1. 系统概述与环境搭建1 系统开发及运行环境电脑商城系统开发所需的环境及相关软件进行介绍。  
   1.操作系统：Windows 10  
   2.Java开发包：JDK 8 （1.8版本及以上）  
   3.项目管理工具：Maven 3.6.1及以上  
   4.项目开发工具：IntelliJ IDEA 2020.3.2 x64  
   5.数据库：mysql5.1及以上版本、MariaDB、navicat  
   6.浏览器：Google Chrome  
   7.服务器架构：Spring Boot 2.4.7 + MyBatis 2.1.4 + AJAX

**2 项目分析**1.在开发某个项目之前，应先分析这个项目中可能涉及哪些种类的数据。本项目中涉及的数据：用户、商品、商品类别、收藏、订单、购物车、收货地址。  
2.关于数据，还应该要确定这些数据的开发顺序。设计开发顺序的原则是：先开发基础、简单或熟悉的数据。

以上需要处理的数据的开发流程是：用**户-收货地址-商品类别-商品-收藏-购物车-订单**。  
3.在开发每种数据的相关功能时，先分析该数据涉及哪些功能。在本项目中以用户数据为例，需要**开发的功能**有：登录、注册、修改密码、修改资料、上传头像。  
4.然后，在确定这些功能的开发顺序。原则上，应先做基础功能，并遵循增查删改的顺序来开发.则用户相关功能的开发顺序应该是：注册-登录-修改密码-修改个人资料-上传头像。  
5.在实际开发中，应先创建该项目的数据库，当每次处理一种新的数据时，应先创建该数据在数据库中的数据表，然后在项目中创建该数据表对应的实体类。  
6.在开发某个具体的功能时，应遵循开发顺序：持久层-业务层-控制器-前端页面。

持久层开发：依据前端页面的设置规划相关的SQL语句，以及配置

业务层开发：核心功能控制，业务操作以及异常处理

控制层开发：接受请求、处理响应。

前端开发：JS、Query、AJAX来连接后台

**3 搭建项目**

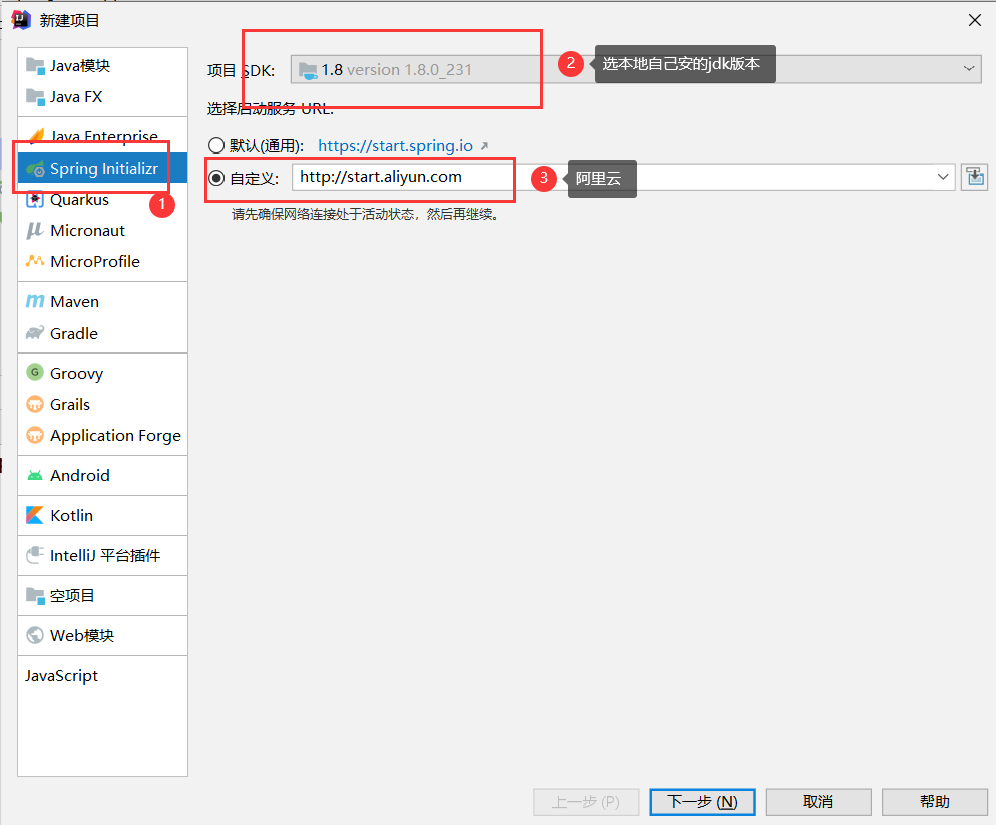
1.项目名称：store，表示商城

2.结构：com.fx.store

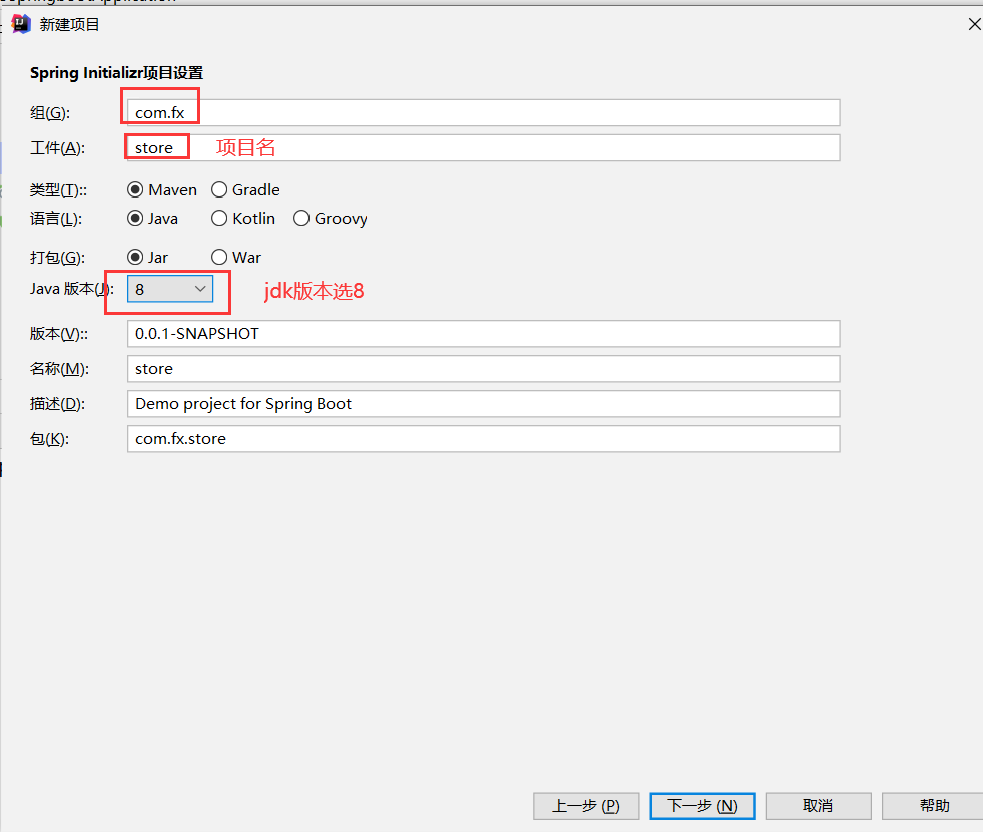
3.资源文件：resources文件夹下（static(静态资源)、templates（模板））

4.单元测试：test.com.fx.store

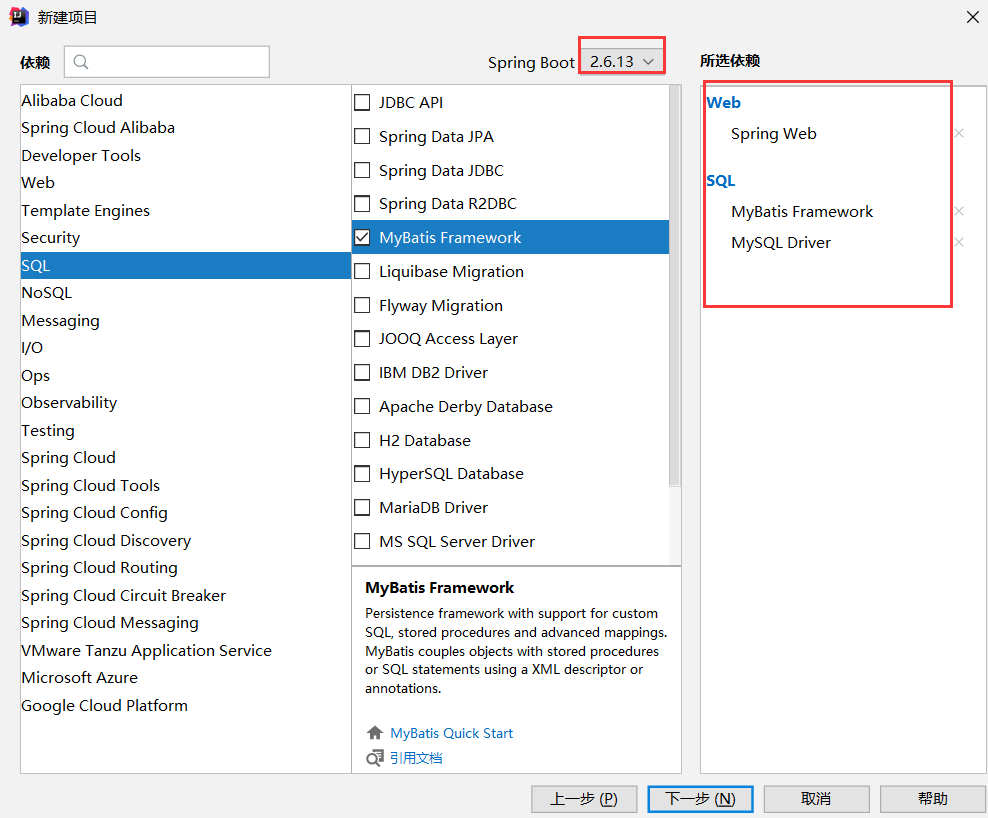
**4 创建Spring Initializr项目**本质上Spring Initializr是一个Web应用程序，它提供了一个基本的项目结构，能够帮助开发者快速构建一个基础的Spring Boot项目。在创建Spring Initializr类型的项目时需在计算机连网的状态下进行创建。  
1.首先确保计算机上安装了JDK、IDEA、mysql等开发需要使用的软件，并在IDEA中配置了Maven3.6.3项目管理工具。  
2.在IDEA欢迎界面，点击【New Project】按钮创建项目，左侧选择【Spring Initializr】选项进行SpringBoot项目快速构建。



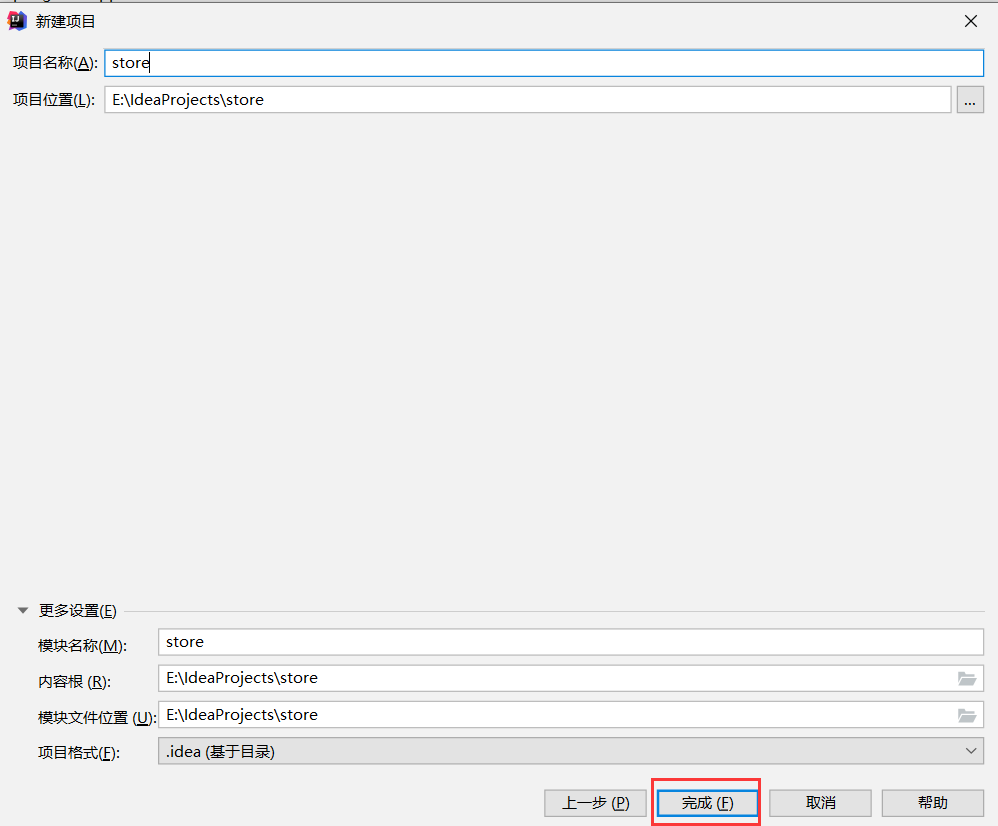
3.将Group设置为com.fx，Artifact设置为store，并做以下标红的修改。单击【Next】进入Spring Boot场景依赖选择界面。



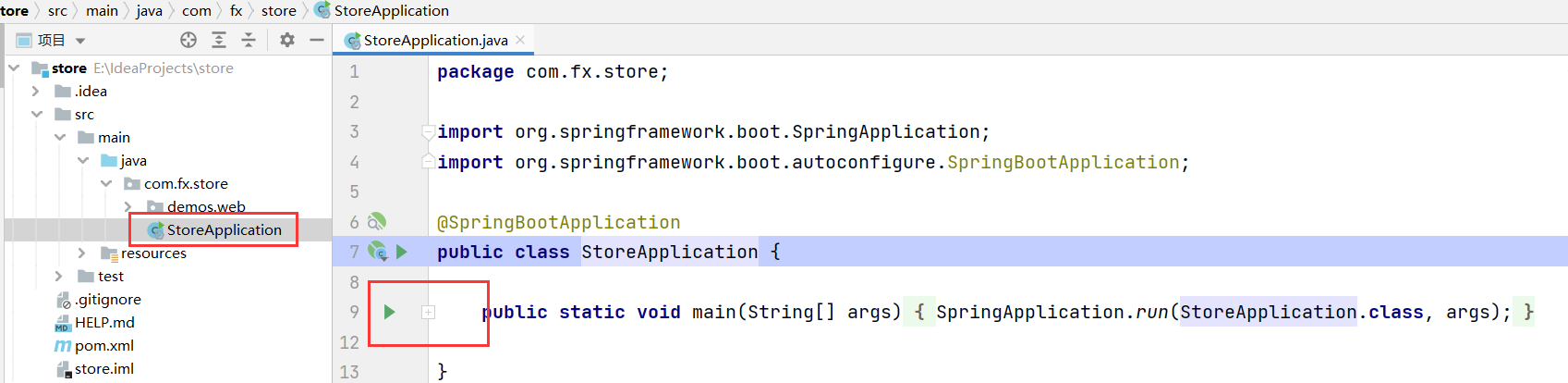
4.给项目添加Spring Web、MyBatis Framework、MySQL Driver的依赖。点击【Next】按钮完成项目创建。



5.首次创建完Spring Initializr项目时，解析项目依赖需消耗一定时间（Resolving dependencies of store...）。

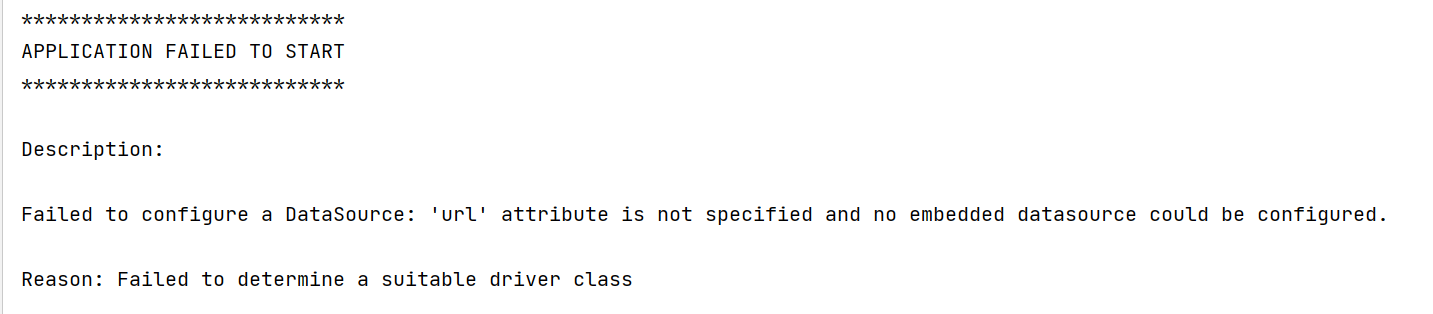


**5 配置并运行项目  
5.1 运行项目**找到项目的入口类（被@SpringBootApplication注解修饰），然后运行启动类；启动过程如果控制台输出Spring图形则表示启动成功。





**5.2 配置项目**1.如果启动项目时提示：“配置数据源失败:'url'属性未指定，无法配置内嵌的数据源”。有如下的错误提示。

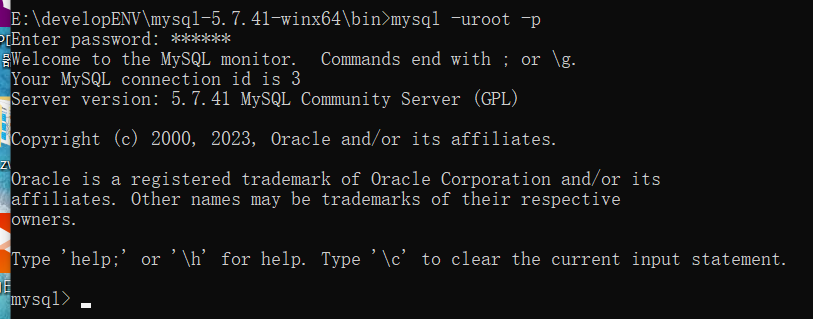


2.解决以上操作提示的方法：在resources文件夹下的application.properties文件中添加数据源的配置（用户名、密码和url）。

|  |
| --- |
| spring.datasource.url=jdbc:mysql://localhost:3306/store? useUnicode=true&characterEncoding=utf-8&serverTimezone=Asia/Shanghai spring.datasource.username=root spring.datasource.password=123456 |

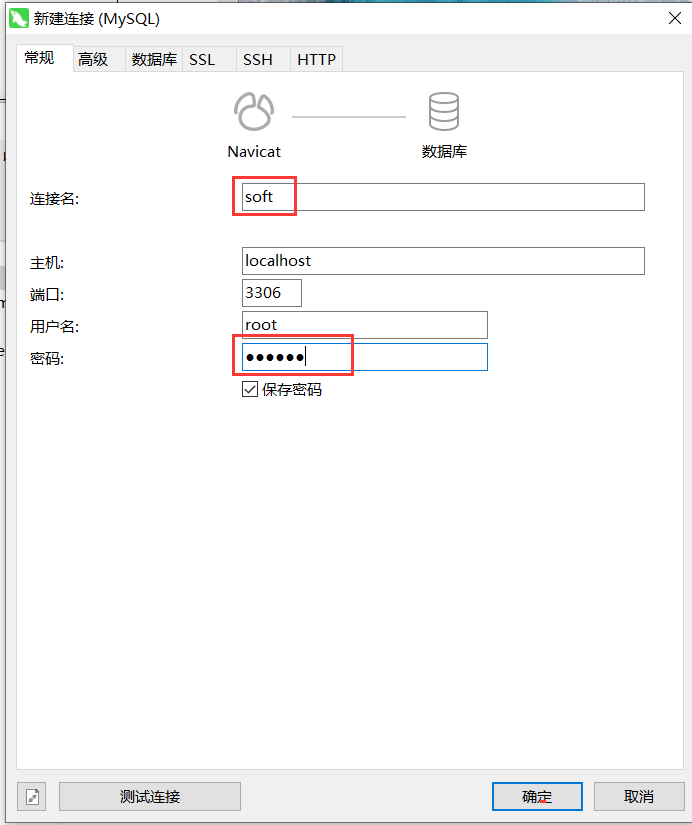
3.创建数据库store

打开MySQL，并打开navicat连接mysql

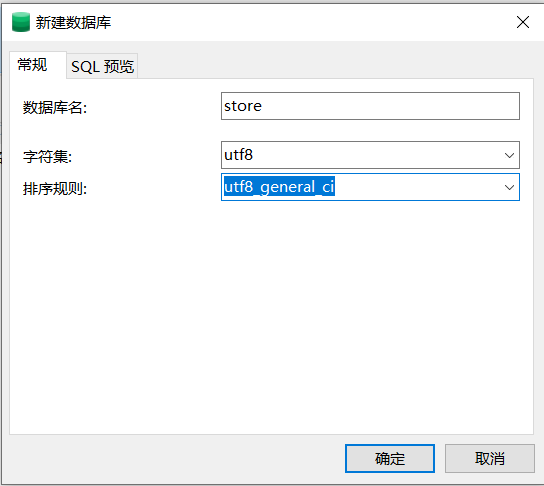


输入命令：create database store character set utf8;

或者使用navicat新建连接soft



创建数据库

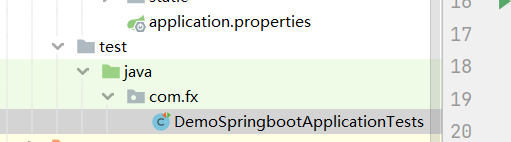


4.测试连接

（1）启动springboot的主类，是否有对应的spring图形输出

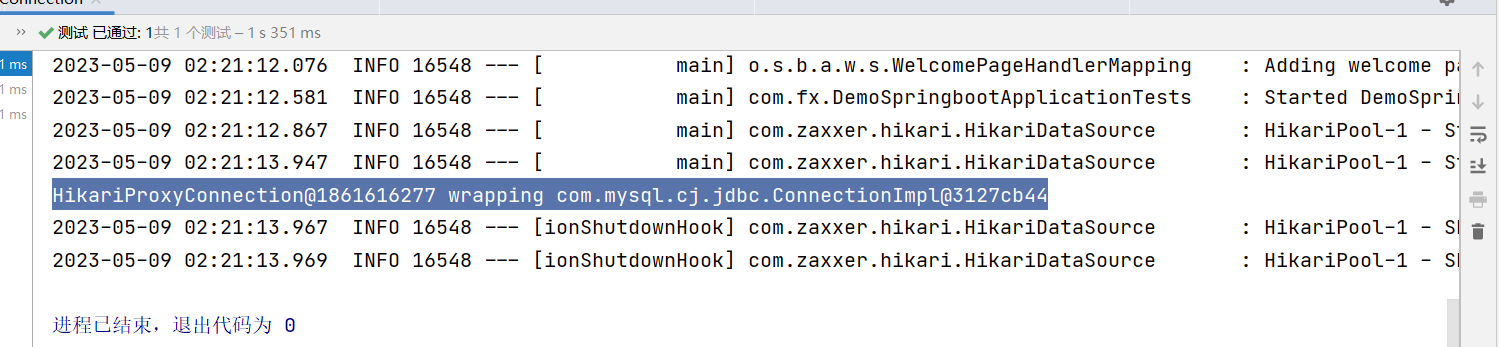
（2）在单元测试类中数据库的连接是否可以正常的加载

在测试文件夹Test下的DemoSpringbootApplicationTests类下添加以下代码。



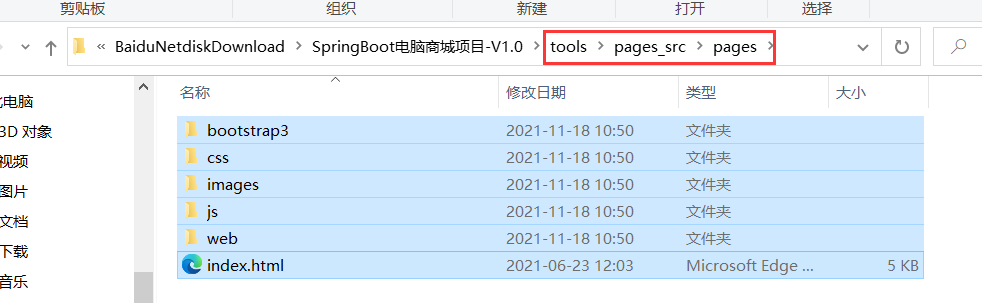
|  |
| --- |
| **package** com.fx;  **import** org.junit.jupiter.api.Test; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.boot.test.context.SpringBootTest;  **import** javax.sql.DataSource; **import** java.sql.SQLException;  @SpringBootTest **class** DemoSpringbootApplicationTests {  @Autowired**//自动装配  private** DataSource **dataSource**;   @Test  **void** contextLoads() {  }   **/\*\*  \* 数据库连接池  \* 1.DBCP（jdbc下）  \* 2.C3P0（springMVC基本淘汰）  \* 3.Hikari（springboot默认数据库连接池，号称世界上最快的连接池，由日本开发者开发，底层依旧采用C3P0管理数据库连接对象）  \* HikariProxyConnection@1861616277 wrapping com.mysql.cj.jdbc.ConnectionImpl@3127cb44  \* @throws SQLException  \*/** @Test  **void** getConnection() **throws** SQLException {  System.***out***.println(**dataSource**.getConnection());  }  } |

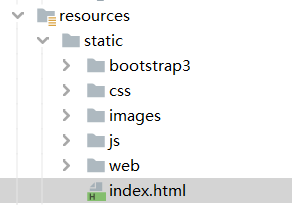
运行getConnection()方法显示以下信息说明数据库连接成功。



5.访问项目的静态资源，看是否能够正常加载，将所有的静态资源复制到这个项目结构的static目录下。

打开资源包下的tools，找到pages，将该目录下的所有资源都放到项目的static下





注意：idea对于JS代码的兼容性较差，编写js代码但是有时候不能正常去加载，可以通过以下三种方式解决：

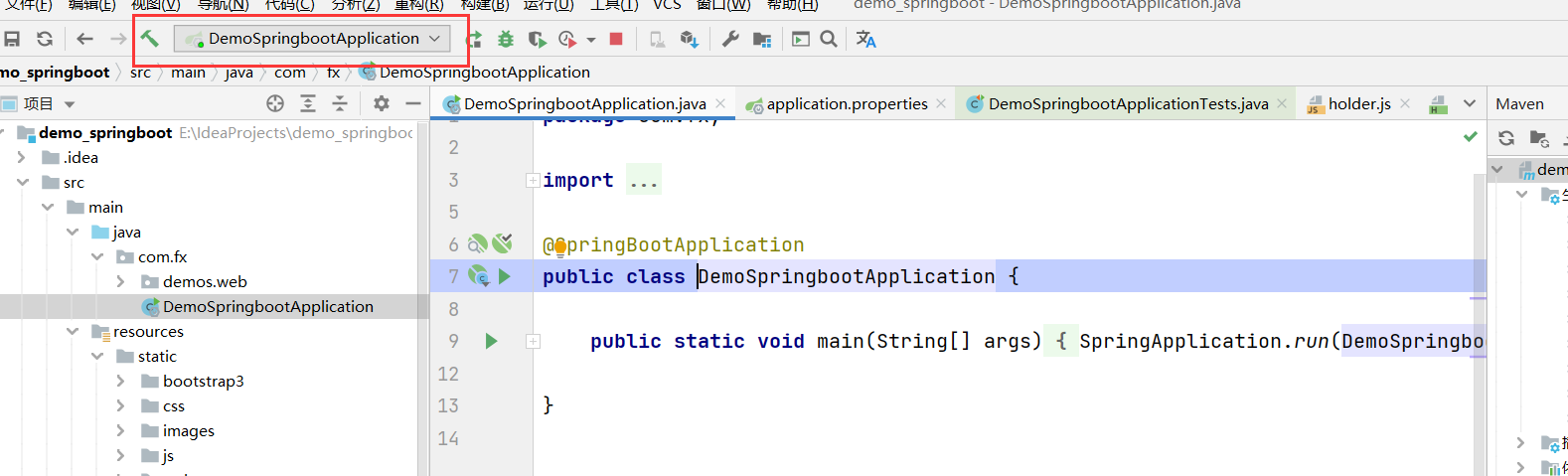
（1）idea缓存清理

（2）clear-install（maven生命周期）

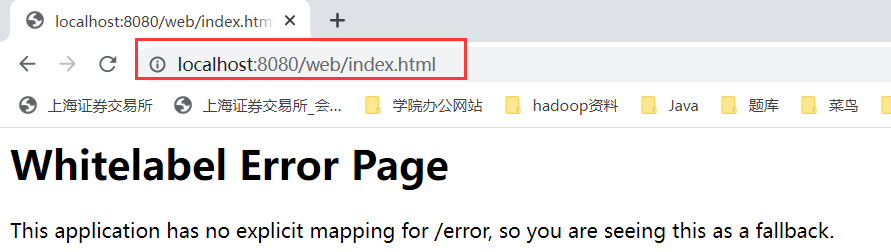
（3）rebuid重新构建

（4）重启idea和操作系统

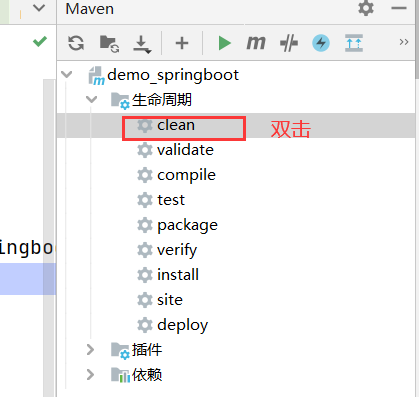
重启服务：



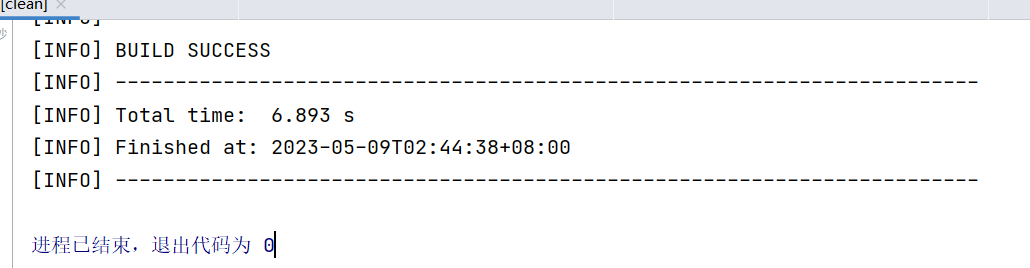
打开浏览器无法显示页面，出现以下：

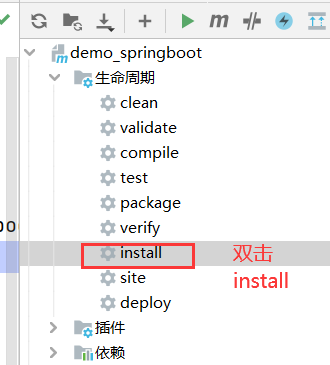


第一次加载静态资源idea可能出现找不到的情况，使用第二种解决方式：

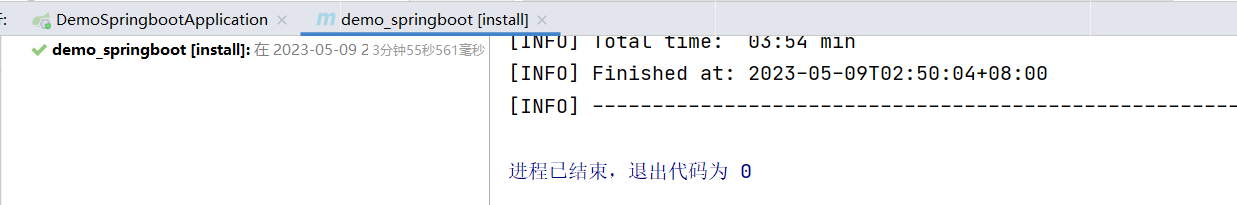


出现以下信息：





出现以下信息：



则重新浏览页面，出现以下信息说明静态资源加载成功：

