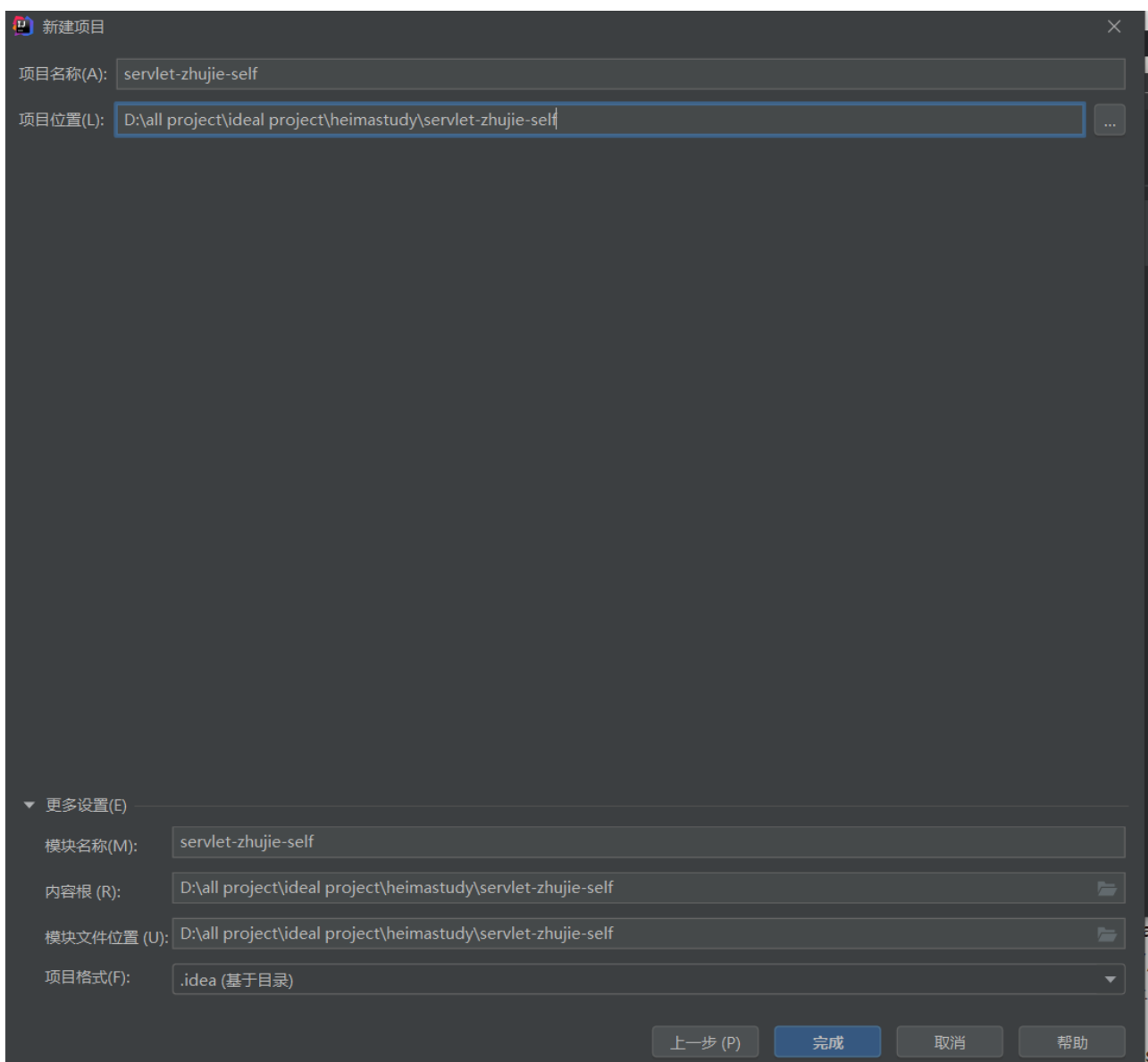
1.注解的认识:

- Servlet3.0版本之前只能进行web.xml配置;
Servlet3.0版本之后不仅能进行web.xml配置也能进行注解配置
- 好处: 支持注解配置, 可以不需要web.xml了 (可以用xml配置文件配置, 也可以不用配置文件配置)

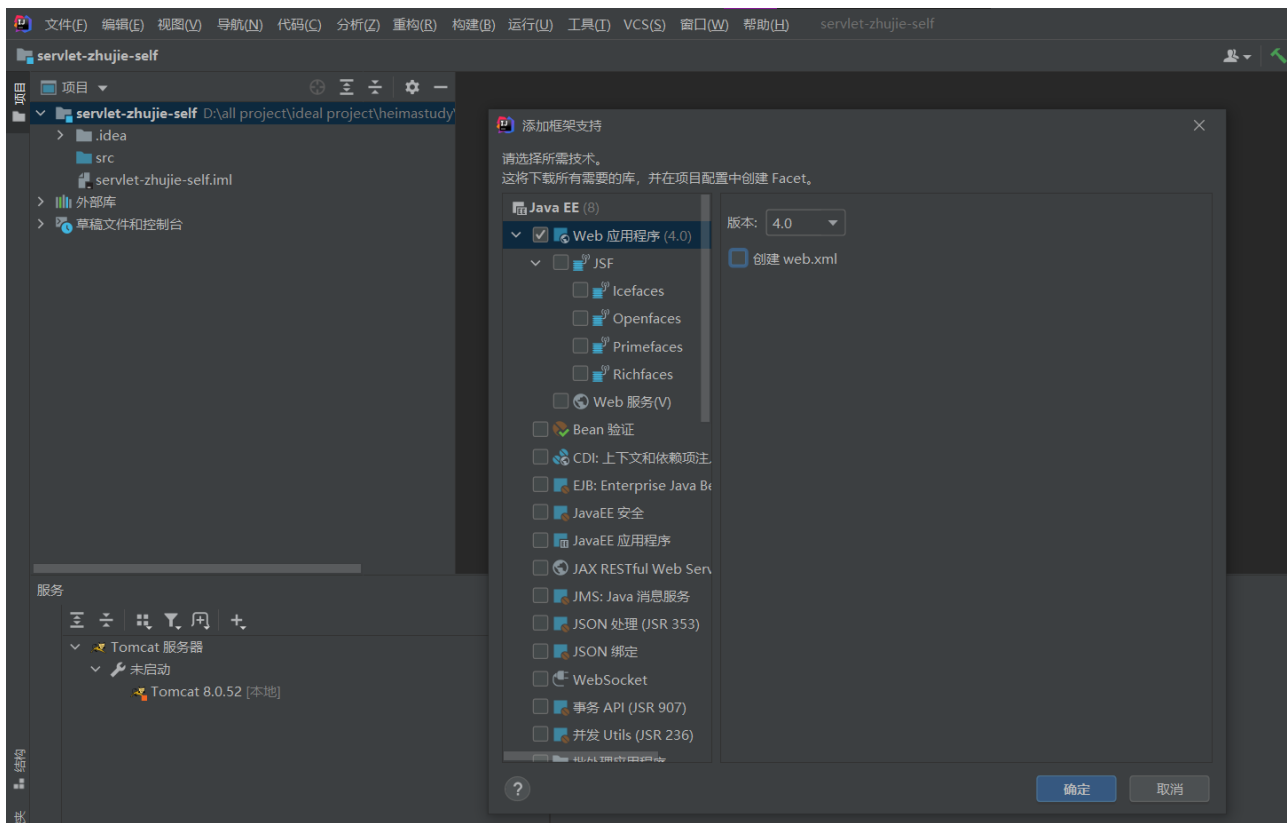
1.IDEA创建

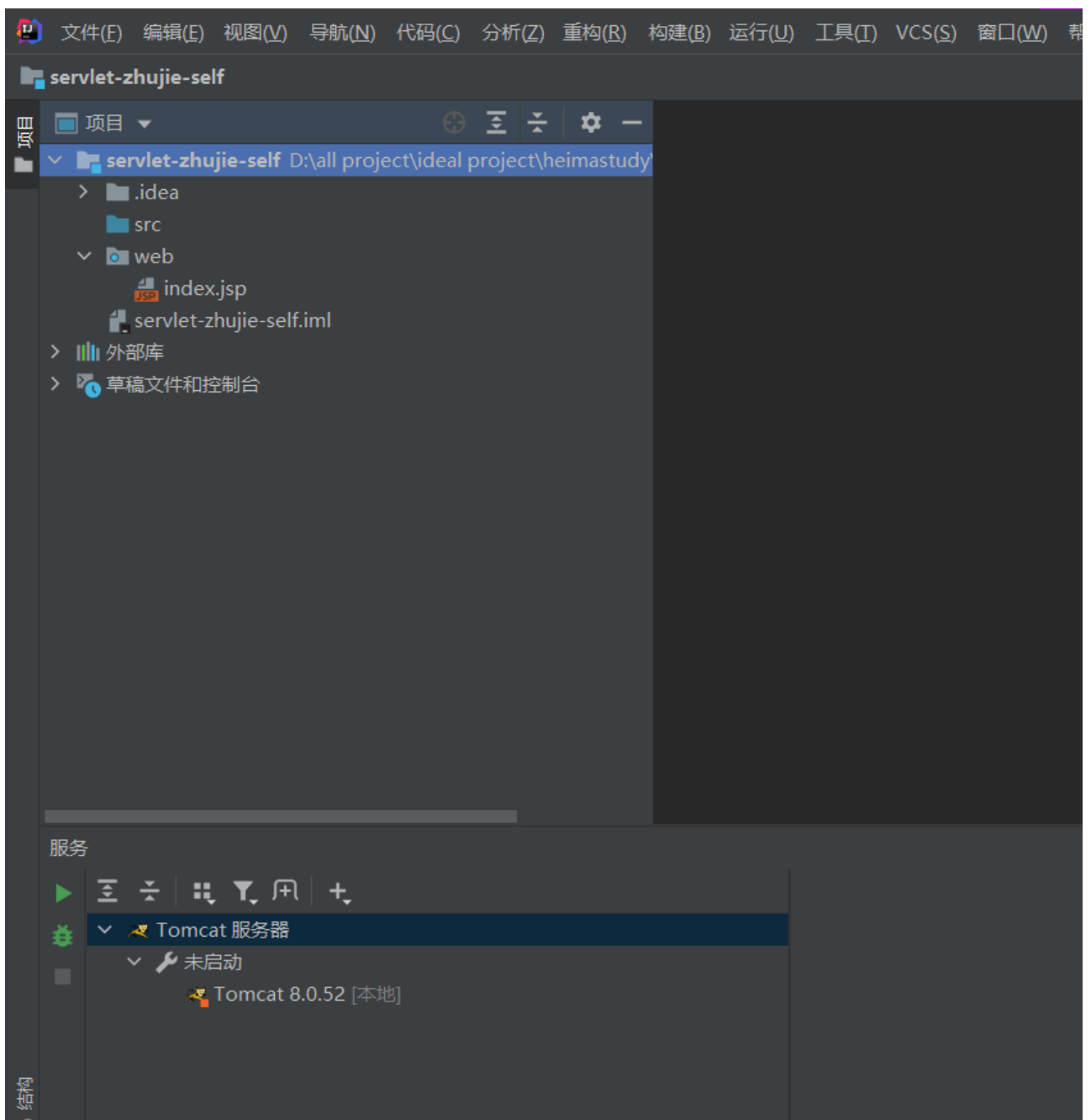
2.IDEA中注解配置

- 创建"Java模块", 并进行tomcat配置; 并项目结构lib添加servlet的api

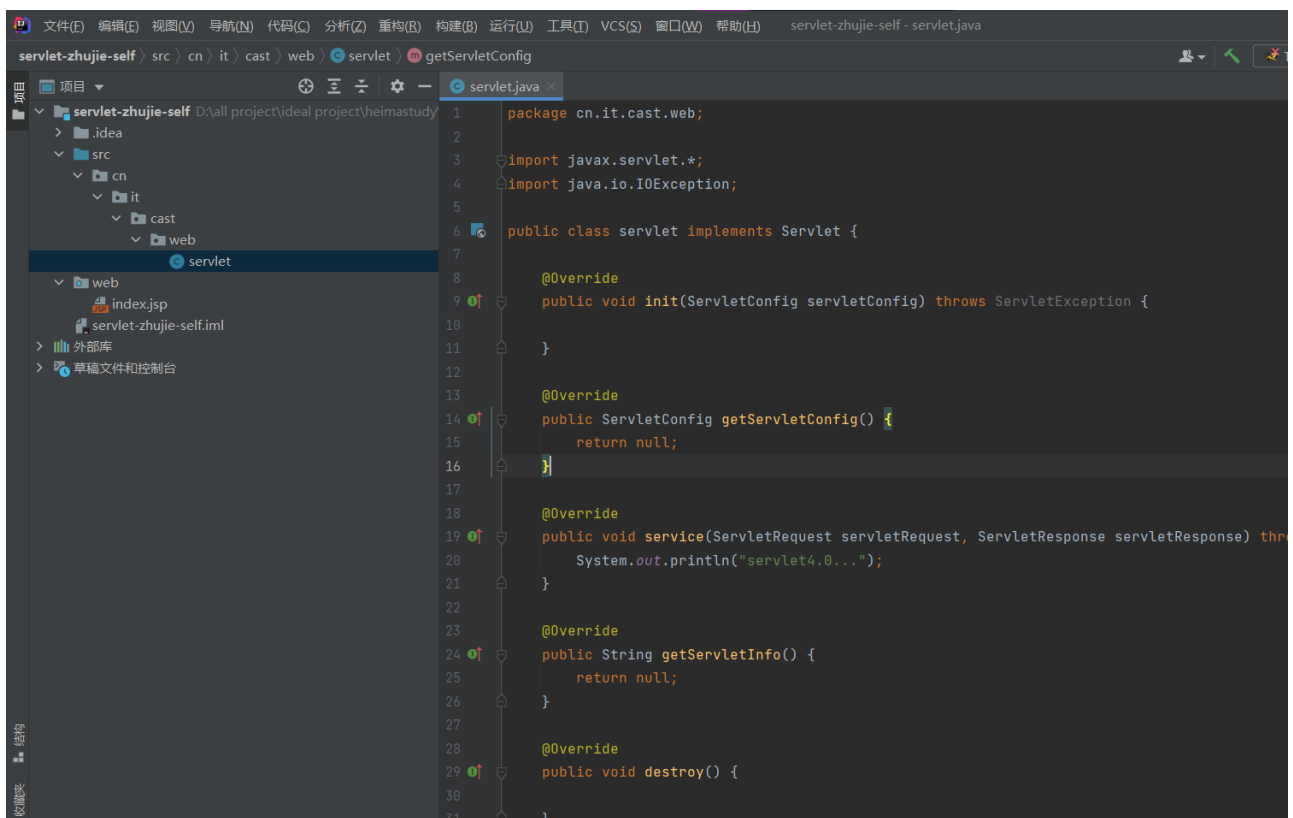


- 框架选择Web-----不要勾选"创建web.xml", 如下为创建后的文件目录结构 (WEB-INF文件夹也没有了)



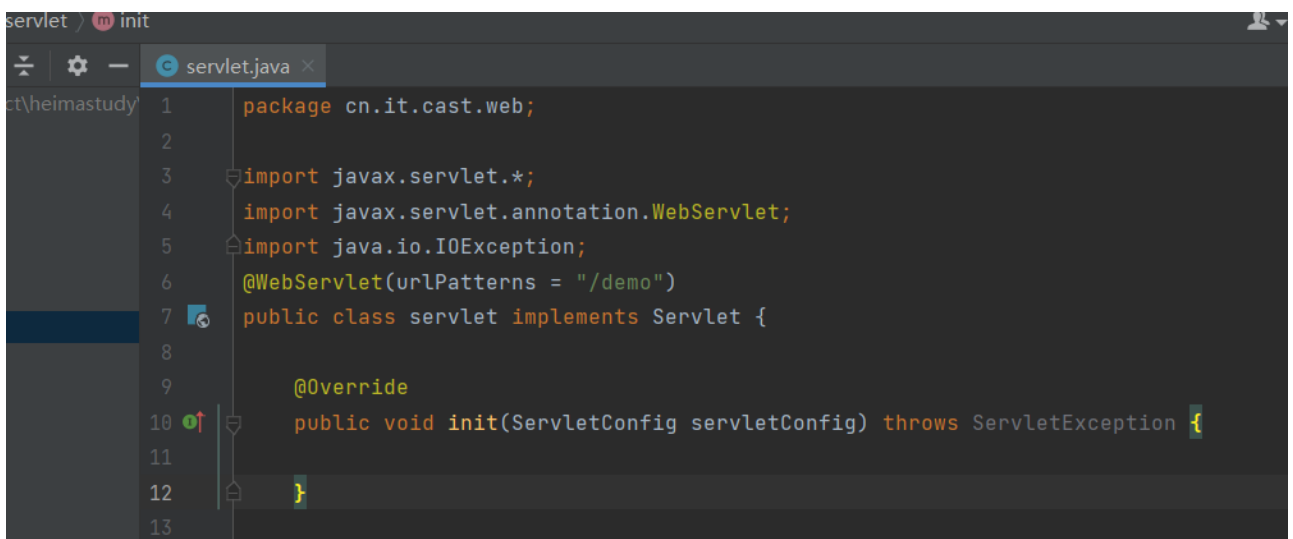


- 定义一个类，实现Servlet接口； 复写方法service方法内写个输出

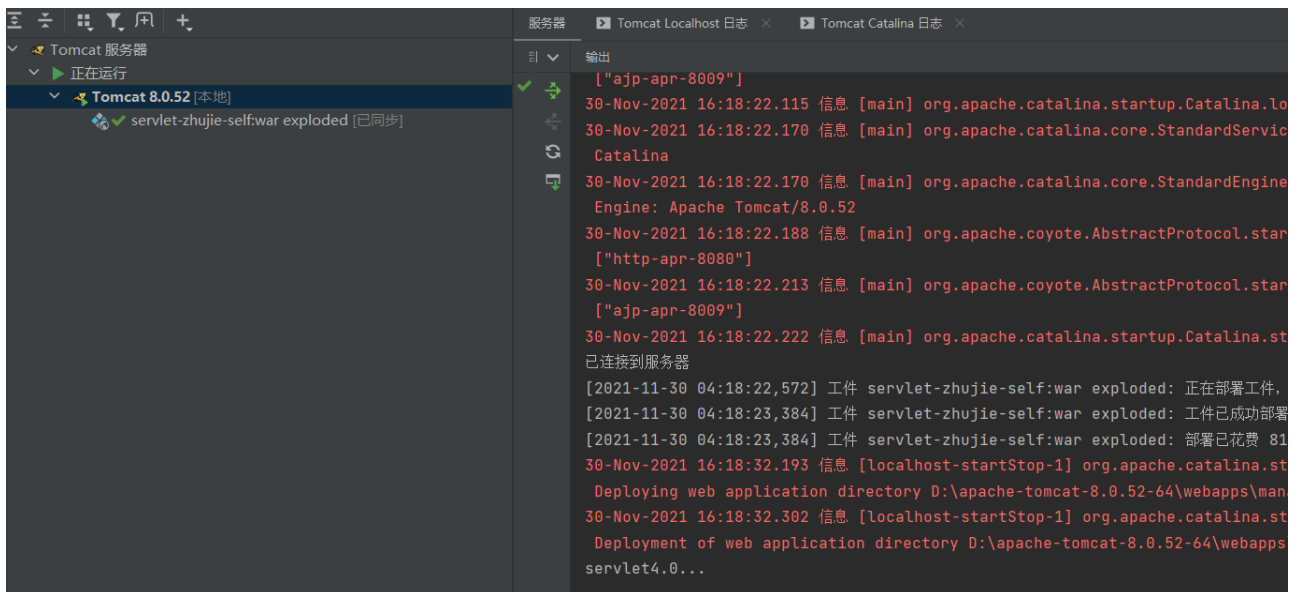
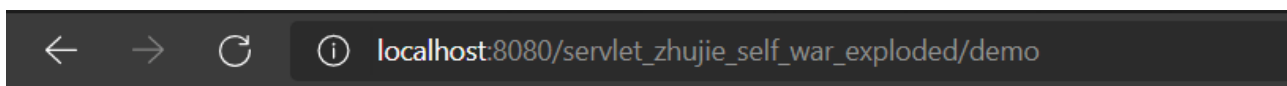


- 在类上使用注解以下代码，进行配置

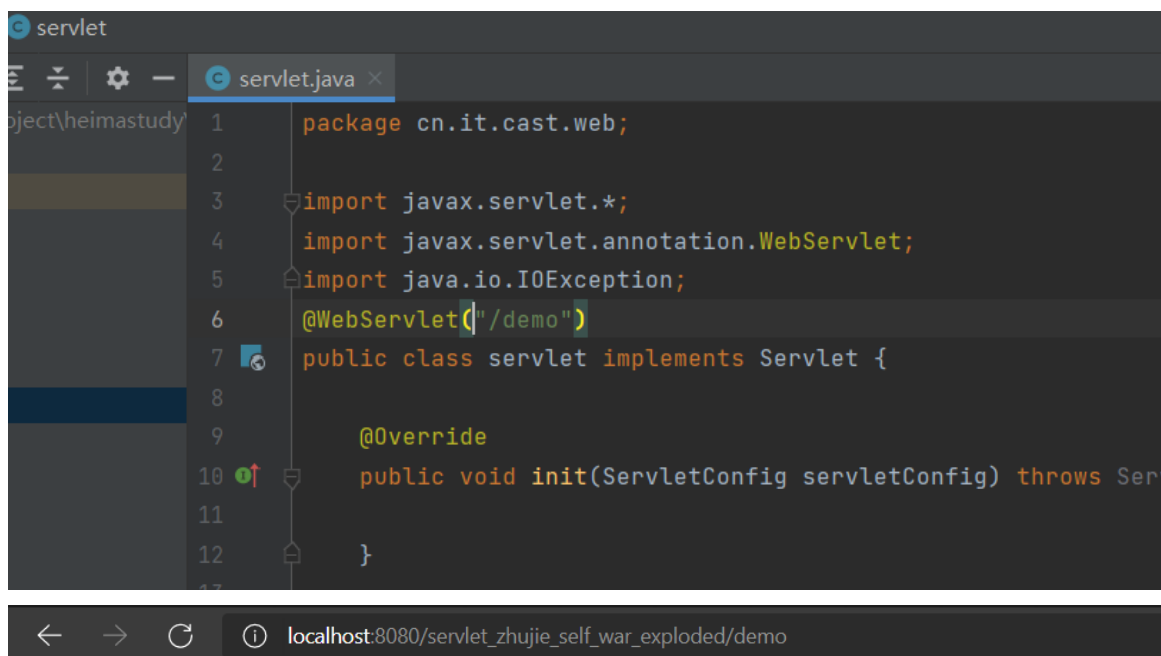
```
@WebServlet(urlPatterns = "/demo")
```



- 启动tomcat服务器，并访问
localhost:8080/servlet_zhujie_self_war_exploded/demo可以成功，且控制台会进行service方法输出



- 若注解@WebServlet () 这括号里只有一个属性的话, 可以用 @WebServlet("/demo") 代替 @WebServlet(urlPatterns = "/demo")




```
[2021-11-30 04:32:03,981] 工件 servlet-zhujie-self:wa
30-Nov-2021 16:32:03.520 信息 [main] org.apache.coyot
["ajp-apr-8009"]
30-Nov-2021 16:32:03.531 信息 [main] org.apache.catal
已连接到服务器
[2021-11-30 04:32:03,981] 工件 servlet-zhujie-self:wa
[2021-11-30 04:32:04,803] 工件 servlet-zhujie-self:wa
[2021-11-30 04:32:04,803] 工件 servlet-zhujie-self:wa
30-Nov-2021 16:32:13.506 信息 [localhost-startStop-1]
Deploying web application directory D:\apache-tomcat-9.0.40\
30-Nov-2021 16:32:13.611 信息 [localhost-startStop-1]
Deployment of web application directory D:\apache-tomcat-9.0.40\
servlet4.0...
```

Servlet继承体系结构

1.Servlet的体系结构:servlet下面总共有两个实现类，分别是GenericServlet(它的儿子) 和 HttpServlet(它的孙子)

Servlet -- 接口（爷爷）

|

GenericServlet -- 抽象类（爸爸）（真正开发中并不用这种GenericServlet方式）

|

HttpServlet -- 抽象类（孙子）（真正开发中常用这个）

2.GenericServlet: 实现了servlet接口，是一个抽象类，将Servlet接口中其他的方法做了默认空实现，只将service()方法作为抽象;将来定义Servlet类时，可以继承GenericServlet，实现service()方法即可(大多数情况中只用到service（）方法）

- 以下四个图片是GenericServlet类中对其他四个方法的空实现

```
public void destroy() {  
}
```

```
public void init() throws ServletException {  
}
```

```
public String getServletInfo() { return ""; }
```

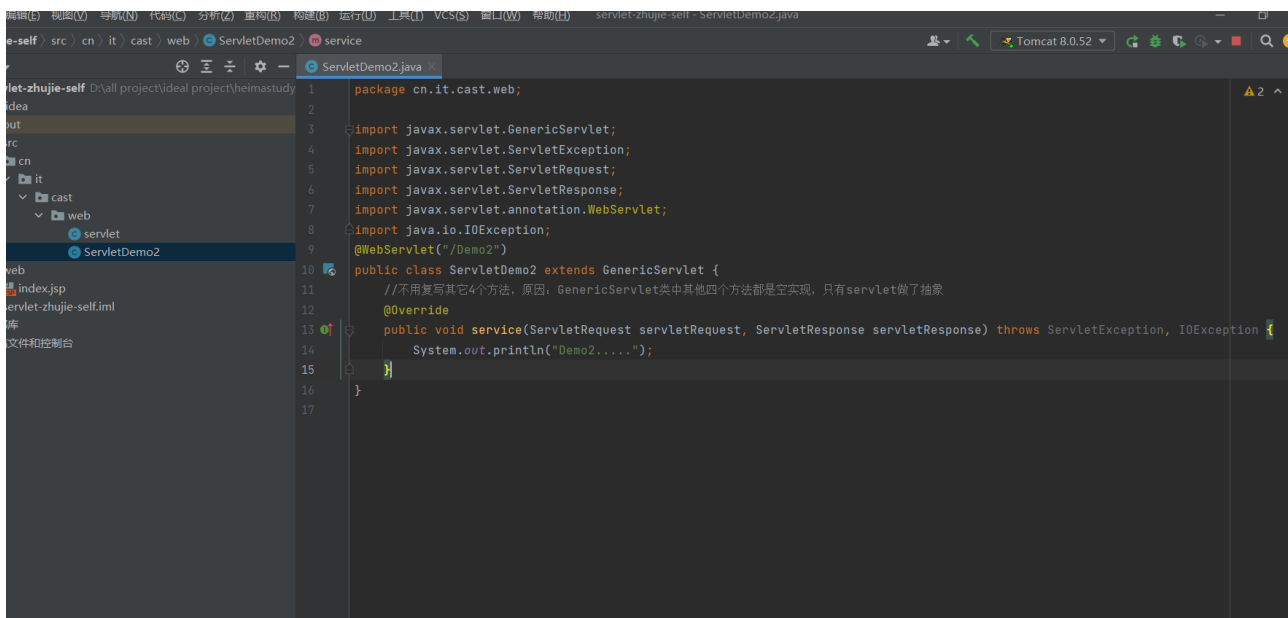
```
public ServletConfig getServletConfig() { return this.config; }
```

- 以下图片是GenericServlet类中对service()方法的抽象实现

```
public abstract void service(ServletRequest var1, ServletResponse var2) throws ServletException, IOException;
```

- 在servlet-zhujie-self里新建ServletDemo2，继承GenericServlet，并实现如下service方法，不用复写其它4个方法，原因：GenericServlet类中其他四个方法都是空实现，只有servlet做了抽象如下为代码；也可以用注解来；然后也可以通过访问localhost:8080/servlet_zhujie_self_war_exploded/Demo2开使控制台输出

注：如果想在ServletDemo2利用其他方法，只需手动添加复写方法即可



3.HttpServlet: GenericServlet的子类，也是一个抽象类，对http协议（HTTP协议一共有7种请求）的一种封装与描述，简化操作

- 使用步骤：定义类继承HttpServlet;复写doGet/doPost方法（不需要再复写service方法）
- 图片讲解：比如我们通过一个表单发送了一些数据（比如username、password），这些数据可以传送到我们的servlet，在servlet里面可以通过service方法里面的一些操作可以来获取这些数据，所以在service里的第一件事就是获取数据； 表单的提交方式常用的有两种（get方式与post方式），也就是说get方式与post方式它们封装数据的位置与格式是不一样的，也就是说在获取数据之前需要先做一个操作来判断请求方式，判断的方法：有一个对象req，它有方法，会返回一个是get还是post的字符串，然后通过if与来判断，然后通过get或者post方式来解析获取数据； 然后通过不同的请求方式做出不同的代码逻辑，这个过程是非常麻烦的,而且这个过程是所有service方法里面都需要做的； 所以sun公司就想了一

个HttpServlet类，它可以帮助我们把这个过程做好，以后不用再去判断这个请求方式，他帮我们把每一个service里面的代码给写好了，即下面图片中的

HttpServlet

```
service

    //判断请求方式
    String method = req.getMethod();
    if("GET".equals(method)){
        //get方式获取数据
        doGet();
    }else if ("POST".equals(method)){
        //post方式获取数据
        doPost();
    }

    doGet() {}
    doPost() {}
```

- 以下是HttpServlet类中的service方法源码（这代码里面有7种判断，也证实了HTTP协议一共有7种请求，但是我们只需要关心get与post就行了）

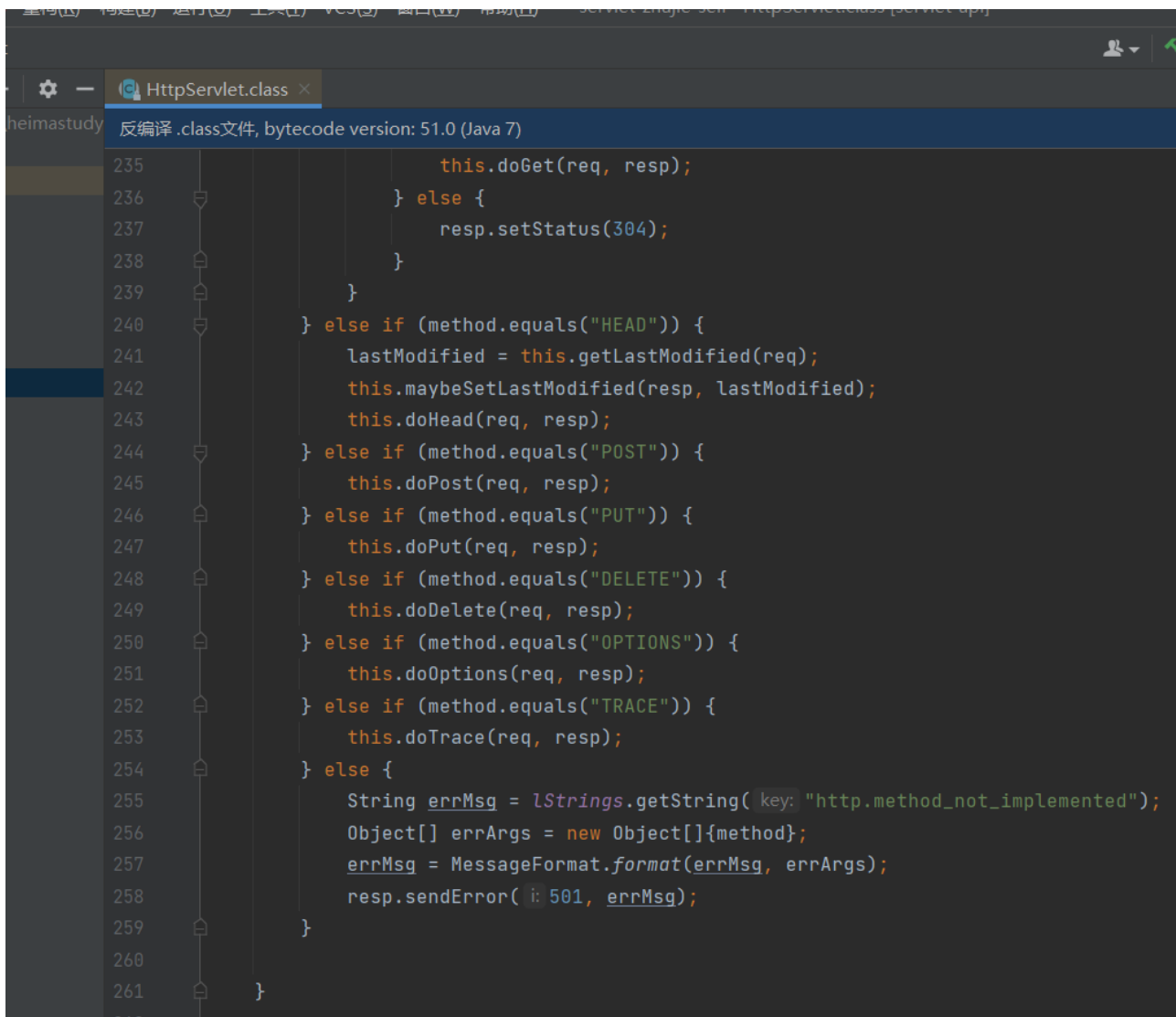
HttpServlet.class ×

反编译 .class 文件, bytecode version: 51.0 (Java 7)

```

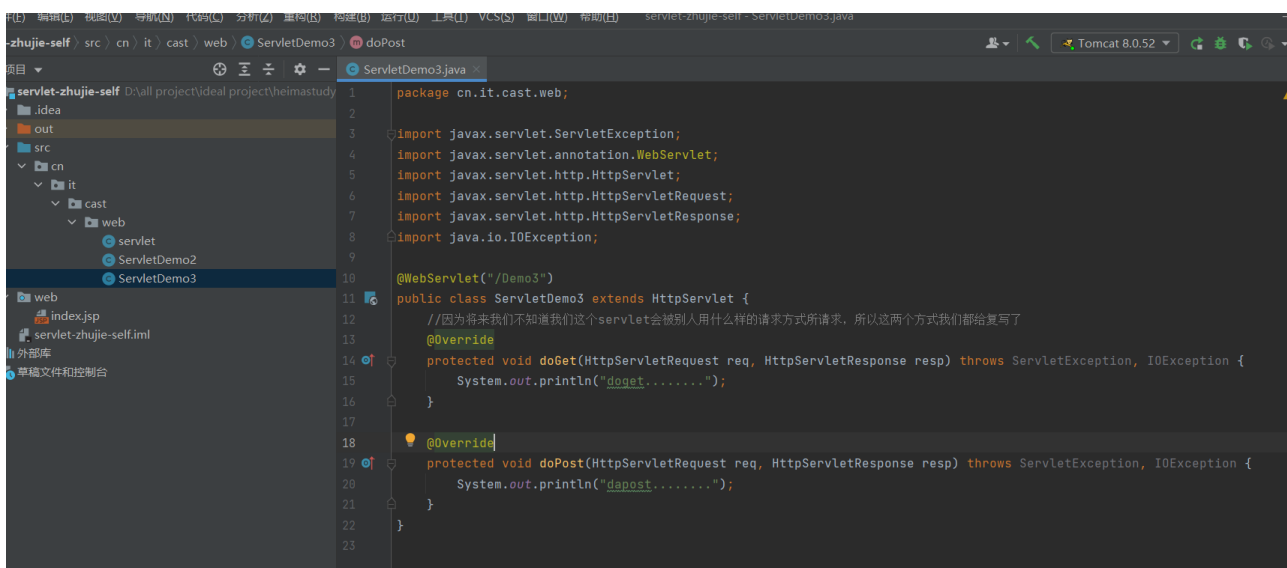
217
218     protected void service(HttpServletRequest req, HttpServletResponse resp) thro
219         String method = req.getMethod();
220         long lastModified;
221         if (method.equals("GET")) {
222             lastModified = this.getLastModified(req);
223             if (lastModified == -1L) {
224                 this.doGet(req, resp);
225             } else {
226                 long ifModifiedSince;
227                 try {
228                     ifModifiedSince = req.getDateHeader("If-Modified-Since");
229                 } catch (IllegalArgumentException var9) {
230                     ifModifiedSince = -1L;
231                 }
232
233                 if (ifModifiedSince < lastModified / 1000L * 1000L) {
234                     this.maybeSetLastModified(resp, lastModified);
235                     this.doGet(req, resp);
236                 } else {
237                     resp.setStatus(304);
238                 }
239             }
240         } else if (method.equals("HEAD")) {
241             lastModified = this.getLastModified(req);
242             this.maybeSetLastModified(resp, lastModified);
243             this.doHead(req, resp);
244         } else if (method.equals("POST")) {
245             this.doPost(req, resp);
246         } else if (method.equals("PUT")) {

```



```
反编译 .class文件, bytecode version: 51.0 (Java 7)
235         this.doGet(req, resp);
236     } else {
237         resp.setStatus(304);
238     }
239 }
240 } else if (method.equals("HEAD")) {
241     lastModified = this.getLastModified(req);
242     this.maybeSetLastModified(resp, lastModified);
243     this.doHead(req, resp);
244 } else if (method.equals("POST")) {
245     this.doPost(req, resp);
246 } else if (method.equals("PUT")) {
247     this.doPut(req, resp);
248 } else if (method.equals("DELETE")) {
249     this.doDelete(req, resp);
250 } else if (method.equals("OPTIONS")) {
251     this.doOptions(req, resp);
252 } else if (method.equals("TRACE")) {
253     this.doTrace(req, resp);
254 } else {
255     String errMsg = lStrings.getString( key: "http.method_not_implemented");
256     Object[] errArgs = new Object[]{method};
257     errMsg = MessageFormat.format(errMsg, errArgs);
258     resp.sendError( 501, errMsg);
259 }
260
261 }
```

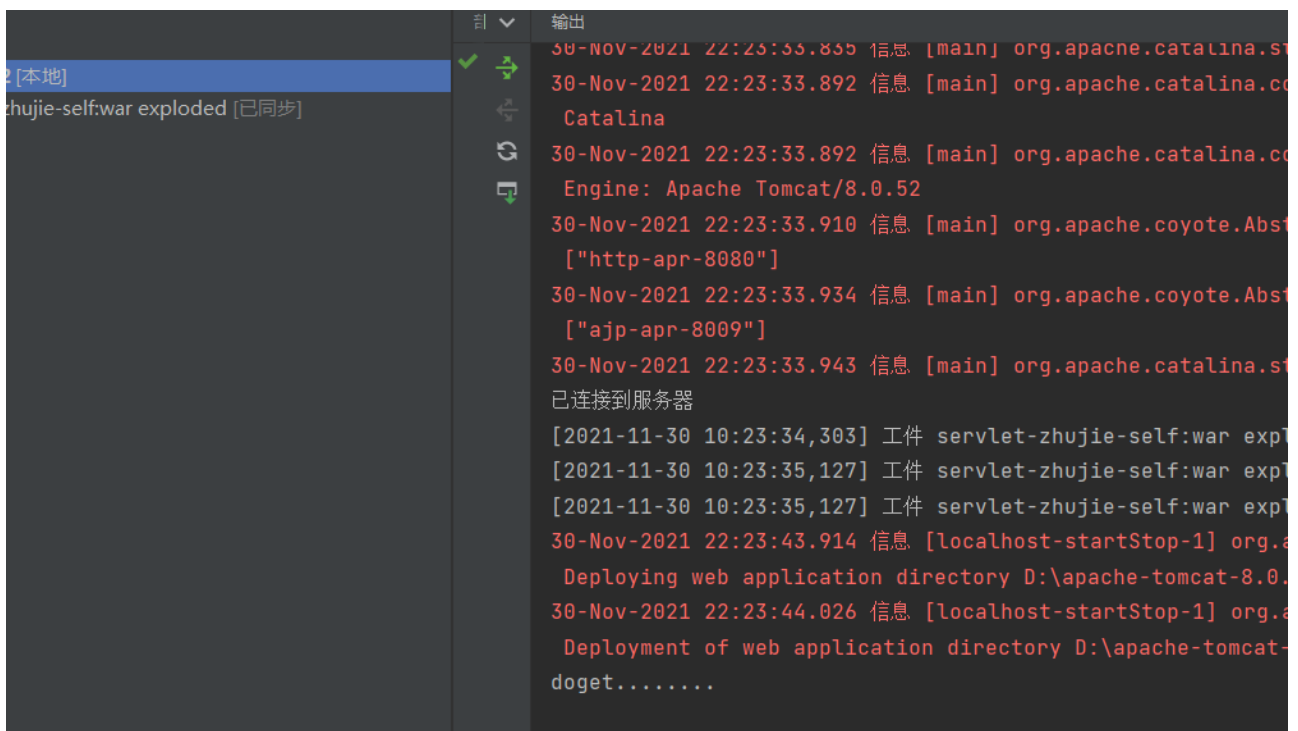
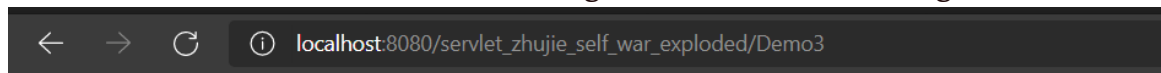
- 在servlet-zhujie-self里新建ServletDemo3，继承HttpServlet，并如下复写doget与dopost方法（继承类之后，ServletDemo3依旧是空的，所以需要我们输入doGet根据提示按回车进行快速复写，dopost也是这样）；也可以用注解来；



```
ServletDemo3.java
1 package cn.it.cast.web;
2
3 import javax.servlet.ServletException;
4 import javax.servlet.annotation.WebServlet;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8 import java.io.IOException;
9
10 @WebServlet("/Demo3")
11 public class ServletDemo3 extends HttpServlet {
12     //因为将来我们不知道我们这个servlet会被别人用什么样的请求方式所请求，所以这两个方式我们都给复写了
13     @Override
14     protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
15         System.out.println("doget.....");
16     }
17
18     @Override
19     protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
20         System.out.println("dopost.....");
21     }
22 }
23
```

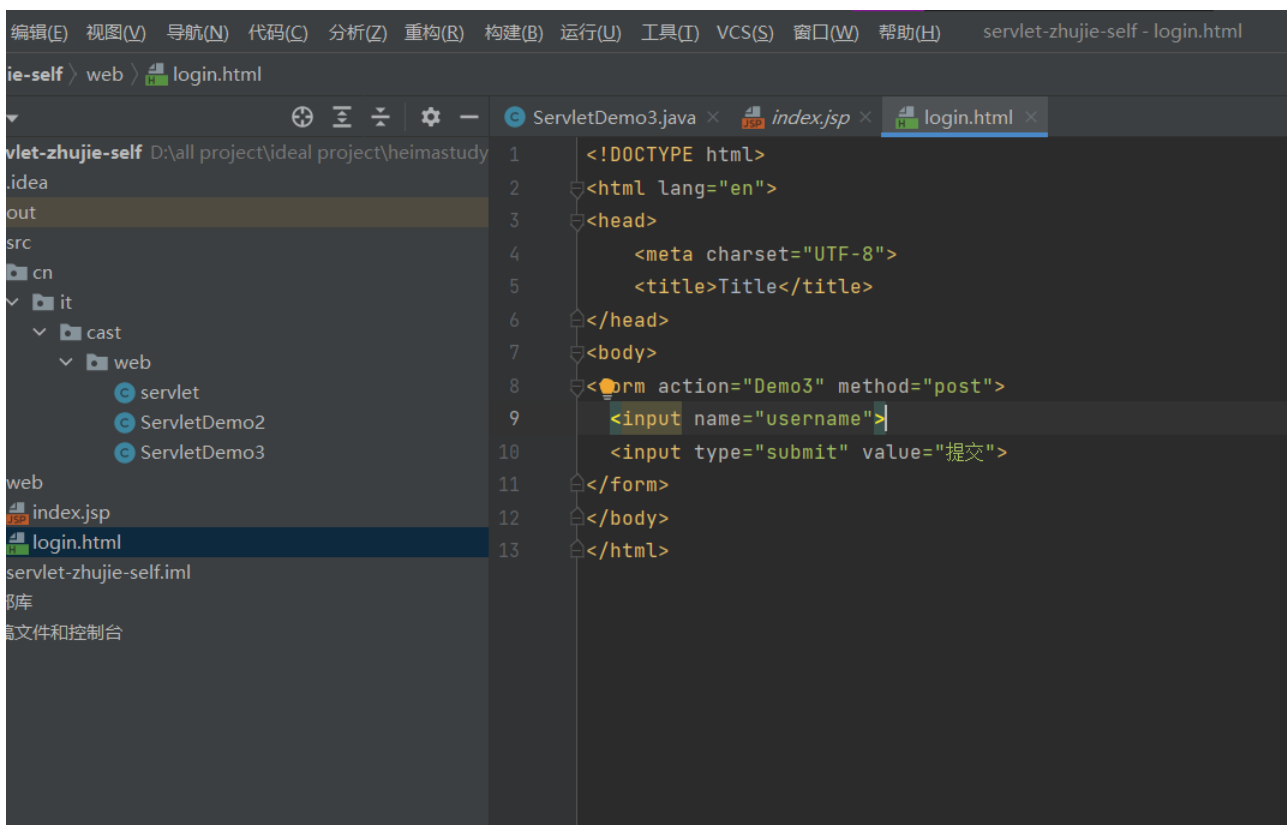
- 启动tomcat，在浏览器输入
localhost:8080/servlet_zhujie_self_war_exploded/Demo3，仍然是空白的，我们

自己通过浏览器直接输入网页进行访问时get方式，控制端输出“doget.....”



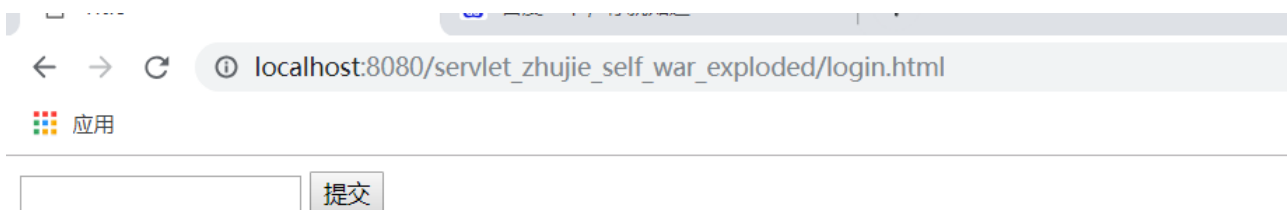
- 现在所掌握的知识只能通过表单来完成post的请求，右键web---新建----HTML文件---命名login

注：action里填的是虚拟目录（没有斜杠/）



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8     <form action="Demo3" method="post">
9         <input name="username">
10        <input type="submit" value="提交">
11    </form>
12 </body>
13 </html>
```

- 输入http://localhost:8080/servlet_zhujie_self_war_exploded/login.html之后会弹出以下页面，但是此时idea控制台不会输出“dopost”



- 随便输入一串，然后点击提交（需要在谷歌浏览器当中进行提交，不能在microsoft里），网页会跳转到http://localhost:8080/servlet_zhujie_self_war_exploded/Demo3(但依旧是同一标签，不会是新增一标签)，此时网页是空白页，控制台会输出“dopost”



localhost:8080/servlet_zhujie_self_war_exploded/login.html

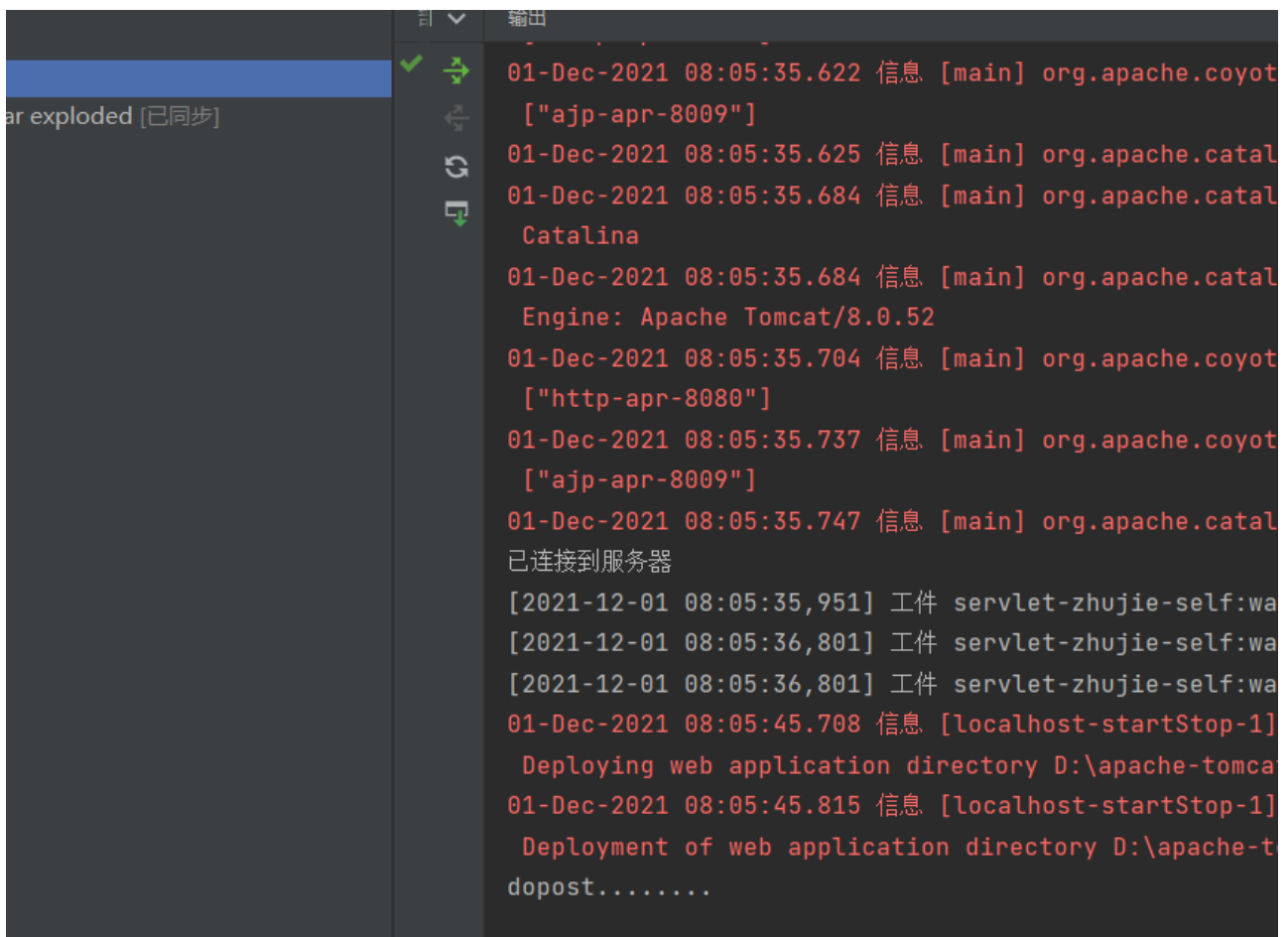
应用

aaa



localhost:8080/servlet_zhujie_self_war_exploded/Demo3

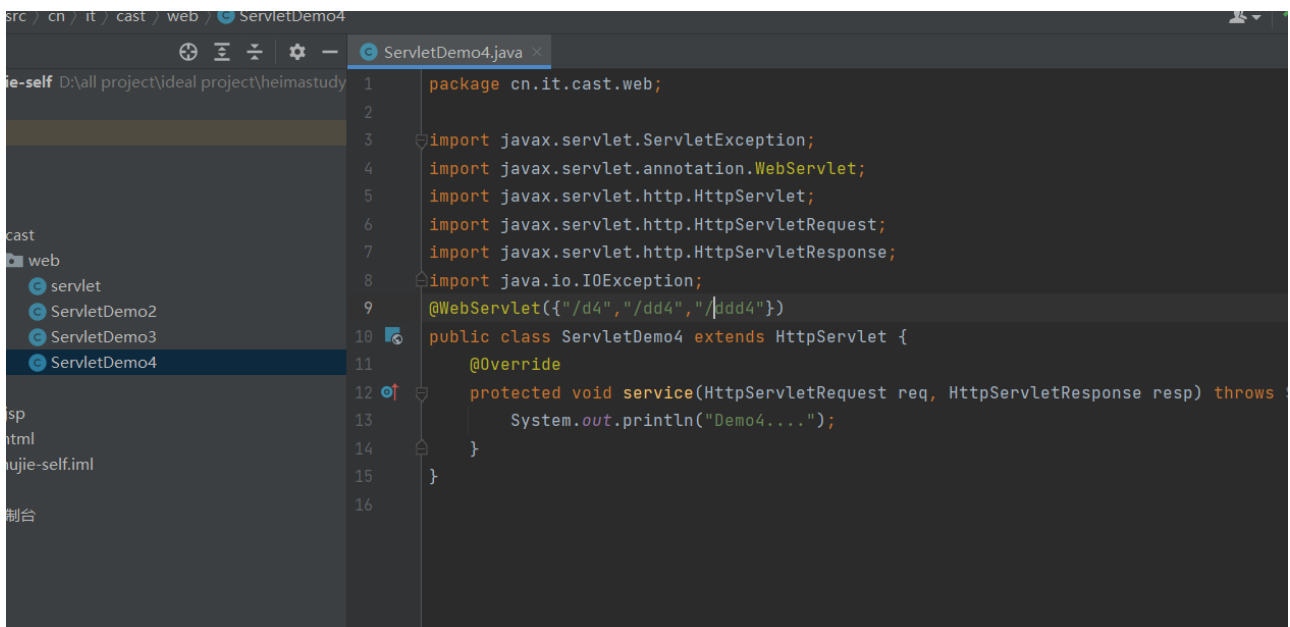
应用



```
01-Dec-2021 08:05:35.622 信息 [main] org.apache.coyote
["ajp-apr-8009"]
01-Dec-2021 08:05:35.625 信息 [main] org.apache.catal
01-Dec-2021 08:05:35.684 信息 [main] org.apache.catal
Catalina
01-Dec-2021 08:05:35.684 信息 [main] org.apache.catal
Engine: Apache Tomcat/8.0.52
01-Dec-2021 08:05:35.704 信息 [main] org.apache.coyote
["http-apr-8080"]
01-Dec-2021 08:05:35.737 信息 [main] org.apache.coyote
["ajp-apr-8009"]
01-Dec-2021 08:05:35.747 信息 [main] org.apache.catal
已连接到服务器
[2021-12-01 08:05:35,951] 工件 servlet-zhujie-self:war
[2021-12-01 08:05:36,801] 工件 servlet-zhujie-self:war
[2021-12-01 08:05:36,801] 工件 servlet-zhujie-self:war
01-Dec-2021 08:05:45.708 信息 [localhost-startStop-1]
Deploying web application directory D:\apache-tomcat\dopost
01-Dec-2021 08:05:45.815 信息 [localhost-startStop-1]
Deployment of web application directory D:\apache-tomcat\dopost
dopost.....
```

Servlet路径相关配置

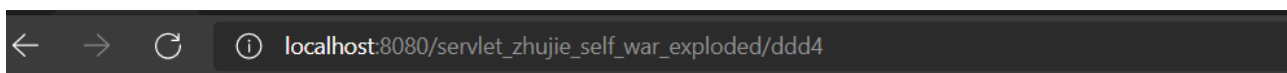
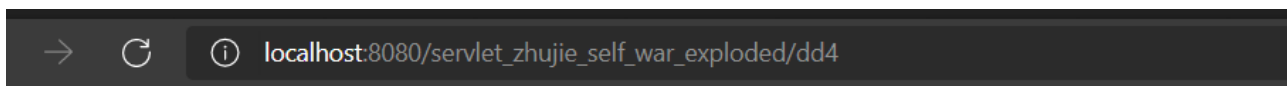
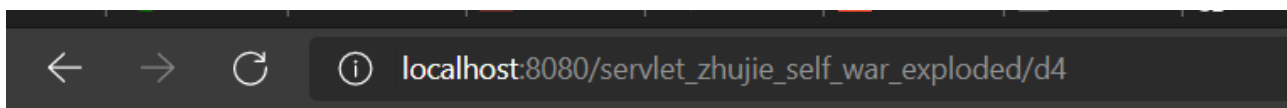
- 新建ServletDemo4，并编写



```
package cn.it.cast.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet({"/d4", "/dd4", "/add4"})
public class ServletDemo4 extends HttpServlet {
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("Demo4...");
    }
}
```



```
01-Dec-2021 08:35:16.671 信息 [main] org.apache.catalina.startup.Catalina.start Server startup in 1
已连接到服务器
[2021-12-01 08:35:17,085] 工件 servlet-zhujie-self:war exploded: 正在部署工件, 请稍候...
[2021-12-01 08:35:17,924] 工件 servlet-zhujie-self:war exploded: 工件已成功部署
[2021-12-01 08:35:17,924] 工件 servlet-zhujie-self:war exploded: 部署已花费 839 毫秒
Demo4....
01-Dec-2021 08:35:26.642 信息 [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deploy
Deploying web application directory D:\apache-tomcat-8.0.52-64\webapps\manager
01-Dec-2021 08:35:26.762 信息 [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deploy
Deployment of web application directory D:\apache-tomcat-8.0.52-64\webapps\manager has finished i
Demo4....
Demo4....
```

- urlpartten:Servlet访问路径
- 一个Servlet可以定义多个访问路径：@WebServlet({" /d4","/dd4","/ddd4"})
注：如果多个的话路径，需要加大括号；但如果一个的话，不需要加大括号
- 路径定义规则：

第一种：/xxx （路径匹配） 例如：

@WebServlet({" /d4","/dd4","/ddd4"})

```
import java.io.IOException;

@WebServlet({" /d4","/dd4","/ddd4"})
```

第二种：/xxx/xxx/xxx/xxx/xxx/~（多层路径，目录结构） 例如：

@WebServlet("/user/demo5")

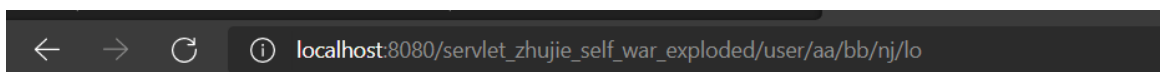
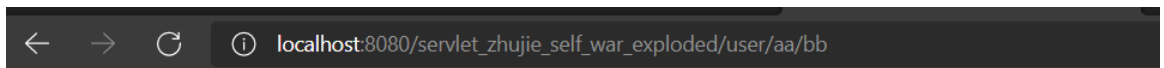
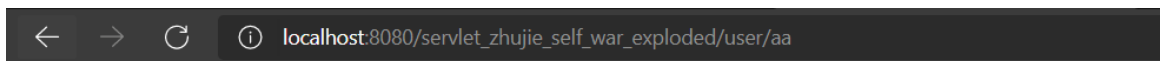
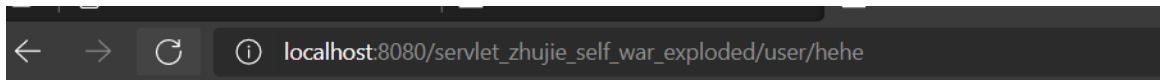
```
@WebServlet("/user/demo5")
```



第三种：/xxx/* （*是通配符，后面不管写什么都可以，不管再写几层路径都可以）

例如： @WebServlet("/user/*")

```
@WebServlet("/user/*")
```



第四种：*.do （不要加 /，*代表所有，do是后缀名，扩展名匹配，.do前不管写什么都可以，不管再写几层路径都可以）

```
3  import ...
9
10 @WebServlet("*.do")
11 public class ServletDemo7 extends HttpServlet {
12     @Override
13     protected void service(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
14         System.out.println("Demo7....");
15     }
16 }
17
```

localhost:8080/servlet_zhujie_self_war_exploded/411.do

localhost:8080/servlet_zhujie_self_war_exploded/444/assda/ada/dp.do

```
[2021-12-01 09:38:21,044] 工件 ser
[2021-12-01 09:38:21,044] 工件 ser
01-Dec-2021 09:38:29.710 信息 [loc
    Deploying web application direct
01-Dec-2021 09:38:29.846 信息 [loc
    Deployment of web application di
Demo7....
Demo7....
```