

**Java Web程序设计实训指导手册**

EmarketMall电子超市系统



**步骤一**

2022-6-13

重庆电子工程职业学院人工智能与大数据学院

软件工程系JavaWeb实训团队编制

**目 录**

[一、实验名称：EmarketMall电子超市系统 2](#_Toc106062350)

[1、相关技能 2](#_Toc106062351)

[2、相关知识点 2](#_Toc106062352)

[3、实现效果 2](#_Toc106062353)

[3.1、依赖引入 2](#_Toc106062354)

[3.2、框架结构 4](#_Toc106062355)

[3.3、用户管理运行结果 4](#_Toc106062356)

[4、实验步骤 6](#_Toc106062357)

[4.1、创建基于Gradle的JavaWeb项目 6](#_Toc106062358)

[4.2、导入所需的依赖和jar包 7](#_Toc106062359)

[4.3、在MySQL数据库管理工具中导入本项目所需的数据脚本 7](#_Toc106062360)

[4.4、创建项目属性文件，通过druid数据连接池连接 9](#_Toc106062361)

[4.5、框架基本功能实训步骤 10](#_Toc106062362)

[1）部署和运行 10](#_Toc106062363)

[2）用户登录 12](#_Toc106062364)

[3）用户管理 14](#_Toc106062365)

# 一、实验名称：EmarketMall电子超市系统

1、相关技能

* 创建基于Gradle的JavaWeb项目
* Gradle配置
* 通过Gradle方式引入Jar包依赖
* 导入数据脚本
* 初始化项目结构

2、相关知识点

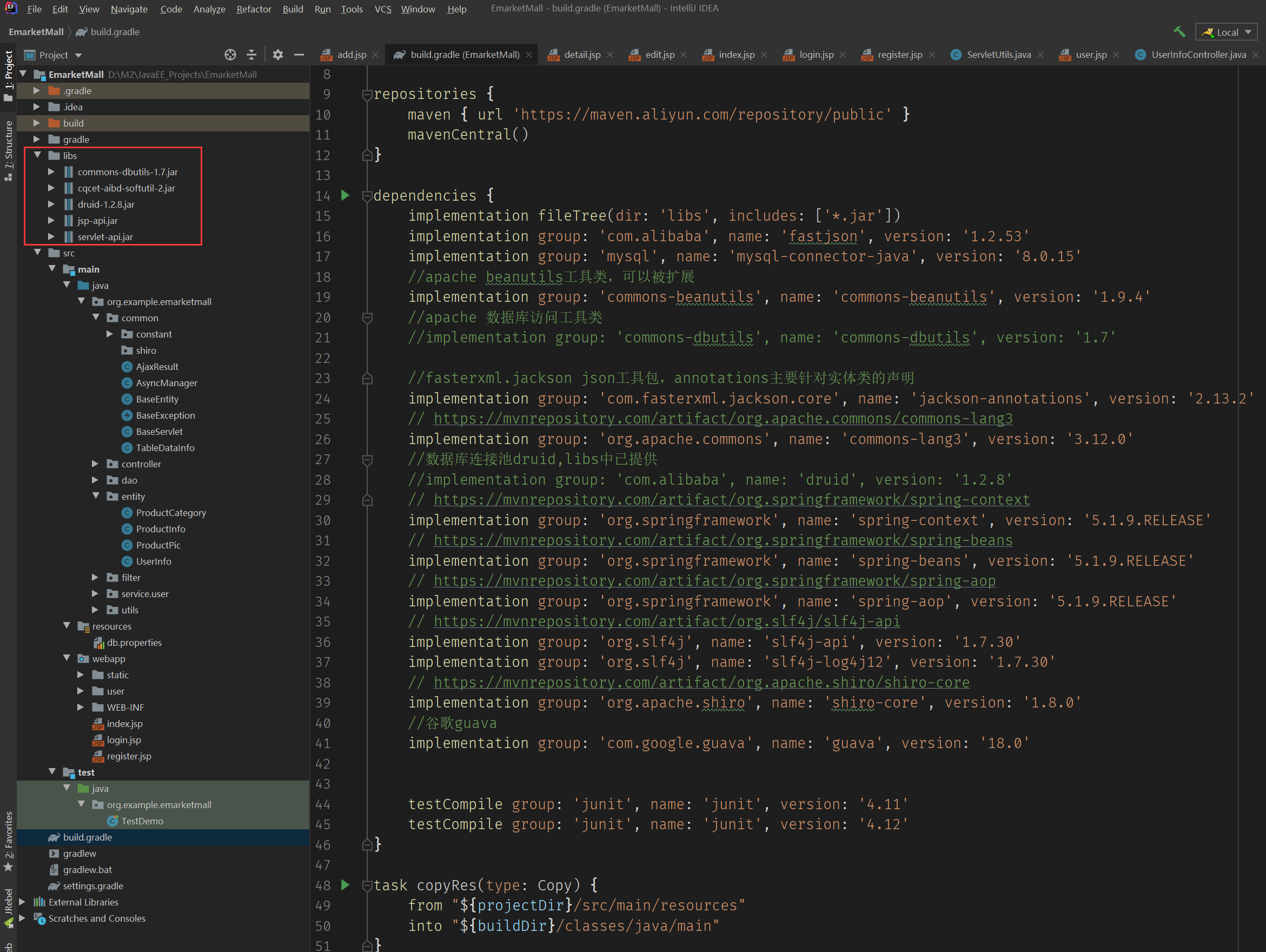
* model，dao，service，controller，view
* Build.gradle配置
* 引入依赖
* 通过MySQL数据库管理工具导入项目数据脚本

3、实现效果

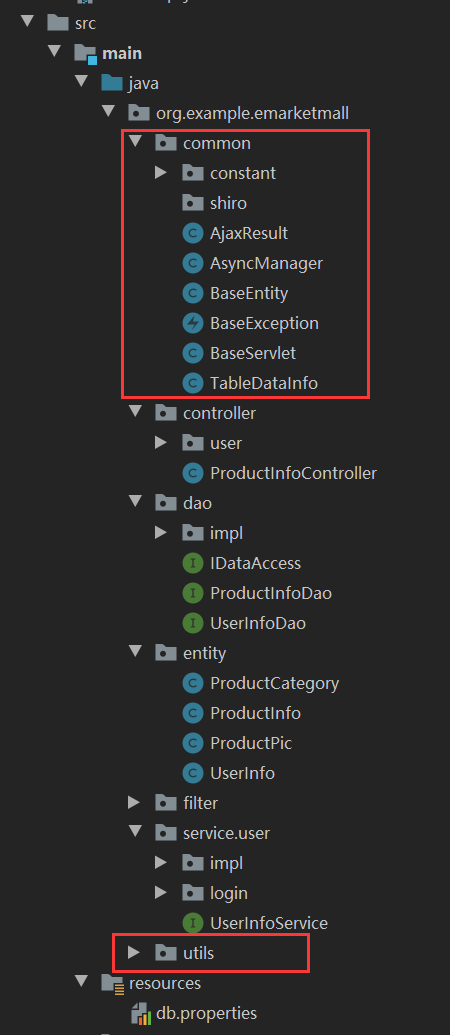
## 3.1、依赖引入

由于JavaWeb实训项目涉及到更多依赖和jar包，除了Java实训中用到的一些本地jar包外，build.gradle中dependencies闭包中也从mavan上提供更多所需的maven库依赖地址

**注意：**部分依赖只是为框架中的工具类提供支撑，并未之间使用。本项目前端使用jsp模板引擎和bootstrap，jQuery，后台使用传同的MVC模式和servlet搭配。



## 3.2、框架结构



common包：框架中的通用类

constant包：框架中的常量

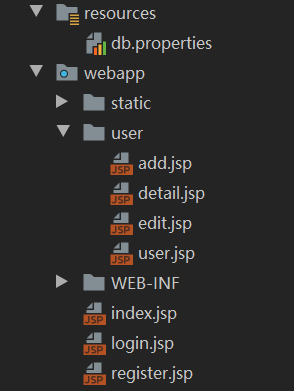
shiro：未使用

AjaxResult：从后端向前端返回数据的基本封装类，包含code，msg，data三个属性；ajax请求返回，servlet和jsp页数据传递均可以使用该类封装对象

TableDataInfo：从servlet向jsp页的bootstrap-table传递列表数据，并使用bootstrap-table方式显示时，使用该类封装列表数据，包含total，rows，code，msg四个属性。

controller，dao，entity，service和java实训中知识点一致。

由于部分功能，如user包含登录，注册，用户管理多个控制器，所以单独在对应的mvc包下创建子包。



resouces为项目的资源文件夹，保存druid数据连接池的属性文件

webapp为本框架项目的前端文件夹

static包含本项目中包含的所有css，js的本地文件

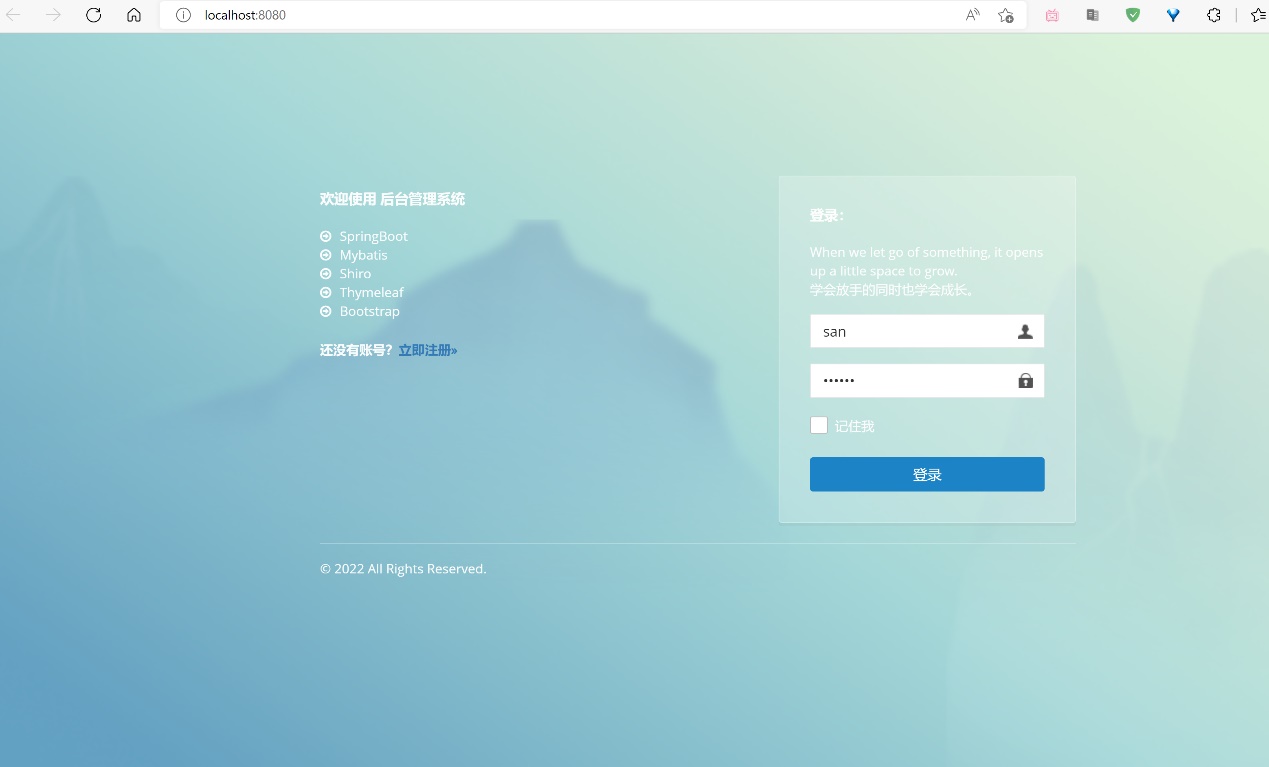
web-inf包含项目的配置文件，如web.xml

其余为本项目的功能页面文件。

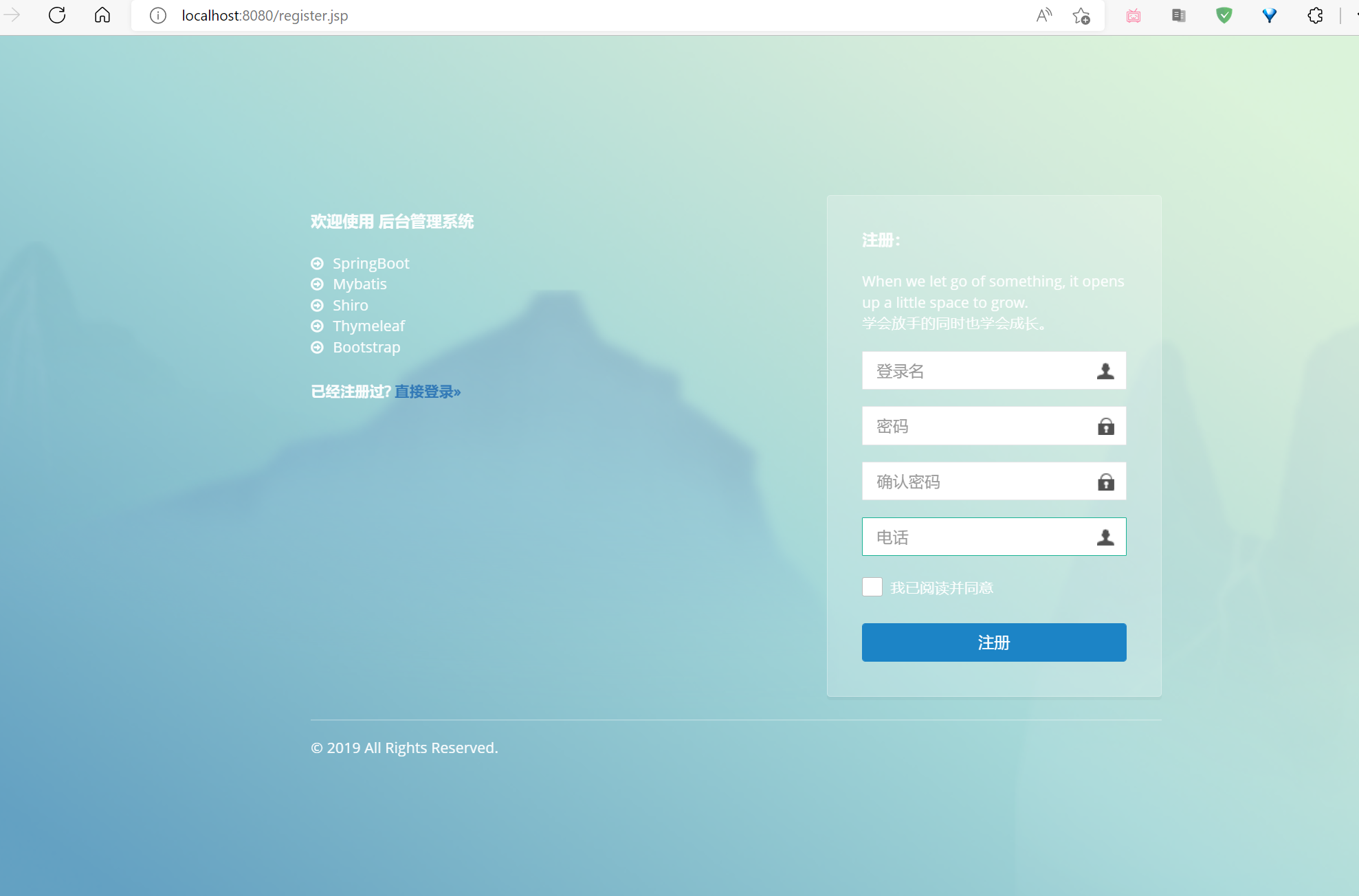
user包包含用户管理的前端页面，即user列表，新增，编辑，详情4个jsp页面。

## 3.3、用户管理运行结果

登录



注册



用户管理

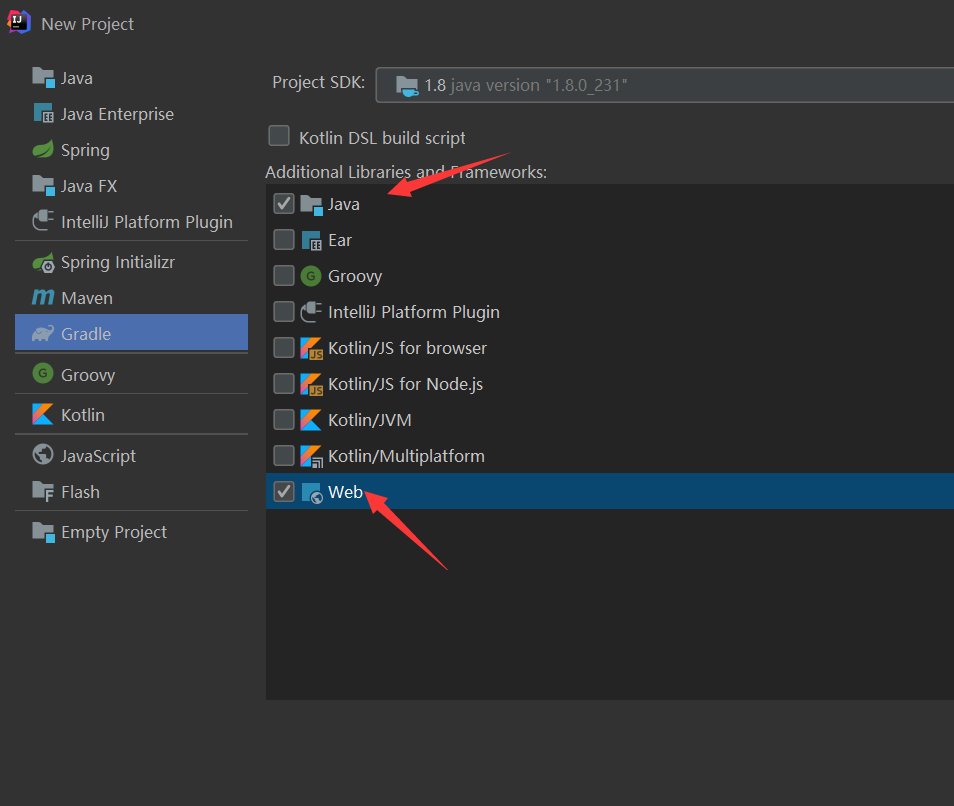


4、实验步骤

4.1、创建基于Gradle的JavaWeb项目

1）File->New->Project

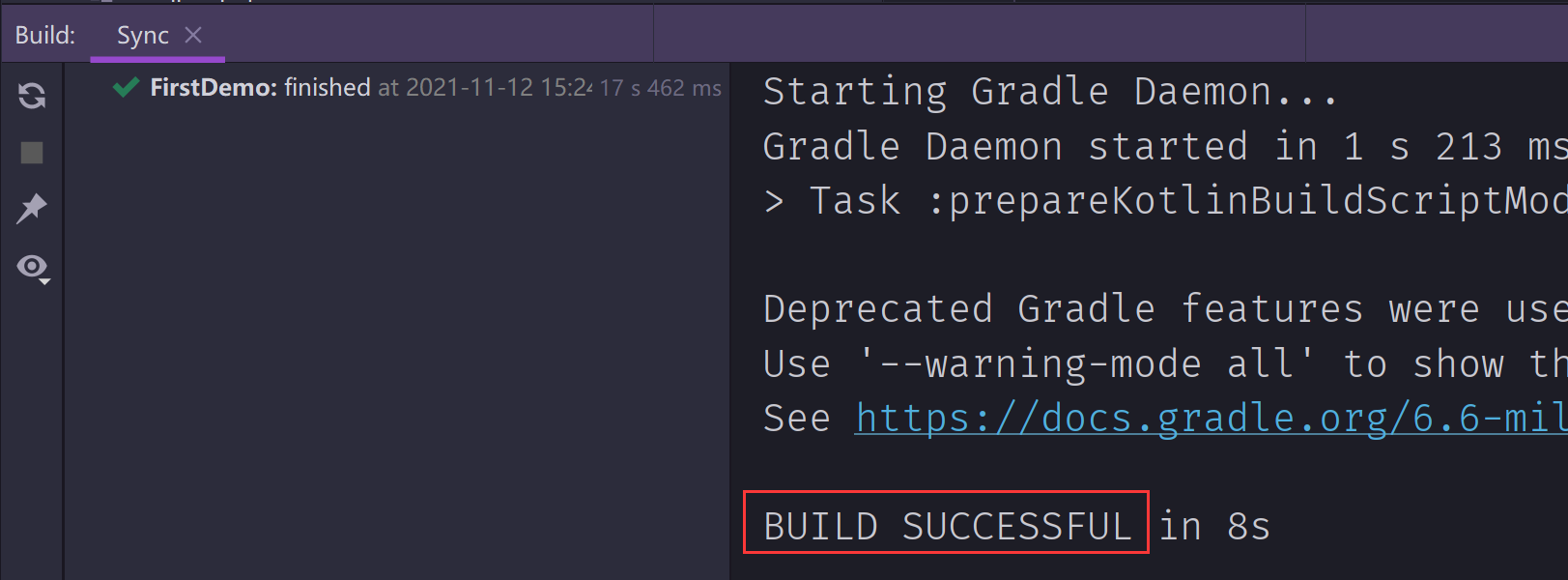
2）New Project对话框选择如下



3）保持网络通畅

如果未设置gradle本地路径，项目会根据gradle/wrapper/gradle-wrapper.properties文件中指定的distributionUrl地址下载gradle sdk

4）基于gradle的java项目创建成果，提示BUILD SUCCESSFUL



4.2、导入所需的依赖和jar包

本地jar包导入和java实训相同，libs文件夹中已经包含了框架所需要导入的本地jar包，说明如下：

1）common-dbuitls-1.7.jar apache数据访问底层包

2）cqcet-aibd-softutil-2.jar 集成druid配置类，并封装dbutils

3）druid-1.2.8 阿里巴巴数据连接池包

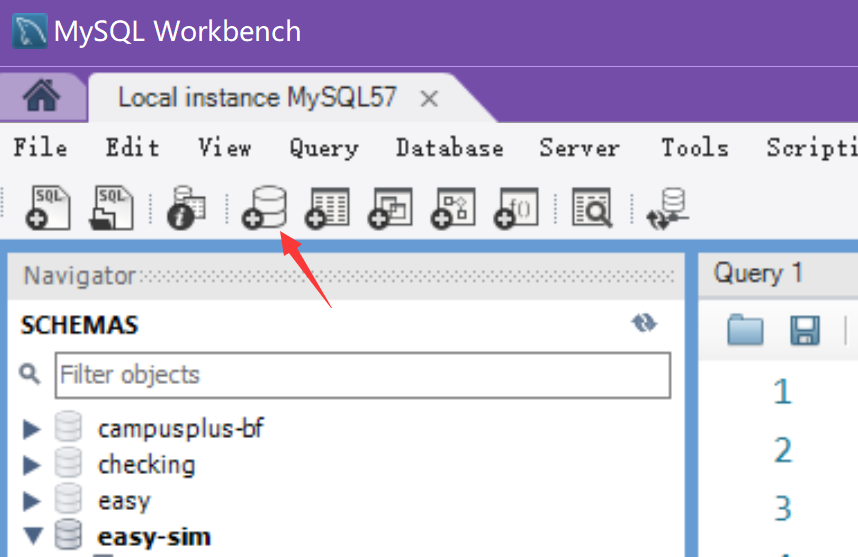
4）jsp-api.jar,servlet-api.jar 因为tomcat服务器中未引入这2个jar，需要重新引入

Build.gradle文件中dependencies闭包中引入的jar自行检索目的

4.3、在MySQL数据库管理工具中导入本项目所需的数据脚本

以MySQL workbench为例

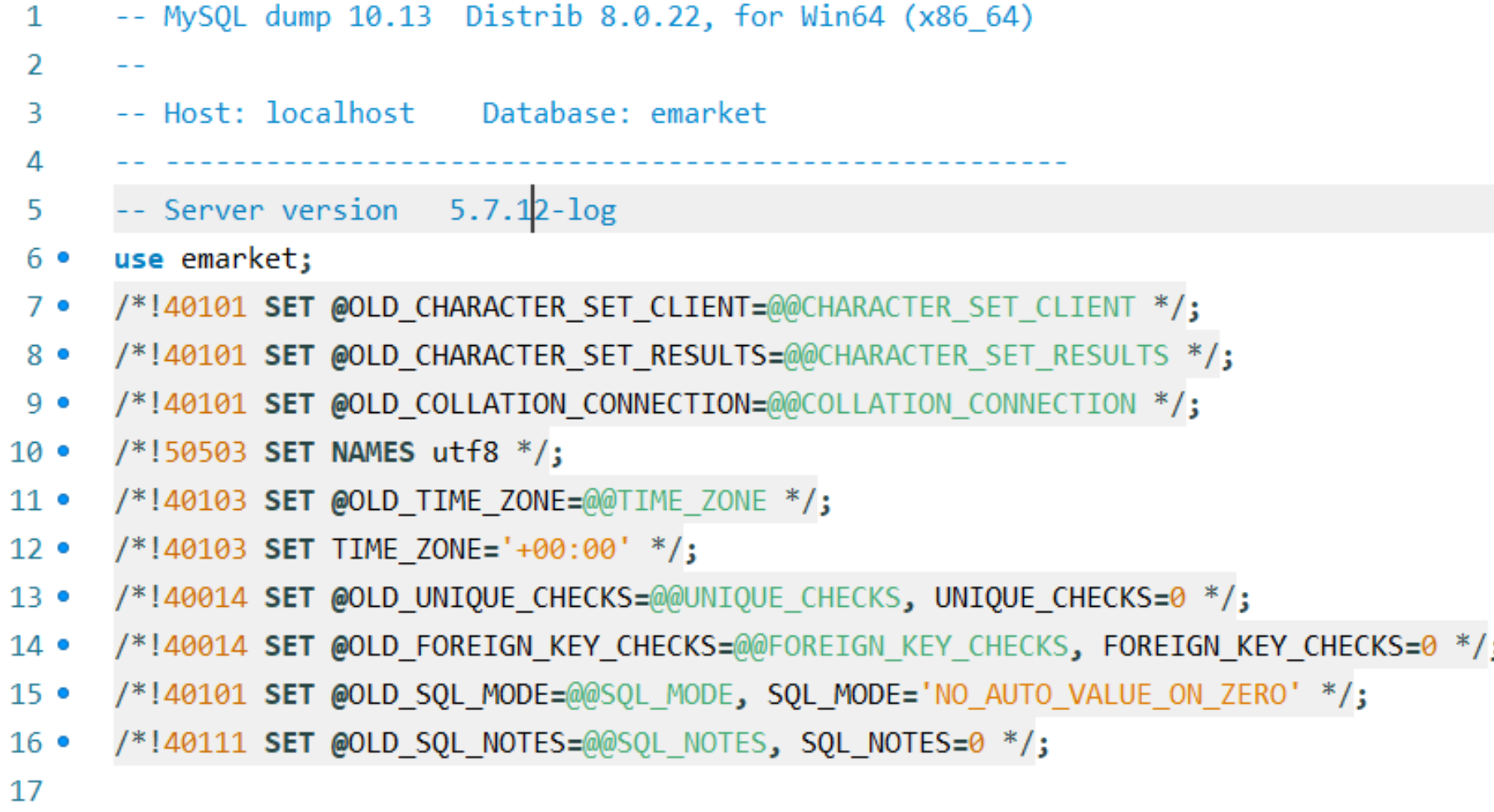
1）创建schema（数据库）



2）输入数据库名，如emarket\_mall

3）把某个数据脚本，如user\_info.sql拖入到MySQL workbench的工作区，显示数据脚本内容

4）添加use emarket;



5）运行脚本，其余脚本操作同上

4.4、创建项目属性文件，通过druid数据连接池连接

1）在src/main/resources目录

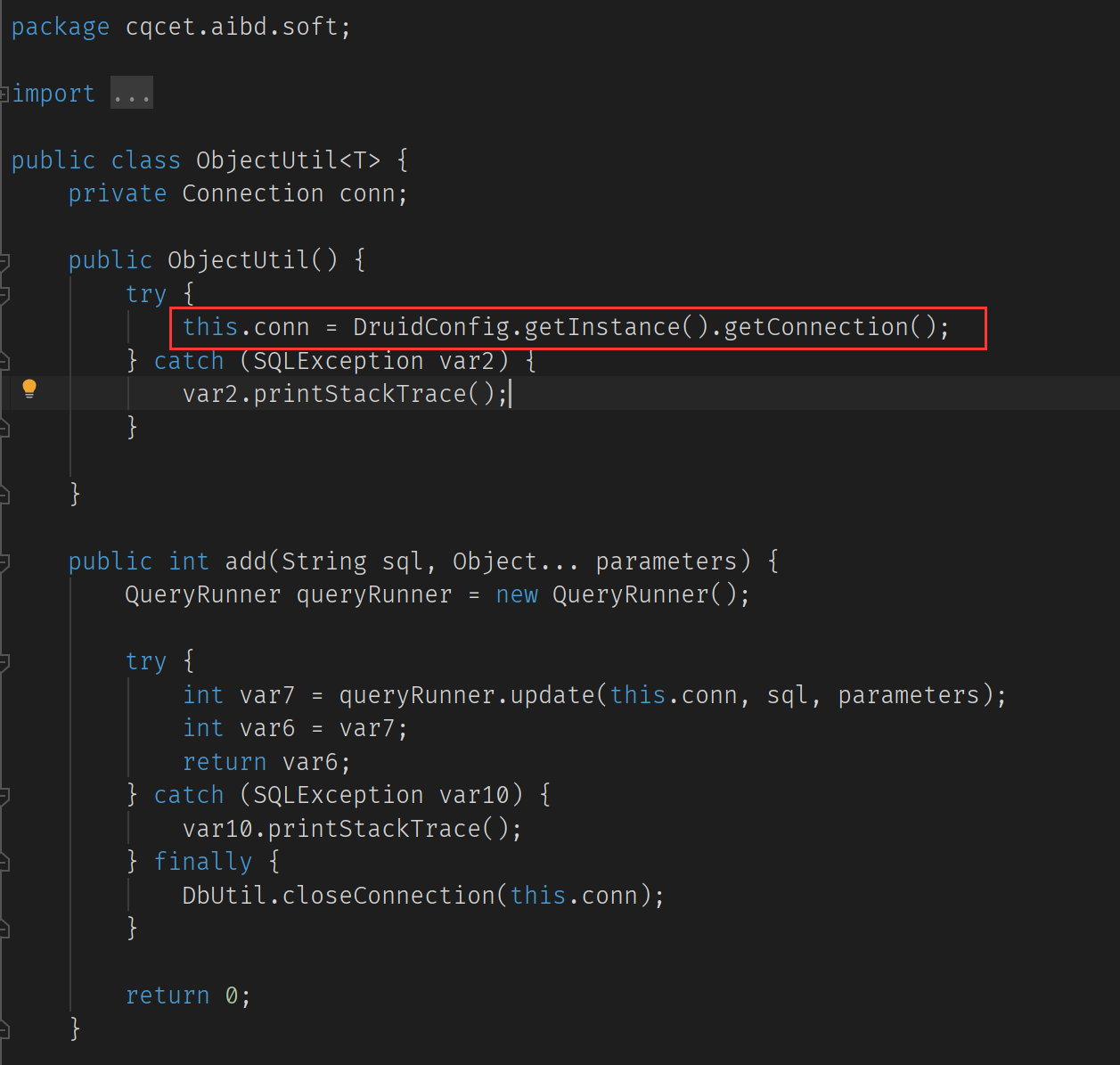
已经定义好了db.properties文件

2）db.properties文件内容包含了项目关联数据库的相关信息

3）打开本地导入的cqcet.aibd.sotfutil-2.jar包，选中ObjectUtil类，双击打开

**注意：**cqcet.aibd.sotfutil-2.jar必须要完成步骤4.2.1，本地jar包导入，并且执行gradle同步后才能展开，查看包含在jar包中类

ObjectUtil类反编译如下：



4）数据连接池的连接已经直接写入了ObjectUtil类的构造方法中，就是说只要调用ObjectUtil的构造方法，就会访问Druid数据连接池。

4.5、框架基本功能实训步骤

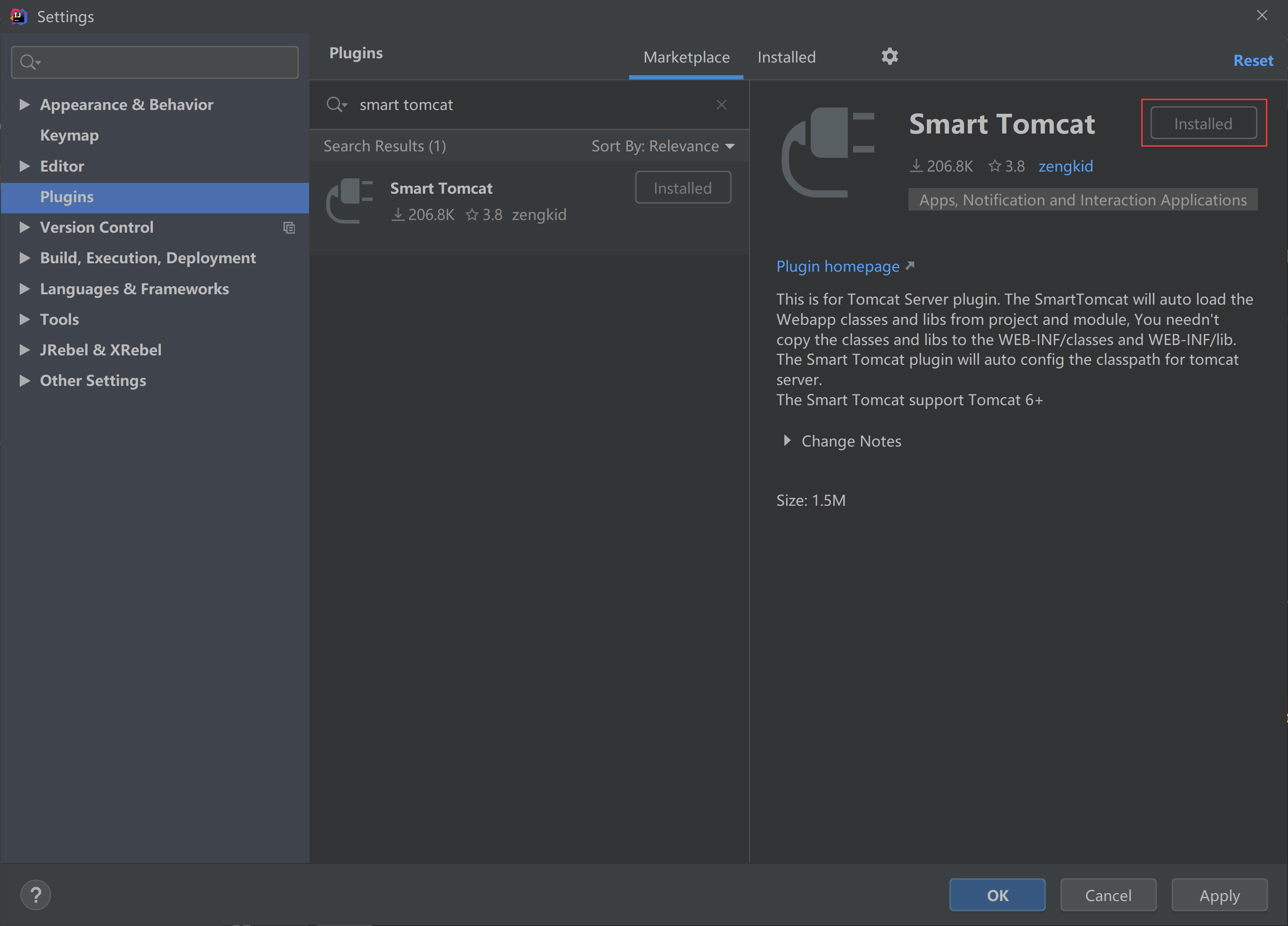
## 1）部署和运行

Idea部署和运行javaweb项目可以使用Tomcat服务器，不同版本的idea对tomcat的支持不同。

本指导书说明使用SmartTomcat插件配置tomcat服务器

IDEA版本不同，在部分新版本中会缺失Tomcat应用程序服务器的配置模板，需要在IDEA插件中添加Smart Tomcat插件。

选择菜单File，Settings（快捷键Ctrl-Alt-S），打开项目设置对话框，在左侧的导航栏中选择Plugins（插件），在右侧对应的Plugins栏中，选择Marketplace（插件市场），在搜索框中输入Smart Tomcat即可找到该插件（忽略大消息），如下图所示。



未安装的插件Install按钮为绿色，已安装过的插件Install按钮为灰色

安装完成之后，IDEA会提示重启应用，根据要求重新启动IDEA，再次进入Plugins，可以看到Smart Tomcat插件安装并处于执行状态，如图5所示。

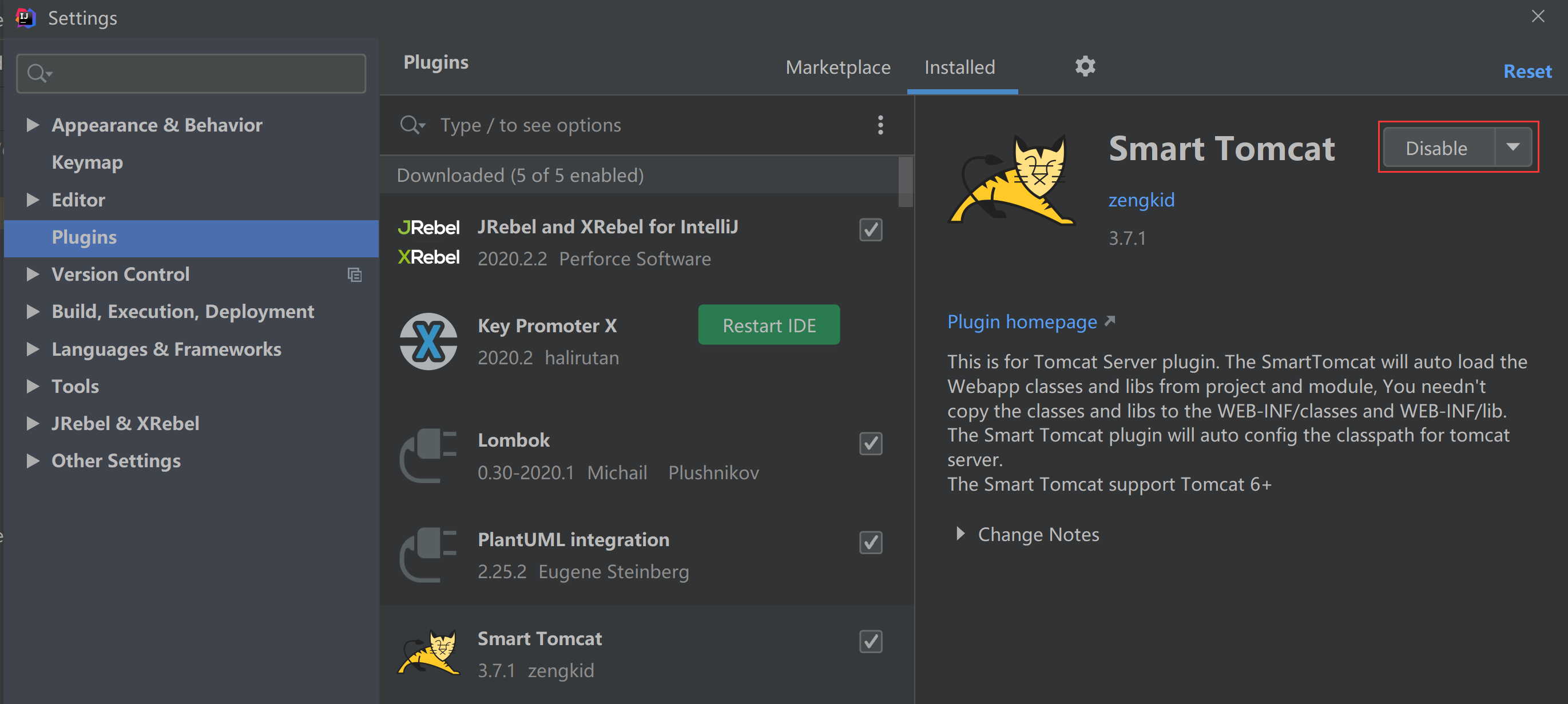


图5：Smart Tomcat插件安装成功

在IDEA菜单栏下方找到应用程序服务器配置按钮，如图7所示。

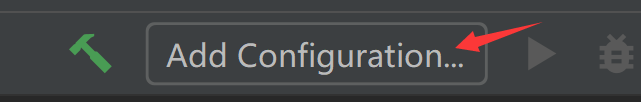


图7：服务器配置按钮

点击Add Configuration，展开Run/Debug Configuration对话框，左侧Templates（模板）导航栏上方选择“+”，找到Smart Tomcat或者Tomcat（部分2020前的IDEA版本自带Tomcat配置模板），如图8所示。

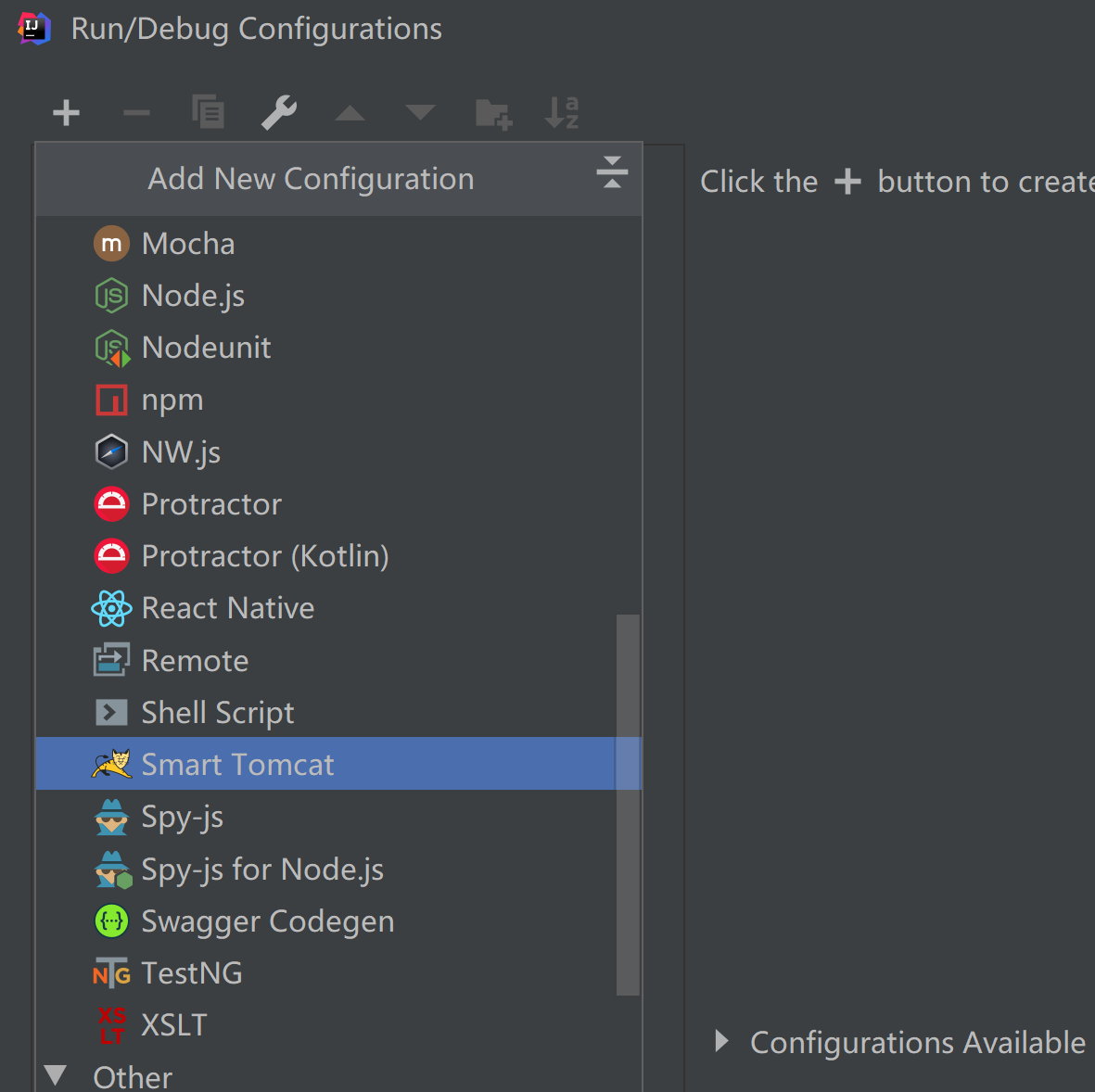


图8： 选择Smart Tomcat模板

选中Smart Tomcat（Tomcat），进入Tomcat服务器配置界面，如图9所示。

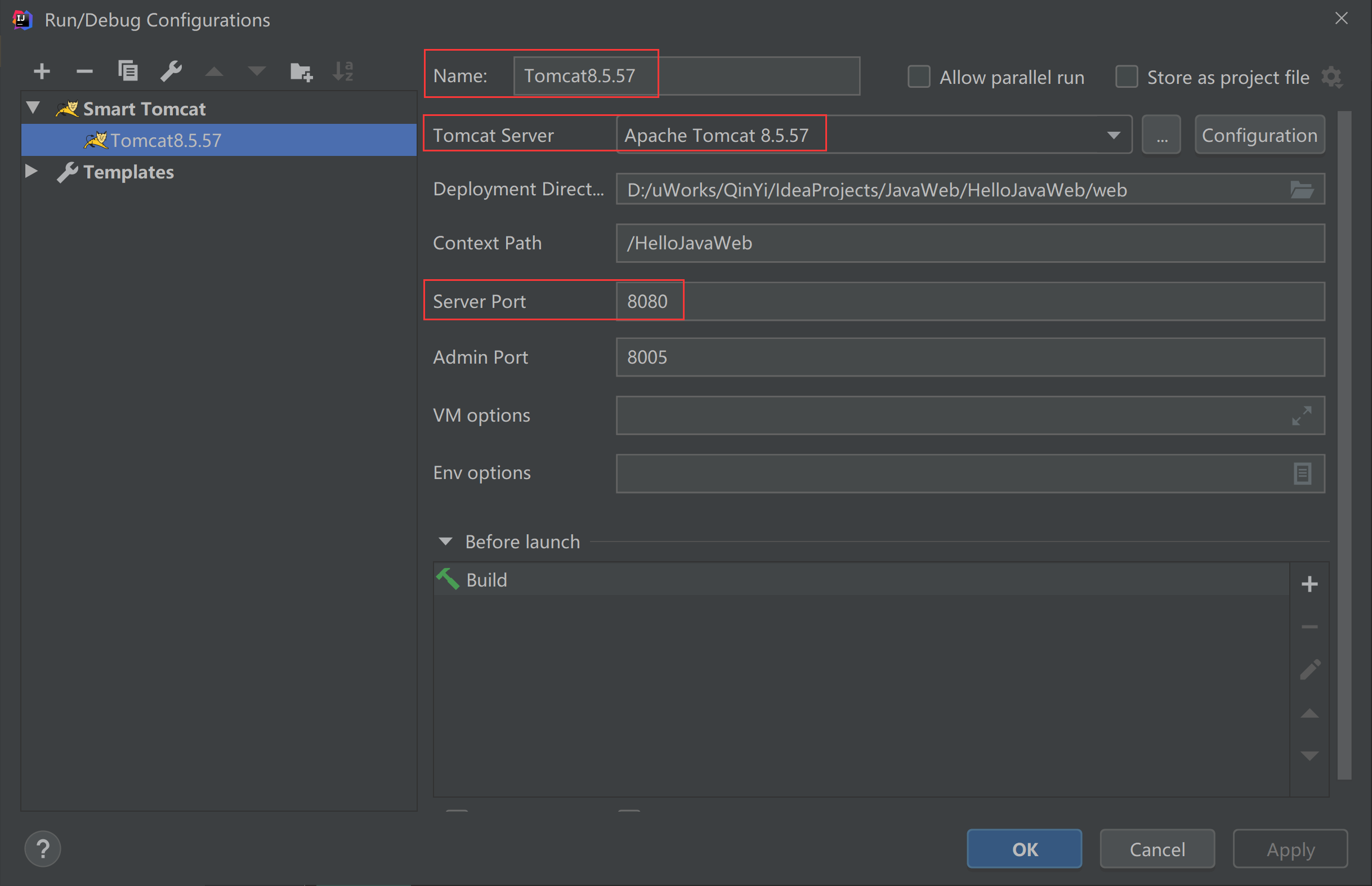
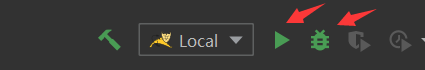


图9：Tomcat服务器配置

Name处根据Tomcat服务器版本自行填入；Tomcat Server需要在本机已经安装了Tomcat服务器的环境下，点击右侧的Configuration按钮，找到Tomcat应用服务器的安装目录指定即可；Server Port为Tomcat应用程序服务器的端口号，在运行Tomcat服务器后， 在浏览器地址栏需键入Tomcat服务器地址和端口号才能访问到该Tomcat服务器。

**注意：**Tomcat服务器请读者在各自的PC上自行完成安装配置。

服务安装成功后， 第一个箭头执行的三角形图标为运行该项目，第二个图片为调试。

## 2）用户登录

login.jsp为用户登录页面，通过点击提交按钮后，调用login()函数，提交ajax请求，实现对后台控制器的访问。



**知识点：**

1. ajax请求，响应
2. url指定的是控制器的映射地址，本项目中即为servlet-mapping的地址

ii）本项目中通过$.model.alertSuccess(),$.model.alertError(),$.model.alertWarning()弹出对话框，可以直接使用

根据url跳转到LoginController

注意：本项目通过servlet模拟MVC控制器，使用@WebServlet注解

由于servlet中包含get和post请求的处理方式doGet和doPost，本例中创建doProcessRequest()方法统一处理。

ServletUtils.getObjectFromPayload(req, UserInfo.class)

该方法获取request请求中的参数，并根据所提供类型，对参数进行封装成指定类型对象。

AjaxResult ajaxLogin(HttpServletRequest req, String loginName, String password, Boolean rememberMe)

该方法通过登录名和密码判断用户信息是否准确

登录成功或失败的消息通过AjaxResult类封装

ServletUtils.renderString()

该方法对需要返回到前端的字符串进行渲染，通过response对象发送到前端页面

在LoginController中调用LoginService的方法，LoginService又调用UserInfoDao中的方法，印证Java实训中所学习的MVC的调用规则。

注册功能同登录类似。

## 3）用户管理

通过在index.jsp页面中的<a>标签跳转到UserController

<a href="${ctx}/mall/user?opt=list">用户列表</a>

**知识点：**

本项目由于使用Servlet模拟控制器，只能通过@WebServlet对Servlet类进行映射

Servlet中的方法，如用户管理的列表，新增，编辑，删除，通过在url后追加opt

参数实现.

opt=list

opt=add

opt=edit&id={id}

opt=remove&id={id}

3.1)用户列表

@WebServlet("/mall/user")  
public class UserInfoController extends HttpServlet

a）请求源来至其它页面或servlet

String referer = req.getHeader("Referer");  
if (referer.contains("index")) {  
 resp.sendRedirect("/user/user.jsp");  
 return;  
}

通过Referer值判定，如本例中，请求来至index页面，且除了opt外没有其它参数 ，则直接转发到user.jsp页面；通过user.jsp页面重新提交请求。

这里的判断可以修改，但是要注意请求页面是否带参，以及参数的内容。

b）根据上面代码，跳转至user.jsp页面，执行$(function () {…}

整个function包含的是一个封装了的bootstrap-table，说明如下：

id: "bootstrap-table",  
url: route + "?opt=list",  
createUrl: route + "?opt=add",  
removeUrl: route + "?opt=remove&id={id}",  
updateUrl: route + "?opt=edit&id={id}",  
detailUrl: route + "?opt=detail&id={id}",

id和<table id="bootstrap-table"></table>关联，表示该function执行的数据和该table标签关联。

url：默认请求地址，就是列表的请求地址

下面的几个Url请各位同学自己思考



columns表示在table标签中显示的列

filed和后台返回的json数据中的key进行对比，相同则显示对应的值，所以field为id，则保存id的值

title则是列名，即在table中显示的列标题

$.table.init(options);

上面的提到的url，columns等都是options变量的属性成员，通过$.table.init()方法初始化options变量，根据不同的url向后台控制器发出请求，并通过返回的json对象完成对table的渲染。

c）返回UserInfoController

由于url的值为/mall/user?opt=list，所以请求再次跳转到用户控制器

ServletUtils.getParamFromPayLoad(req, "opt");

通过getParamFromPayLoad中获取指定参数的值，这里取得opt的值为list

通过条件判断跳转到对应列表执行语句

ServletUtils.renderString(resp, JSON.toJSONString(list(userInfo) != null ? list(userInfo) : AjaxResult.error("用户信息查询异常")));

通过list()方法调用UserService中的selectUserInfoList(user)，返回用户列表，list()的返回对象为TableDataInfo，该返回对象就是和bootstrap-table关联

再通过renderString对json串进行渲染，并响应给请求页面user.jsp，通过table进行显示

c1）list(UserInfo user)带有参数，如果参数为空，则返回所有用户列表；如果参数不为空，则根据参数中的属性进行查询

在user.jsp页提供的查询参数有name，loginName，email和phone，通过$.table.search()获取对应参数并传递到后台，通过userInfoService.selectUserInfoList(user)进行查询

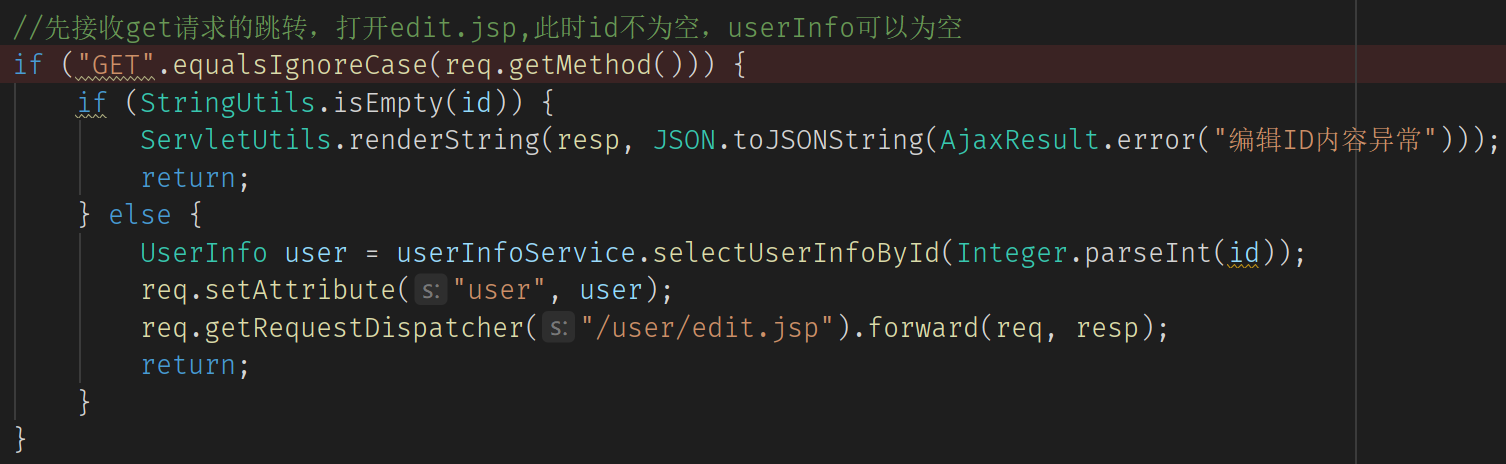
3.2）用户编辑

在user.jsp页面中选择某一个用户姓名前的复选框，点击“修改“

或直接点击操作列下方的“编辑“按钮，均可以对所选则的用户信息进行修改

步骤如下：

1. 第一次请求为get请求，携带opt=edit&id={id}到后台控制器  
   先取得opt和id参数的值，在根据条件跳转到edit条件处执行



通过id查询到用户的相关信息，并保存到request请求对象中，再通过请求转发到edit.jsp页面，即用户编辑页面

1. 在edit.jsp页面中，从request对象中取出用户对象，保存到当前上下文对象中

<% pageContext.setAttribute("user", request.getAttribute("user")); %>

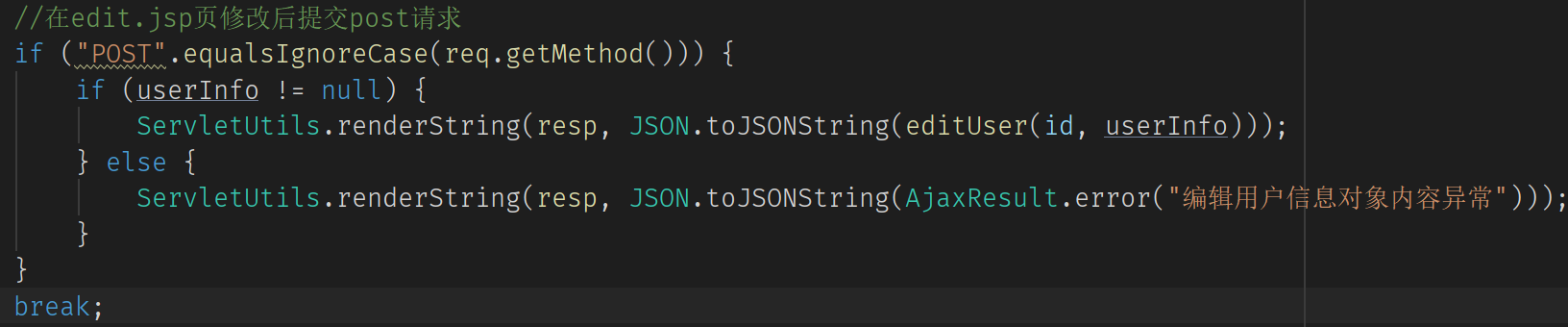
则可以在该上下文范围内完成edit.jsp页面该用户信息的显示

<input name="name" class="form-control" value="${user.name}" type="text" required>

1. 通过submitHandler方法提交表单数据

function submitHandler() {  
 if ($.validate.form()) {  
 $.operate.save(prefix + "?opt=edit", $('#form-user-edit').serialize());  
 }  
}

1. 通过$.operate.save()方法中的路径参数/mall/user?opt=edit，再次携带用户信息跳转到UserController
2. 第二次请求为post



用户信息不为空，则调用editUser()方法更新user\_info数据表中的记录，否则返回异常。