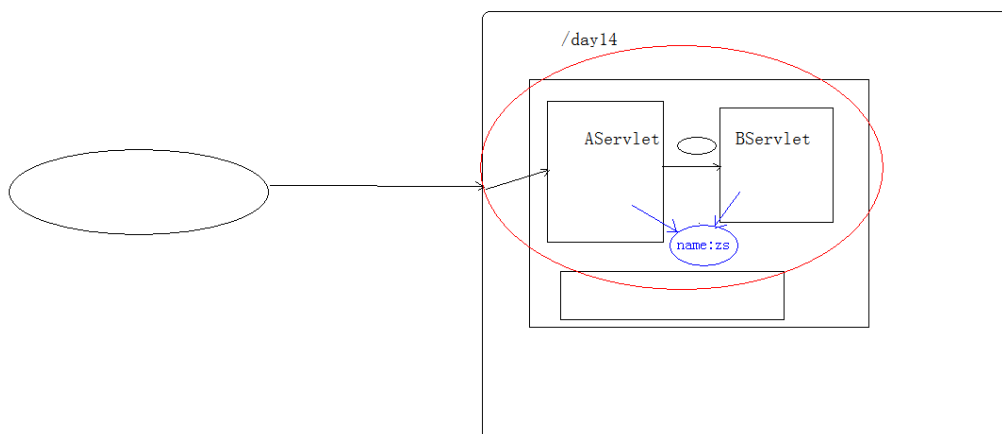
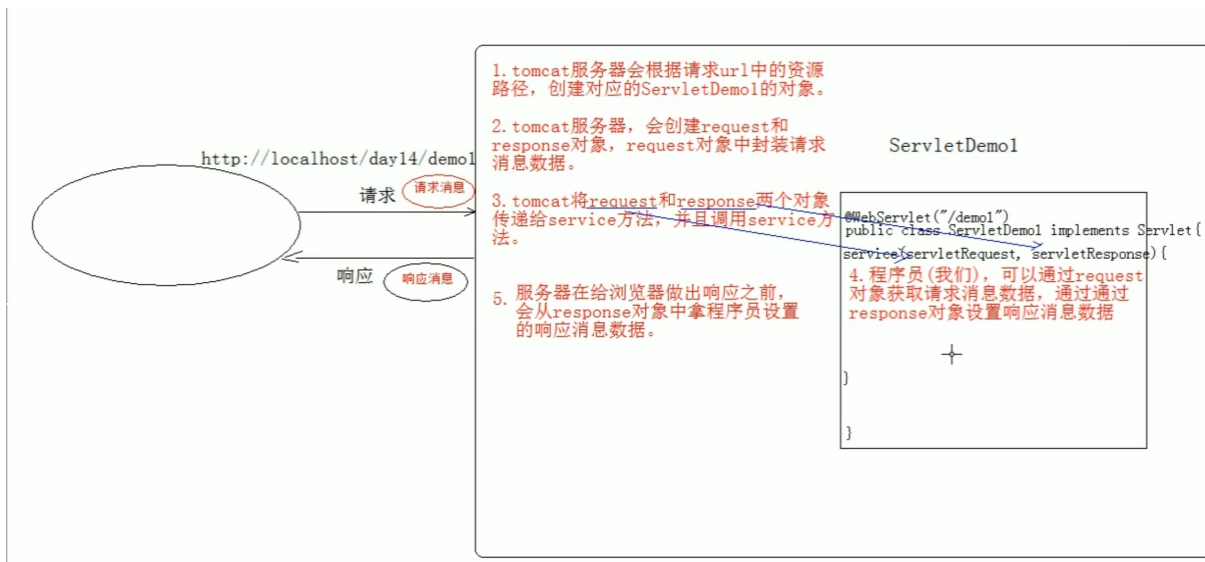
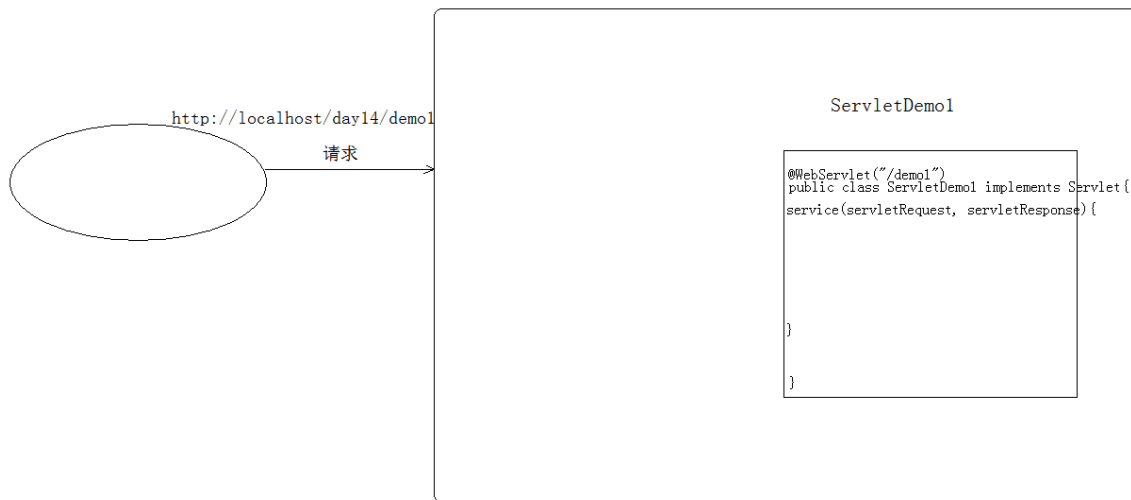


Request方法:



```

4
5 public class ServletDemo1 implements Servlet {
6
7     @Override
8     public void service(ServletRequest servletRequest, ServletResponse servletResponse) throws ServletException, IOException {
9         System.out.println("提供服务的方法-----HelloServlet");
10    }

```

```

public class ServletDemo extends GenericServlet {
    @Override
    public void service(ServletRequest servletRequest, ServletResponse servletResponse) throws ServletException, IOException {
    }
}

```

```

@WebServlet("/Http2")
5 public class HttpServlet2 extends HttpServlet {
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("Http2.....");
    }
}

```

注： **HTTPServlet**也是可以实现**service**方法的

1. request对象和response对象的原理

1. request和response对象是由服务器创建的。我们来使用它们
2. request对象是来获取请求消息，response对象是来设置响应消息

2. request对象继承体系结构:

```

ServletRequest      -- 接口
    |  继承
HttpServletRequest  -- 接口
    |  实现

```

org.apache.catalina.connector.RequestFacade 类(tomcat编写的类（可以在官网下载个tomcat源码包，进行查看），实现了HttpServletRequest接口，tomcat将来会通过这个类来创建request对象，并且传递给service方法)

3. request功能:

(1) 获取请求消息数据

1. 获取"请求行"数据方法

举例get方式表单提交后请求行格式： GET /day14/demo1?

name=zhangsan HTTP/1.1

1. 获取请求方式：GET

* String getMethod()

2. 获取虚拟目录：/day14

* String getContextPath()

3. 获取Servlet路径：/demo1

```

        * String getServletPath()
4. 获取get方式请求参数: name=zhangsan
        * String getQueryString()
5. 获取请求URI: /day14/demo1
        * String getRequestURI():          /day14/demo1
        * StringBuffer
getRequestURL():http://localhost/day14/demo1
        * URL:统一资源定位符 :
http://localhost/day14/demo1      中华人民共和国
        * URI: 统一资源标识符 : /day14/demo1      共和国（不仅
指中国，可以指很多国家）
6. 获取协议及版本: HTTP/1.1
        * String getProtocol()
7. 获取客户机的IP地址:
        * String getRemoteAddr()
        *如果是自己访问自己，则获得自己的IP: 127.0.0.1;
          如果是别人访问自己的网页，则获得的是她的ip

```

(2). 获取"请求头"数据

```

* 方法:
    * String getHeader(String name):通过传入请求头的名称获取
请求头的值
    * Enumeration<String> getHeaderNames():获取所有的请求
头名称
(Enumeration是一个接口，是一个枚举，跟我们之前所了解的枚举不是一回事，
但我们可以把它理解为迭代器)

```

```

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //以下不能写在doPost里面，且不用写login.html,网页搜索localhost:8080/RequestSelf_war_exploded/requestDemo1?name=zhangsan即可看到控制台输出
    // 不需要借助html
    String method = request.getMethod();
    System.out.println(method);// GET

    String contextPath = request.getContextPath();
    System.out.println(contextPath);// /RequestSelf_war_exploded

    String servletPath = request.getServletPath();
    System.out.println(servletPath);// /requestDemo1

    String queryString = request.getQueryString();
    System.out.println(queryString);// name=zhangsan

    String requestURI = request.getRequestURI();
    StringBuffer requestURL = request.getRequestURL();
    System.out.println(requestURI);// /RequestSelf_war_exploded/requestDemo1
    System.out.println(requestURL);// http://localhost:8080/RequestSelf_war_exploded/requestDemo1

    String protocol = request.getProtocol();
    System.out.println(protocol);// HTTP/1.1

    String remoteAddr = request.getRemoteAddr();
    System.out.println(remoteAddr);// | 127.0.0.1
}

```

```

18
19 ① @ protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
20    //1. 获取所有请求头名称
21    Enumeration<String> headerNames = request.getHeaderNames();
22    //2. 遍历
23    while(headerNames.hasMoreElements()){
24        String name = headerNames.nextElement();
25        //根据名称获取请求头的值
26        String value = request.getHeader(name);
27        System.out.println(name+"---"+value);
28    }
29
30 }

```

```

Deployment of web application directory C:\apache-tomcat-8.5.92\webapps\manager has finished in 119 ms
host---localhost:8080
connection---keep-alive
sec-ch-ua---" Not A;Brand";v="99", "Chromium";v="96", "Microsoft Edge";v="96"
sec-ch-ua-mobile---?0
sec-ch-ua-platform---"Windows"
upgrade-insecure-requests---1
user-agent---Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.55 Safari/537.36 Edg/96.0.1054.43
accept---text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8, application/signed-exchange;v=b3;q=0.9
sec-fetch-site---none
sec-fetch-mode---navigate
sec-fetch-user---?1
sec-fetch-dest---document
accept-encoding---gzip, deflate, br
accept-language---zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
cookie---JSESSIONID=E11AB5FCE5A1B56C4778BB58ABDFE4E0; _ga=GA1.1.593866488.1637580990

```

(3) 获取"请求体"数据:

* 请求体: 只有POST请求方式, 才有请求体, 在请求体中封装了POST请求的请求参数

* 步骤:

1. 获取流对象(在request对象, 它对请求体的数据将其封装为流)

* **BufferedReader getReader():** 获取字符输入流, 只能操作字符数据

(字符数据: 比如username=张三, password=123, 普通的表单提交的都是字符的)

* **ServletInputStream getInputStream():** 获取字节输入流, 可以操作所有类型数据

* 在文件上传知识点后讲解, 暂时不讲解

(字节输出流: 比如文件的上传图片、视频等)

(把ServletInputStream 当成InputStream来用, 因为它继承了InputStream)

2. 再从流对象中拿数

4. request其他功能

(1) 获取请求参数通用方式：不论get还是post请求方式都可以使用下列方法来获取请求参数

(之前获取请求参数username=zs&password=123时get是用
getQueryString();post是用 getReader())

*String getParameter(String name):根据参数名称获取参数值

比如: username=zs&password=123

*String[] getParameterValues(String name):根据参数名称获取参
数值的数组 (返回的是一个数组)

比如: hobby=ball&hobby=game (通过一个键获取一个键的两个值,
这两个值返回成一个数组)

*Enumeration<String> getParameterNames():获取所有请求的参数
名称

(Enumeration是一个枚举)

*Map<String,String[]> getParameterMap():获取所有参数的map集
合

(返回一个map, 键是string类型的, 值是string类型的数组)

* 中文乱码问题:

* get方式: tomcat 8 已经将get方式乱码问题解决了

* post方式: 会乱码 (中文或者非法的)

* 解决: 在获取参数前, 设置request的编码

request.setCharacterEncoding("utf-8");

(2) 请求转发: 一种在服务器内部的资源跳转方式

* 步骤:

1. 通过request对象获取请求转发器对象: RequestDispatcher
getRequestDispatcher(String path)

小括号里是: 转发资源的路径, 即要访问的资源路径, 然后返回给我们
一个对象RequestDispatcher

2. 使用RequestDispatcher对象来进行转发:
forward(ServletRequest request, ServletResponse response)

* 特点:

1. 浏览器地址栏路径不发生变化
2. 只能转发到当前服务器内部资源中。
3. A请求转发B加起来仍旧算是1次请求, 不是2次请求 (可以在
f12中看到只有A请求)

(3) 共享数据:

* 域对象: 一个有作用范围的对象, 可以在范围内共享数据

* request域: 代表一次请求的范围, 一般用于请求转发的多个资源中共享
数据

* 方法:

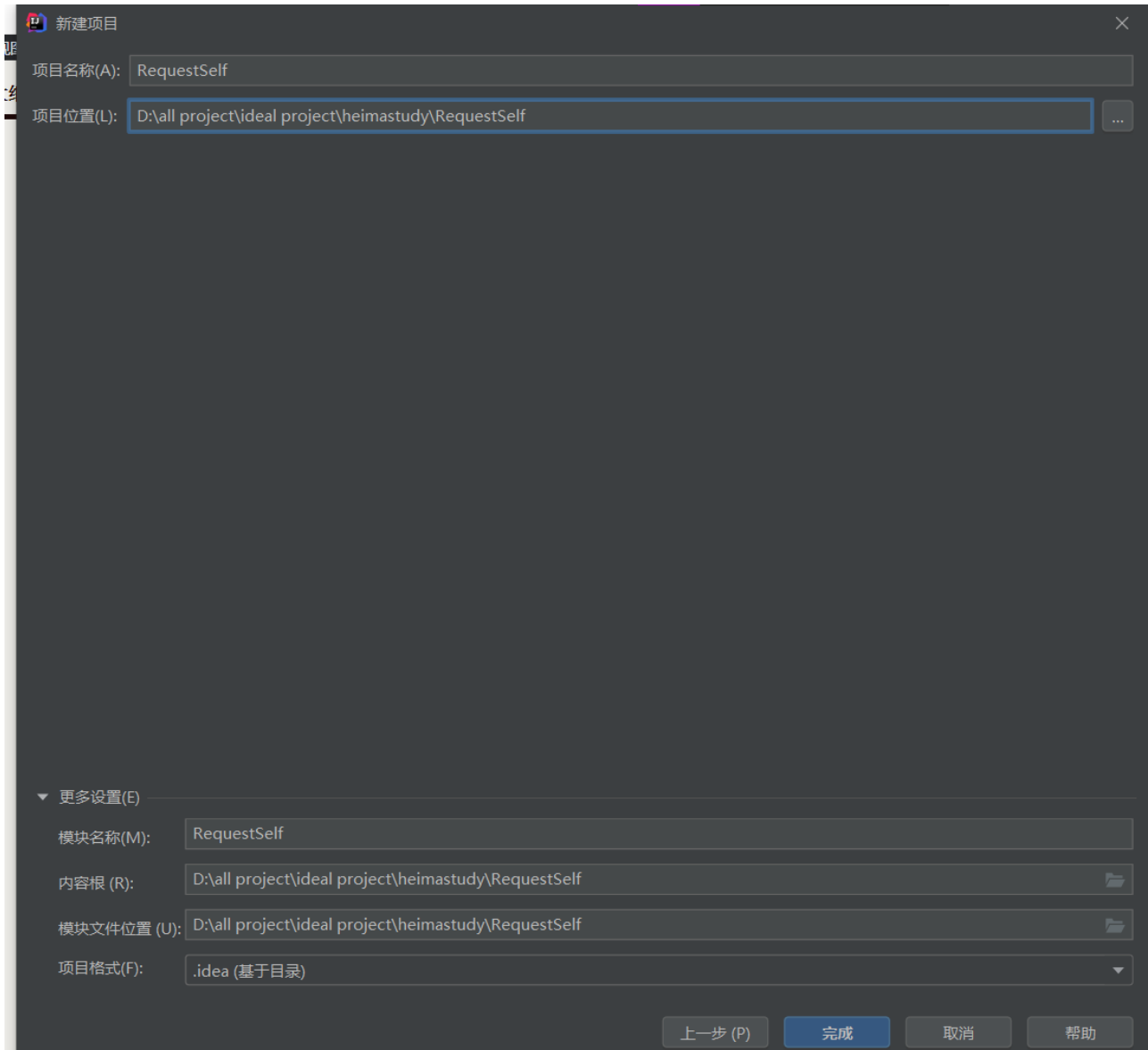
1. void setAttribute(String name, Object obj):存储数据
2. Object getAttribute(String name):通过键获取值
3. void removeAttribute(String name):通过键移除键值对

(4) 获取ServletContext: (功能和方法暂时不讲)

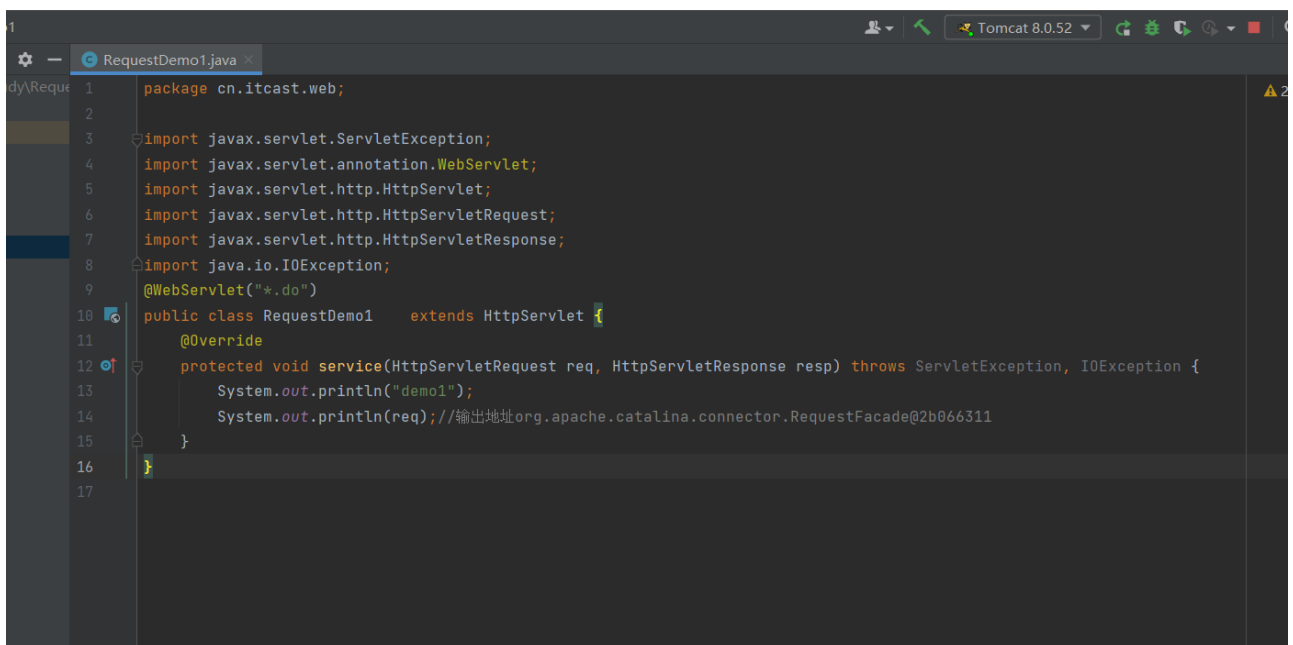
* ServletContext getServletContext()

Request在IDEA应用

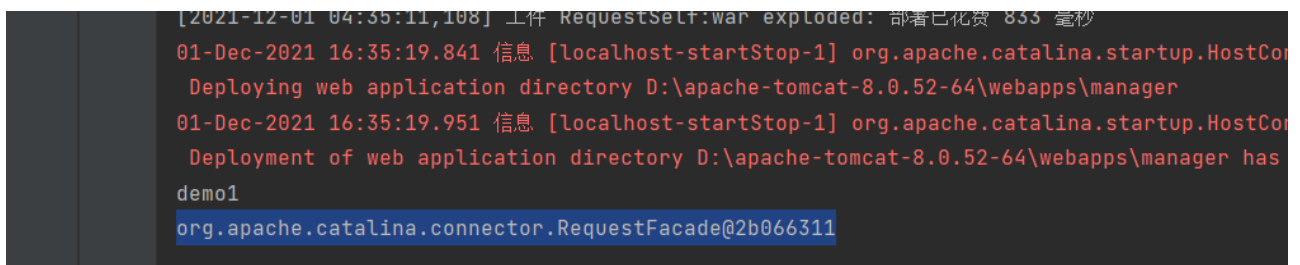
- 创建“Java”模块，及一系列tomcat配置，servlet包，框架（不要web.xml）等等



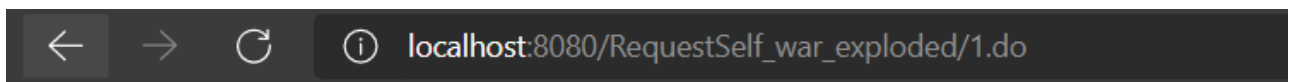
- 定义类（打印req），访问网页，并看控制台输出



```
1 package cn.itcast.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 @WebServlet("*.do")
10 public class RequestDemo1 extends HttpServlet {
11     @Override
12     protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
13         System.out.println("demo1");
14         System.out.println(req); //输出地址org.apache.catalina.connector.RequestFacade@2b066311
15     }
16 }
```

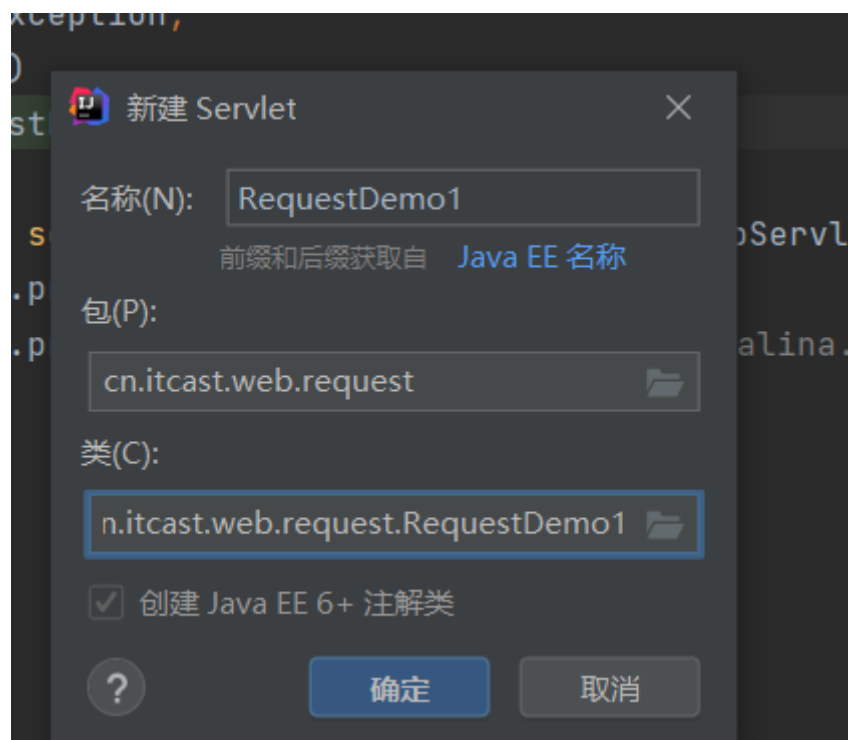
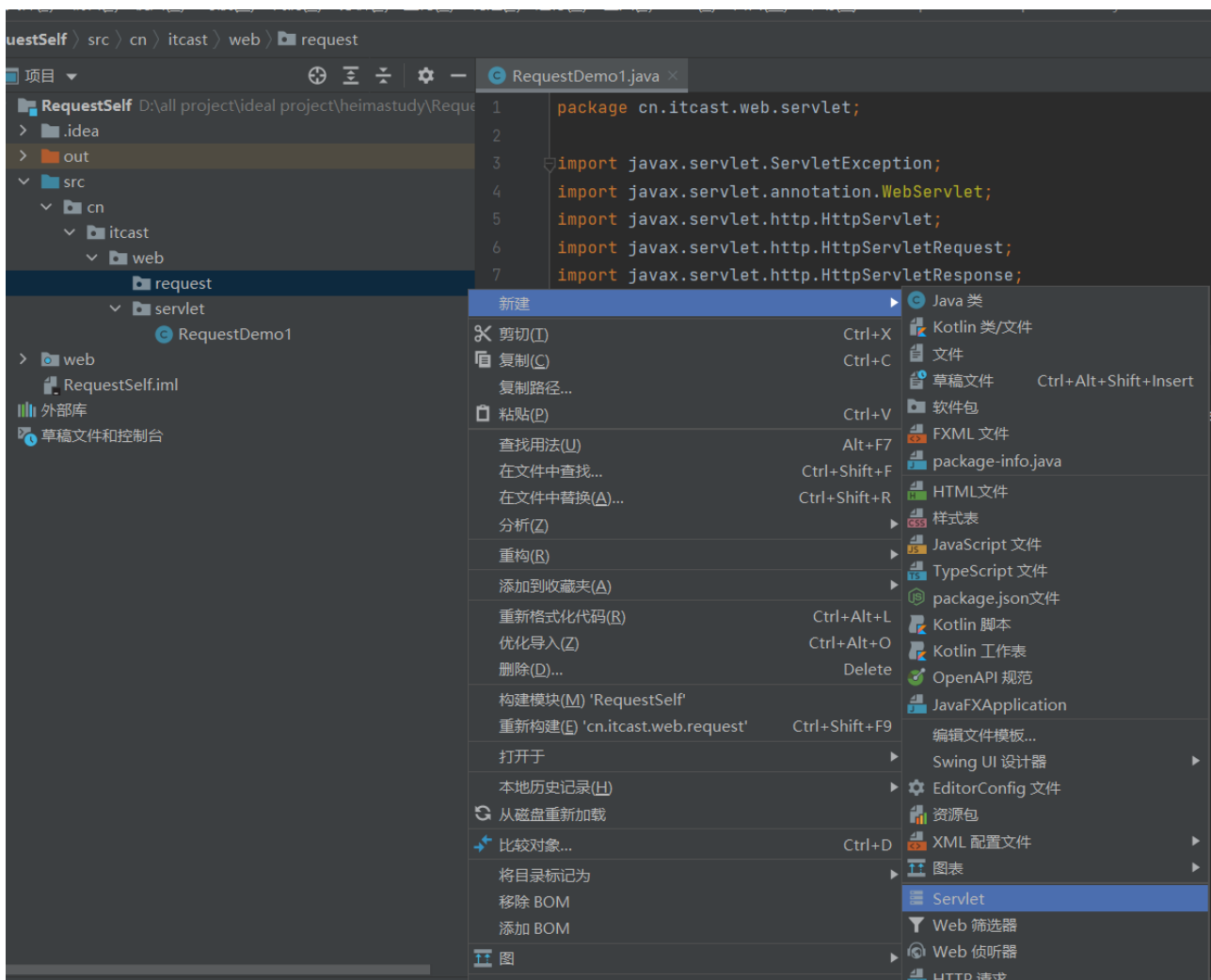


```
[2021-12-01 04:35:11,108] 工件 RequestSelf:war exploded: 部署已花费 833 毫秒
01-Dec-2021 16:35:19.841 信息 [localhost-startStop-1] org.apache.catalina.startup.HostConfig
Deploying web application directory D:\apache-tomcat-8.0.52-64\webapps\manager
01-Dec-2021 16:35:19.951 信息 [localhost-startStop-1] org.apache.catalina.startup.HostConfig
Deployment of web application directory D:\apache-tomcat-8.0.52-64\webapps\manager has
demo1
org.apache.catalina.connector.RequestFacade@2b066311
```



localhost:8080/RequestSelf_war_exploded/1.do

- 新建包“request”，然后利用快捷方式新建servlet，然后把webservlet重命名




```
1 package cn.itcast.web.request;
2
3 import ...
4
5
6
7
8 @WebServlet(name = "RequestDemo1", value = "/RequestDemo1")
9 public class RequestDemo1 extends HttpServlet {
10     @Override
11     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
12
13     }
14
15     @Override
16     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
17
18     }
19 }
20
```

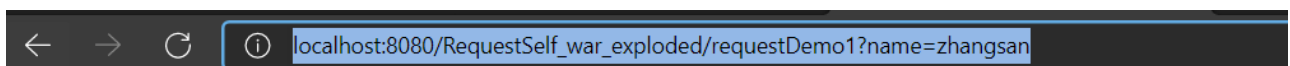
```
1 package cn.itcast.web.request;
2
3 import ...
4
5
6
7
8 @WebServlet("/requestDemo1")
9 public class RequestDemo1 extends HttpServlet {
10     @Override
11     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
12
13     }
14
15     @Override
16     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
17
18     }
19 }
20
```

- 在doget内写入以下方法（不需要写login.html）

```
RequestDemo1 > m doGet

RequestDemo1.java x
10
11 @WebServlet("/requestDemo1")
12 public class RequestDemo1 extends HttpServlet {
13     @Override
14     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
15
16         String method = request.getMethod();
17         System.out.println(method);
18
19         String contextPath = request.getContextPath();
20         System.out.println(contextPath);
21
22         String servletPath = request.getServletPath();
23         System.out.println(servletPath);
24
25         String queryString = request.getQueryString();
26         System.out.println(queryString);
27
28         String requestURI = request.getRequestURI();
29         StringBuffer requestURL = request.getRequestURL();
30         System.out.println(requestURI);
31         System.out.println(requestURL);
32
33         String protocol = request.getProtocol();
34         System.out.println(protocol);
35
36         String remoteAddr = request.getRemoteAddr();
37         System.out.println(remoteAddr);
38     }
```

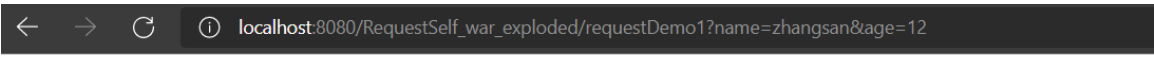
- 网页搜索localhost:8080/RequestSelf_war_exploded/requestDemo1?name=zhangsan



- 查看控制台的输出

```
GET
/RequestSelf_war_exploded
/requestDemo1
name=zhangsan
/RequestSelf_war_exploded/requestDemo1
http://localhost:8080/RequestSelf\_war\_exploded/requestDemo1
HTTP/1.1
127.0.0.1
```

- 不需要关闭tomcat，访问
localhost:8080/RequestSelf_war_exploded/requestDemo1?
name=zhangsan&age=12则输出如下：

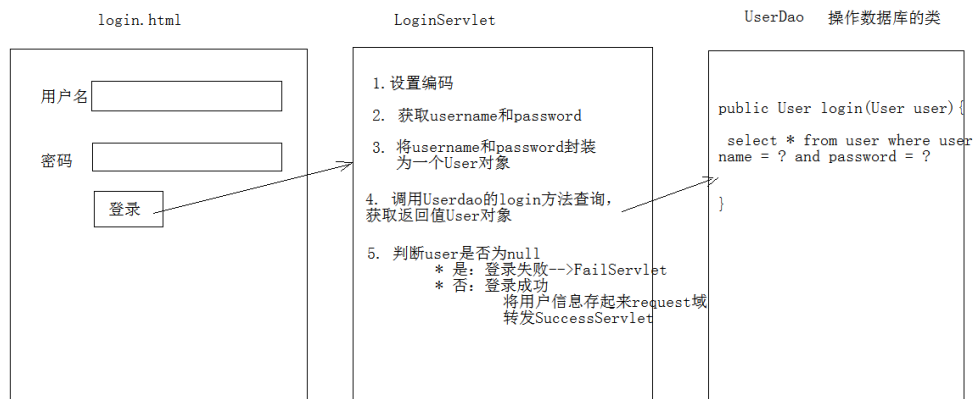


← → ↻ ⓘ localhost:8080/RequestSelf_war_exploded/requestDemo1?name=zhangsan&age=12

```
GET
/RequestSelf_war_exploded
/requestDemo1
name=zhangsan
/RequestSelf_war_exploded/requestDemo1
http://localhost:8080/RequestSelf\_war\_exploded/requestDemo1
HTTP/1.1
127.0.0.1|
GET
/RequestSelf_war_exploded
/requestDemo1
name=zhangsan&age=12
/RequestSelf_war_exploded/requestDemo1
http://localhost:8080/RequestSelf\_war\_exploded/requestDemo1
HTTP/1.1
127.0.0.1
```

- 其余方法则在代码中观看如何实现

案例：用户登录



* 用户登录案例需求:

1. 编写login.html登录页面

username & password 两个输入框

2. 使用Druid数据库连接池技术来操作mysql数据库, "day14"为数据库名, "user"为表名

3. 使用JdbcTemplate技术封装JDBC

4. 登录成功跳转到SuccessServlet展示: 登录成功! 用户名, 欢迎您

5. 登录失败跳转到FailServlet展示: 登录失败, 用户名或密码错误

* 分析

* 开发步骤

1. 创建项目, 导入html页面, 配置文件, jar包

2. 本地mysql创建数据库环境, 并增加一组数据

```
CREATE DATABASE day14;
```

```
USE day14;
```

```
CREATE TABLE USER(  
  

```

```
    id INT PRIMARY KEY AUTO_INCREMENT,  

```

```
    username VARCHAR(32) UNIQUE NOT NULL,  

```

```
    PASSWORD VARCHAR(32) NOT NULL  

```

```
);
```

```
insert into user(id,username,PASSWORD)
```

```
values(1,"superbaby","123");
```

3. src里创建cn.itcast.domain, 创建类User, 对应着User表

4. 创建包cn.itcast.util, 编写工具类JDBCUtils

5. 创建包cn.itcast.dao, 创建类UserDao, 提供login方法 (操作数据库中User表, 将来User表的增删检查都在这个类中; 如果要实现login方法还需要借助druid连接池和JdbcTemplate, druid连接池如果要用的话一般需要提供提供一个JDBC的工具类JDBCUtils, JDBCUtils用的是druid连接池)

6. 编写cn.itcast.web.servlet.LoginServlet类

7. 编写FailServlet和SuccessServlet类

8. login.html中form表单的action路径的写法

* 虚拟目录+Servlet的资源路径

9. BeanUtils工具类，简化数据封装：用于封装JavaBean的（即像User这样的，通常都放在domain下）

10. JavaBean：标准的Java类

1. 要求（跟平常的student、person其实要求一样）：

1. 类必须被public修饰

2. 必须提供空参的构造器

3. 成员变量必须使用private修饰

4. 提供公共setter和getter方法（即getUsername、

setUsername方法）

2. 功能：封装数据

3. 概念：

成员变量：

属性：*属性在大多数情况下跟成员变量是一样的：

属性指的是setter和getter方法的单词截取后的产物（即get后面的单词且把第一个单词小写）

例如：getUsername() --> Username--> username

（get方法名和方法属性名字通常跟成员变量名字一样演变大小写过来的）

*但是在有些特例下是不一样的：方法名随便起也是可以的，但最好不要随便起

```
private String gender;
    public void setHehe(String gender){
        this.gender = gender;
    }
    public String getHehe(){
        return gender;
    }
```

4. 方法：

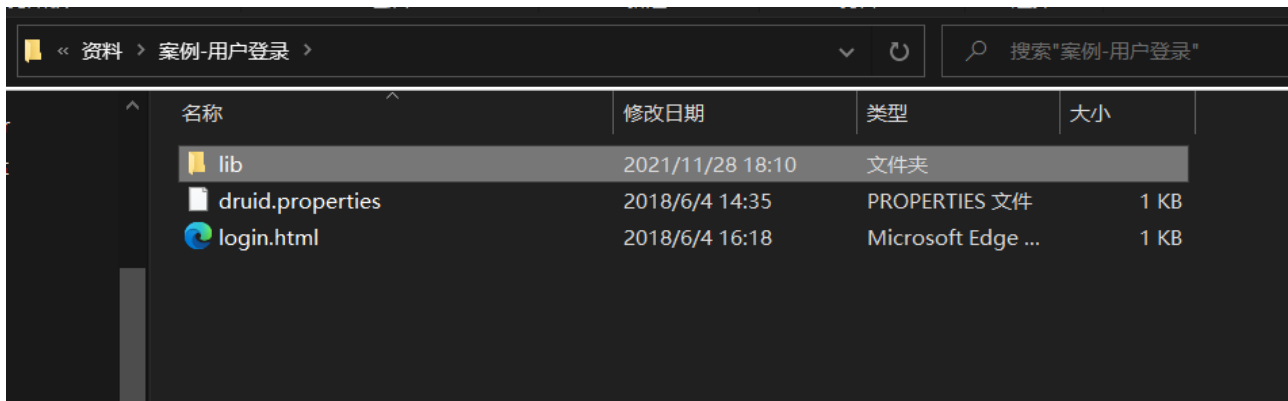
1. setProperty() -----操作的是属性，不是成员变量

2. getProperty() -----操作的是属性，不是成员变量

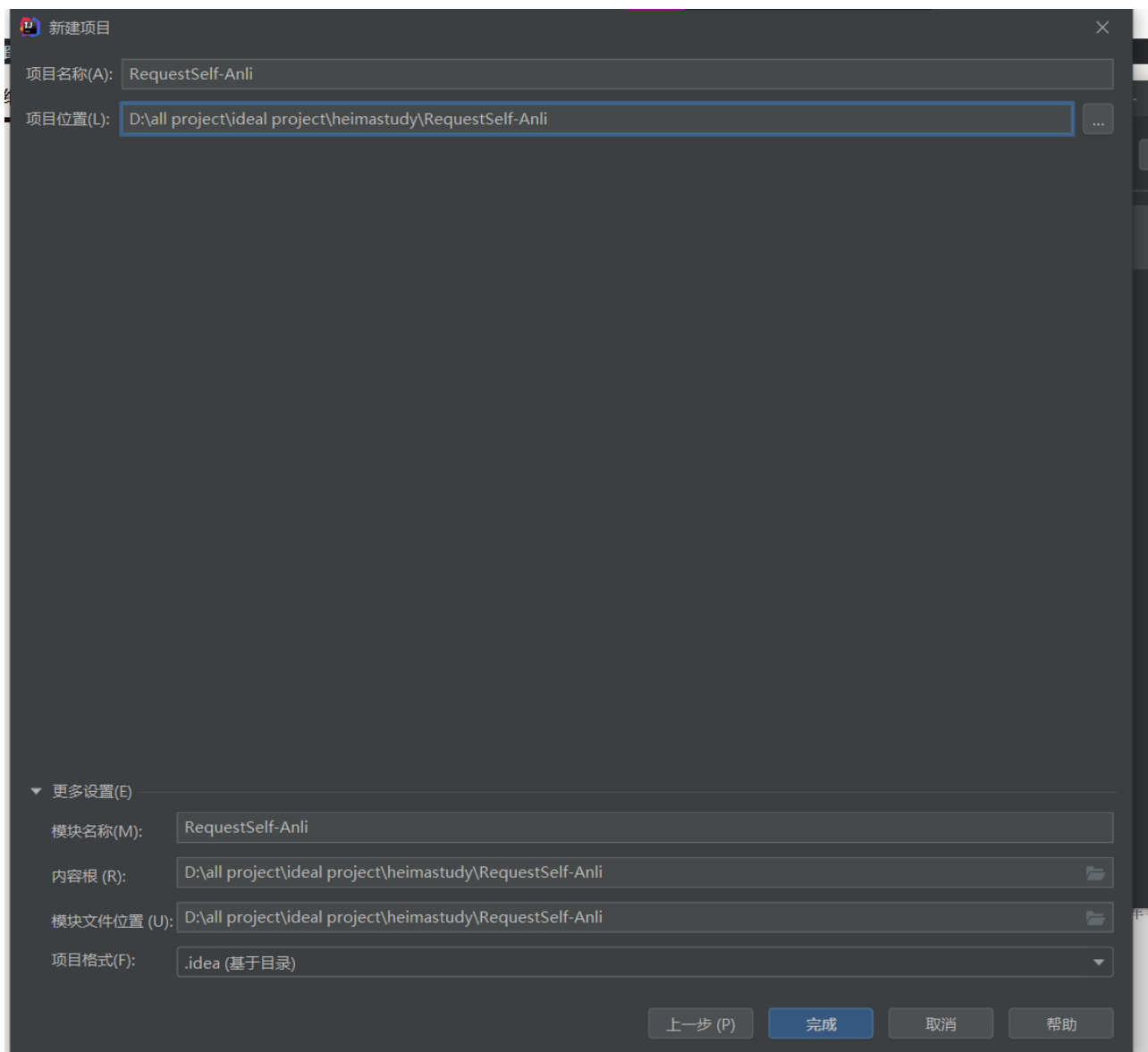
3. populate(Object obj , Map map):将map集合的键值对信息，封装到对应的JavaBean对象中

案例在IDEA中应用

- 下载的原始文件目录结构



- 创建“Java”模块，及一系列tomcat配置，servlet包，框架（不要web.xml）等等,添加tomcat中的servlet到lib

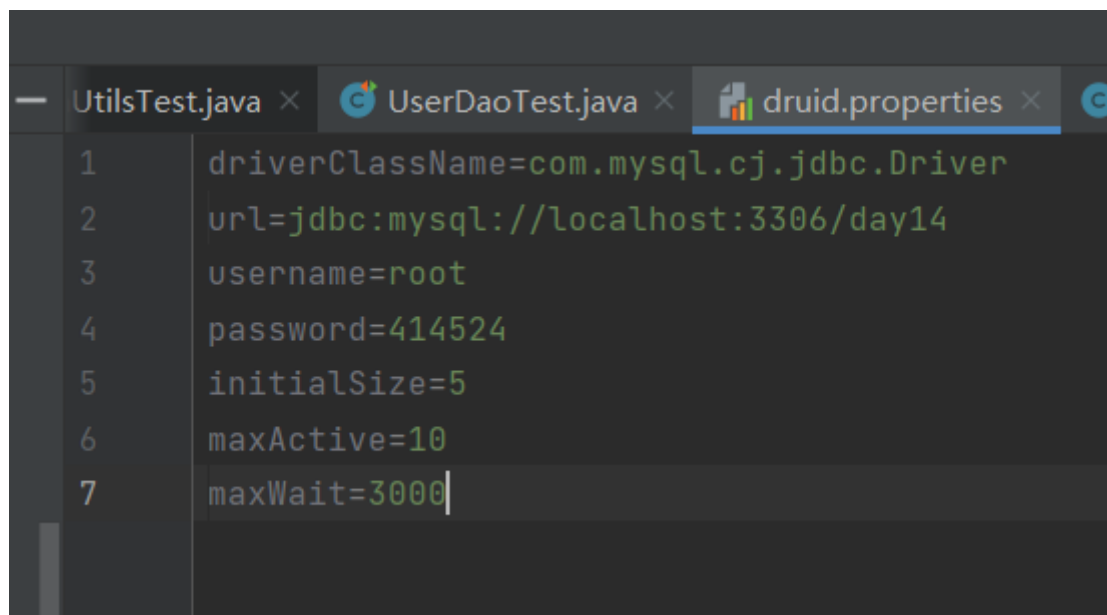


- 把login复制后直接在IDEA中粘贴到web文件夹下；

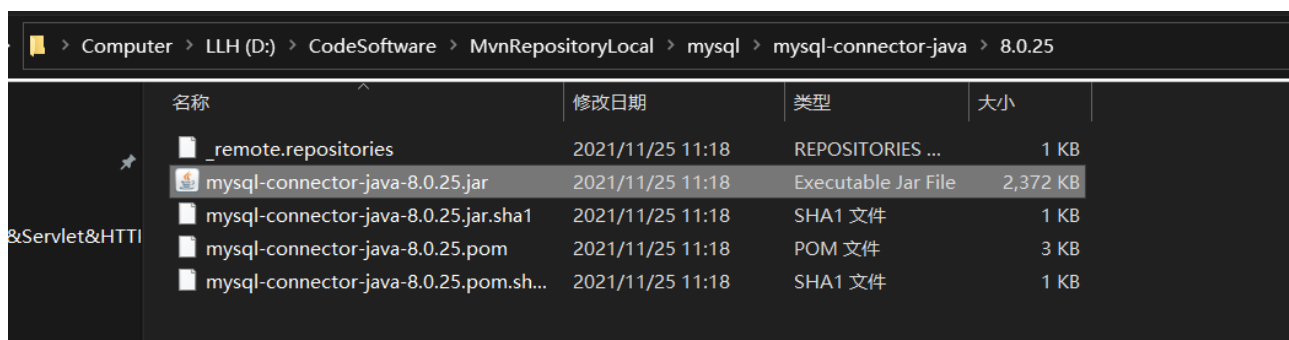
把druid.properties复制后直接在IDEA中粘贴到src文件夹下；

在web文件夹下新建WEB-INF文件夹，把lib文件夹复制后直接在IDEA中粘贴到WEB-INF文件夹下；

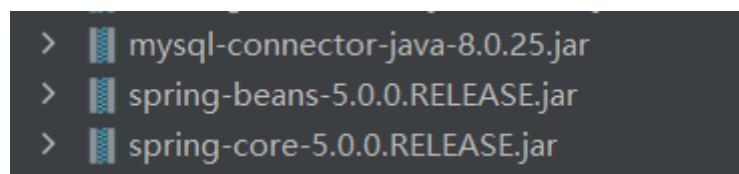
修改druid.properties中的mysql信息；并且替换mysql的connect-jar包（在本地D:\CodeSoftware\MvnRepositoryLocal\mysql\mysql-connector-java\8.0.25进行复制粘贴即可）



```
1 driverClassName=com.mysql.cj.jdbc.Driver
2 url=jdbc:mysql://localhost:3306/day14
3 username=root
4 password=414524
5 initialSize=5
6 maxActive=10
7 maxWait=3000
```



Computer > LLH (D:) > CodeSoftware > MvnRepositoryLocal > mysql > mysql-connector-java > 8.0.25				
	名称	修改日期	类型	大小
* &Servlet&HTT	_remote.repositories	2021/11/25 11:18	REPOSITORIES ...	1 KB
	mysql-connector-java-8.0.25.jar	2021/11/25 11:18	Executable Jar File	2,372 KB
	mysql-connector-java-8.0.25.jar.sha1	2021/11/25 11:18	SHA1 文件	1 KB
	mysql-connector-java-8.0.25.pom	2021/11/25 11:18	POM 文件	3 KB
	mysql-connector-java-8.0.25.pom.sh...	2021/11/25 11:18	SHA1 文件	1 KB



```
> mysql-connector-java-8.0.25.jar
> spring-beans-5.0.0.RELEASE.jar
> spring-core-5.0.0.RELEASE.jar
```

- 在本地创建数据库“day14”，里面创建表“user”，key: id, username, PASSWORD,并赋值一行


```

C:\Windows\system32\CMD.exe - mysql -uroot -p414524
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database day14
-> ;
Query OK, 1 row affected (0.03 sec)

mysql> use day14;
Database changed
mysql> CREATE TABLE USER(
->
-> id INT PRIMARY KEY AUTO_INCREMENT,
-> username VARCHAR(32) UNIQUE NOT NULL,
-> PASSWORD VARCHAR(32) NOT NULL
-> );
Query OK, 0 rows affected (0.08 sec)

mysql> show tables
-> ;
+-----+
| Tables_in_day14 |
+-----+
| user             |
+-----+
1 row in set (0.01 sec)

mysql> insert into user(it,username,PASSWORD) values(1,"superbaby","123")
-> ;
ERROR 1054 (42S22): Unknown column 'it' in 'field list'
mysql> insert into user(id,username,PASSWORD) values(1,"superbaby","123");
Query OK, 1 row affected (0.01 sec)

mysql> A_

```

- src里创建各种包，进入IDEA中继续学习