

1. 环境搭建

安装本节课的必备环境

1.1 JDK 的安装

业界使用最广泛的 JDK 8

1.2 Maven 的安装

Maven 是一个项目管理工具，可以对 Java 项目进行构建、依赖管理

1.3 IntelliJ 的安装

Java 工程师必备的 IDE

1.4 MySQL 的安装

最新版本 MySQL 8

1.5 Navicat 的安装

Navicat 可视化数据库管理软件

第二章 项目环境搭建与活动发布功能

欧阳修

本章导学

完成项目环境的搭建，实现秒杀活动的发布功能

1. 环境搭建



☐ JDK 的安装 (文档 + 视频)

☐ Maven 的安装 (文档 + 视频)

☐ IntelliJ 的安装 (文档 + 视频)

☐ MySQL 的安装 (文档 + 视频)

☐ Navicat 的安装 (文档 + 视频)

2. 创建项目



☐ Spring Boot 介绍

☐ 新建项目

☐ Hello, world!

☐ pom.xml 依赖配置

☐ 启动类配置

3. 秒杀活动发布功能



☐ 功能介绍

☐ 表单字段

☐ 设计秒杀活动表

☐ Spring Boot 整合 MyBatis

☐ MyBatis 逆向生成

☐ properties 配置数据库连接

☐ Spring Boot 跳转发布页

☐ 处理发布页面的表单请求

4. 秒杀活动列表页面



☐ 功能介绍

☐ Spring Boot 跳转活动列表页

☐ 功能测试与演示

使用到的技术

- 1 MySQL 数据库
- 2 Spring Boot
- 3 MyBatis 流行的 ORM 框架
- 4 MyBatis 逆向生成配置

成果预览

本章成果预览



搜索

新增秒杀活动信息

秒杀活动名称:

商品ID:


秒杀价格:

商品原价:

秒杀商品数量:


开始时间:

年 / 月 / 日 ----:--



结束时间:

年 / 月 / 日 ----:--



提交



正品保障
正品保障，提供发票



正品保障
正品保障，提供发票



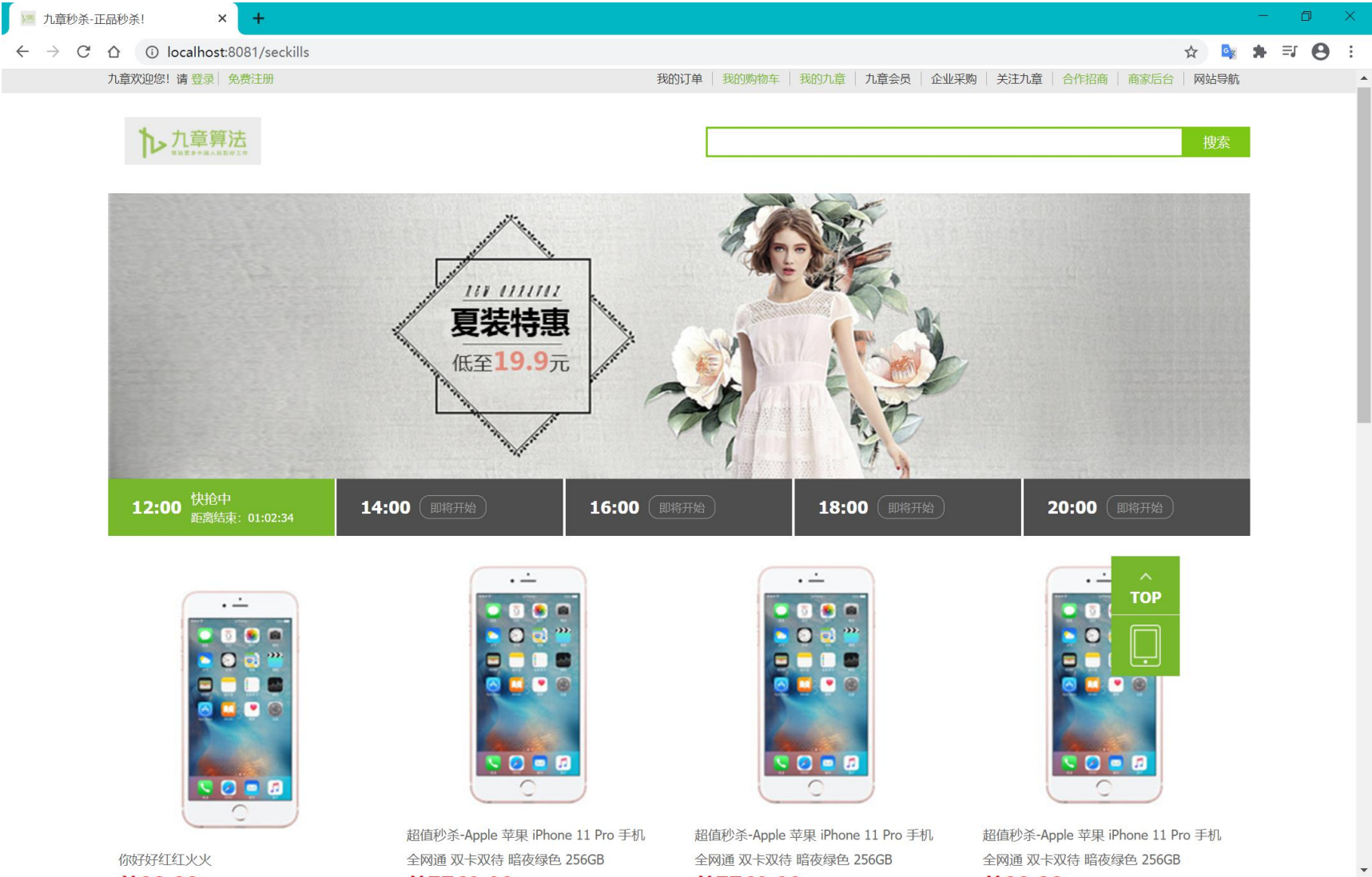
正品保障
正品保障，提供发票



正品保障
正品保障，提供发票

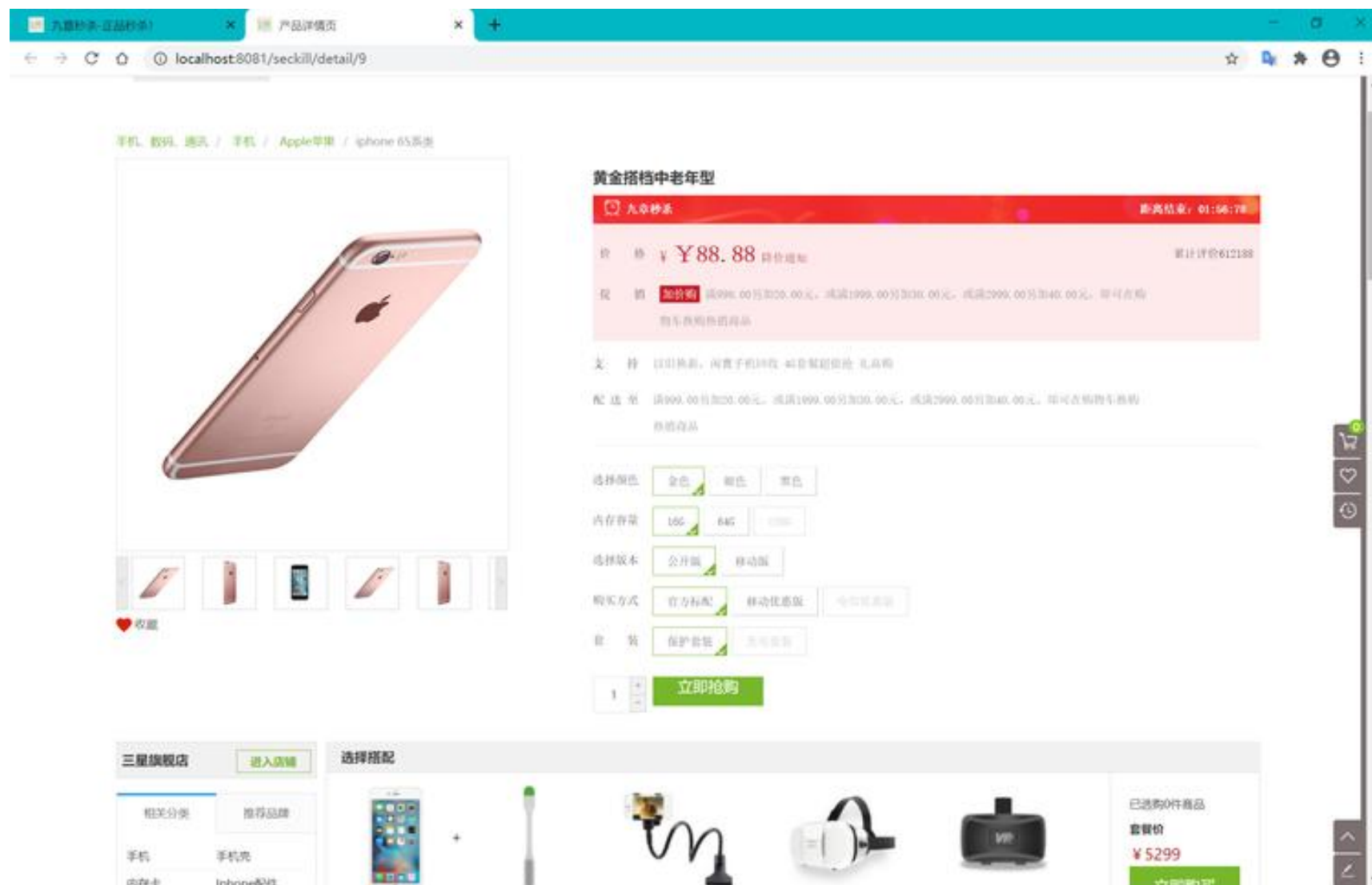


正品保障
正品保障，提供发票



秒杀活动列表页

商品详情页



2. 创建 Spring Boot 项目

Hello, Spring Boot!

Spring Boot 是什么?

你真的了解 Spring Boot 吗?



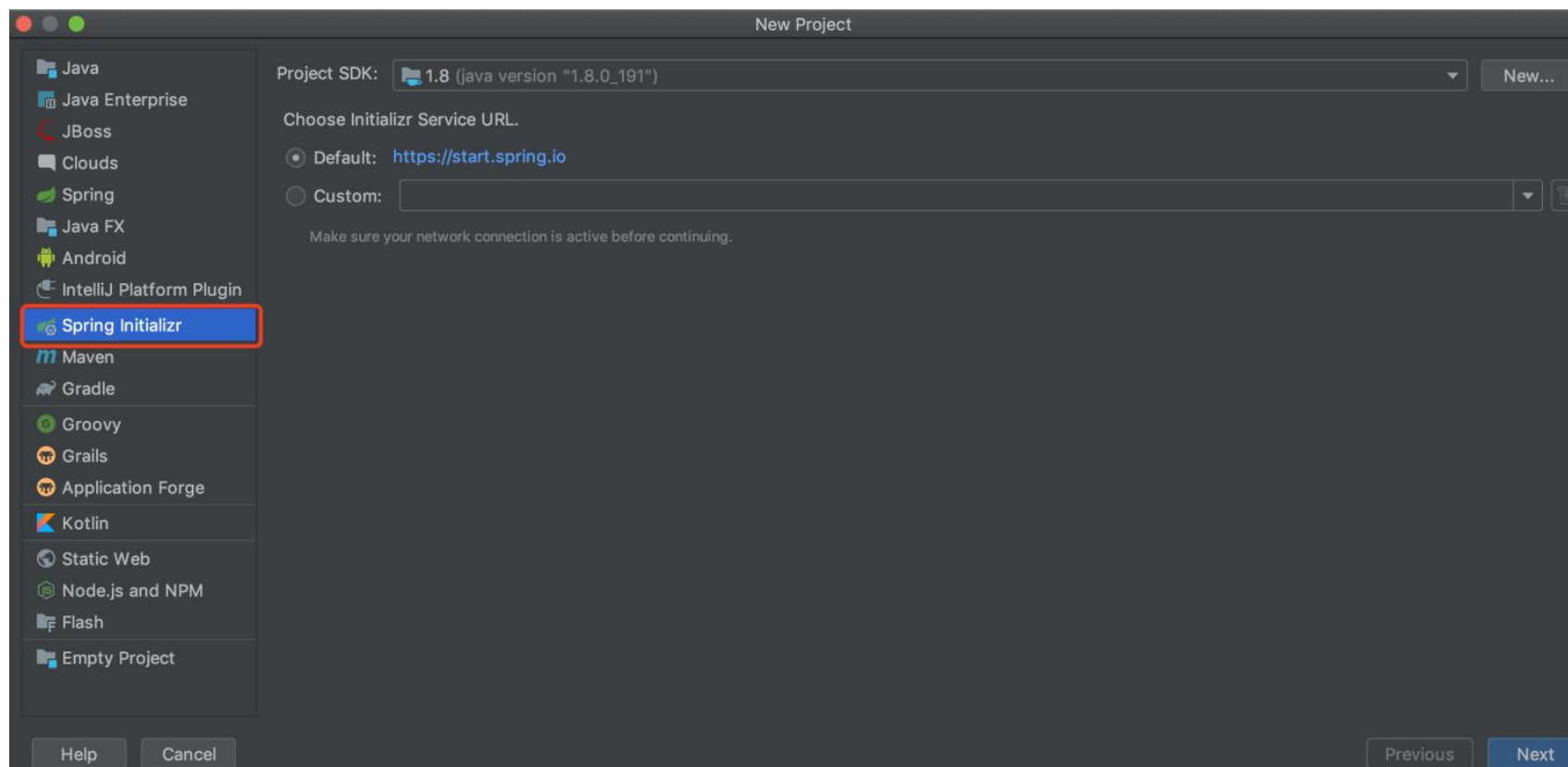
1. Spring Boot 是开发框架

2. 基于 Spring Framework

3. 目的在于简化 Spring 开发流程

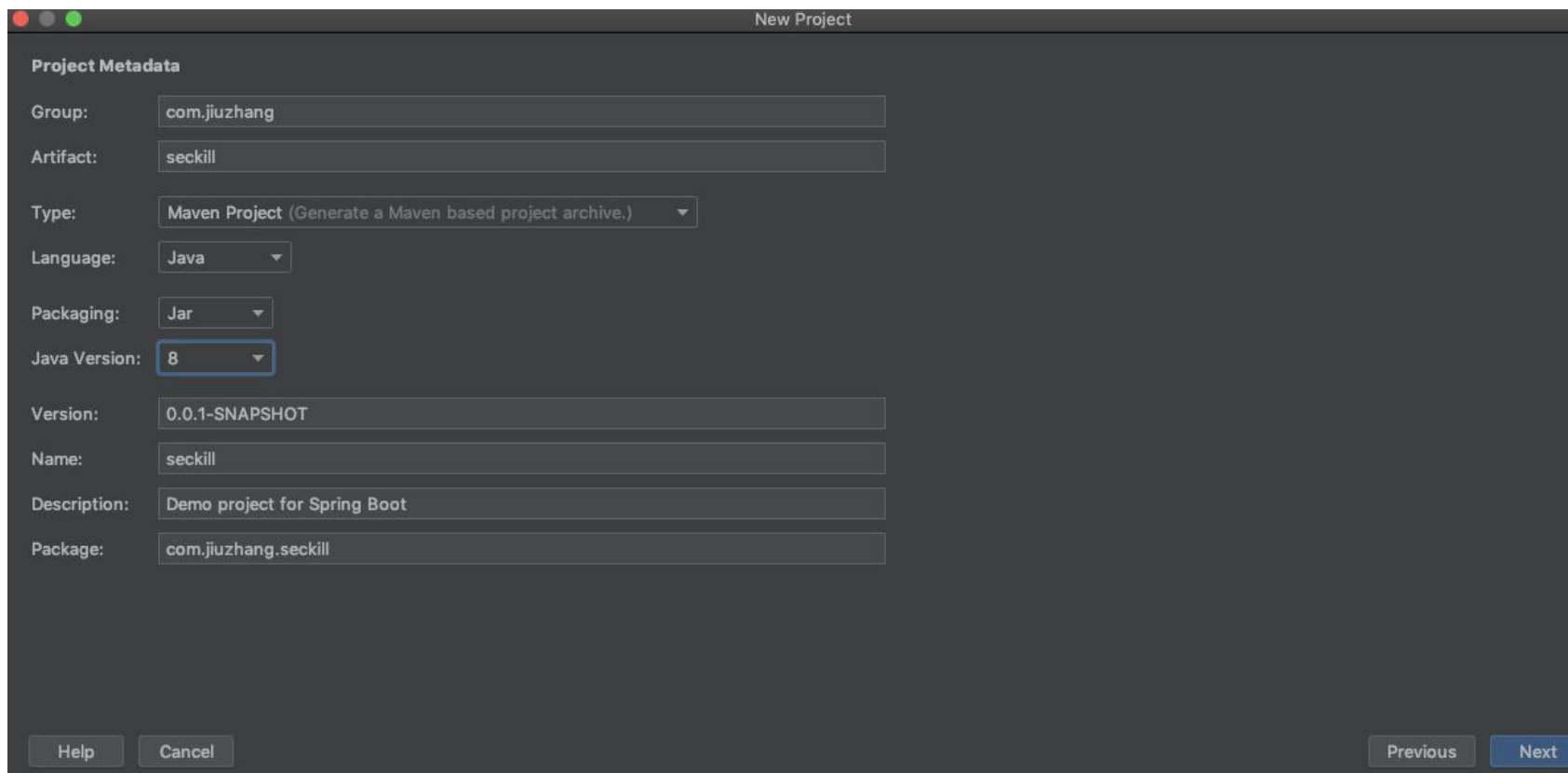
1. 使用集成的 Spring Initializer 创建项目

默认使用 start.spring.io 作为创建服务



2. 配置项目信息

如包名、JDK 版本等



New Project

Project Metadata

Group: com.jiuzhang

Artifact: seckill

Type: Maven Project (Generate a Maven based project archive.)

Language: Java

Packaging: Jar

Java Version: 8

Version: 0.0.1-SNAPSHOT

Name: seckill

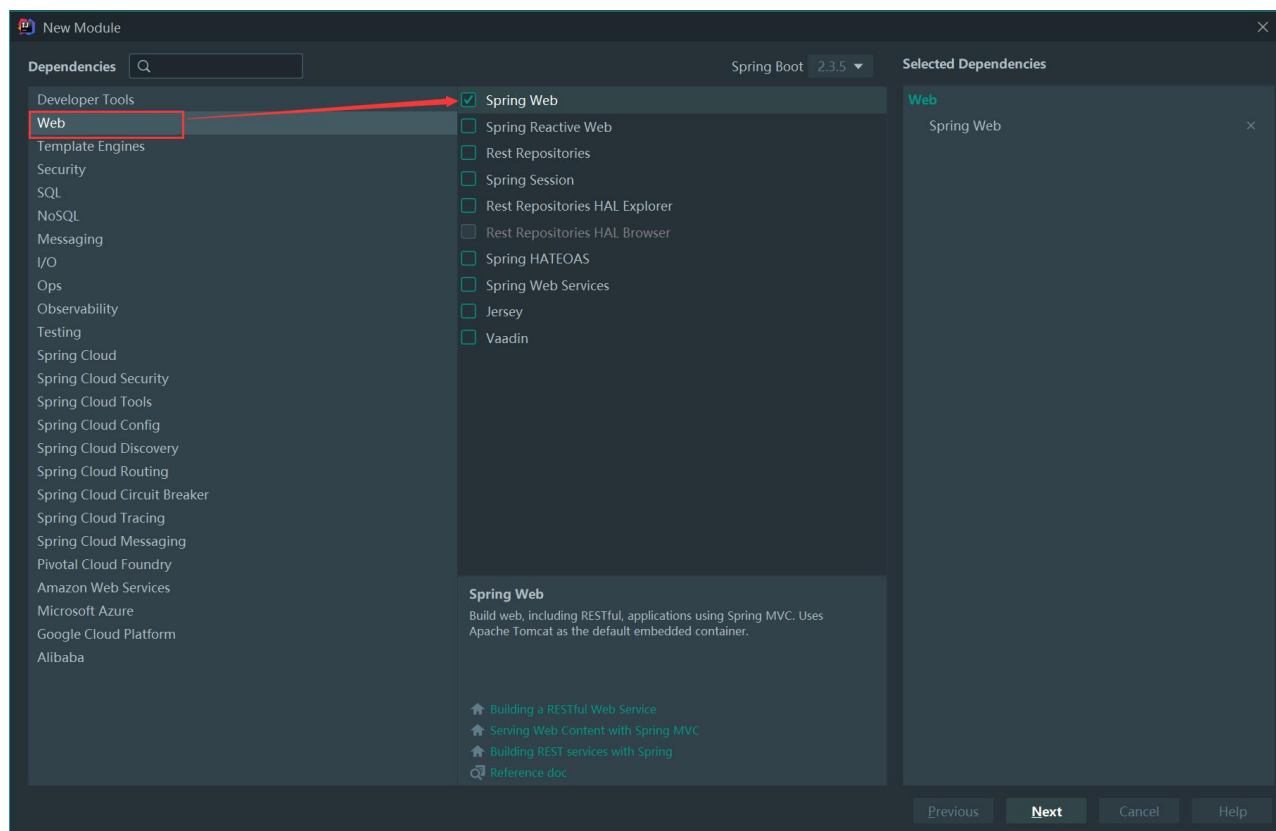
Description: Demo project for Spring Boot

Package: com.jiuzhang.seckill

Help Cancel Previous Next

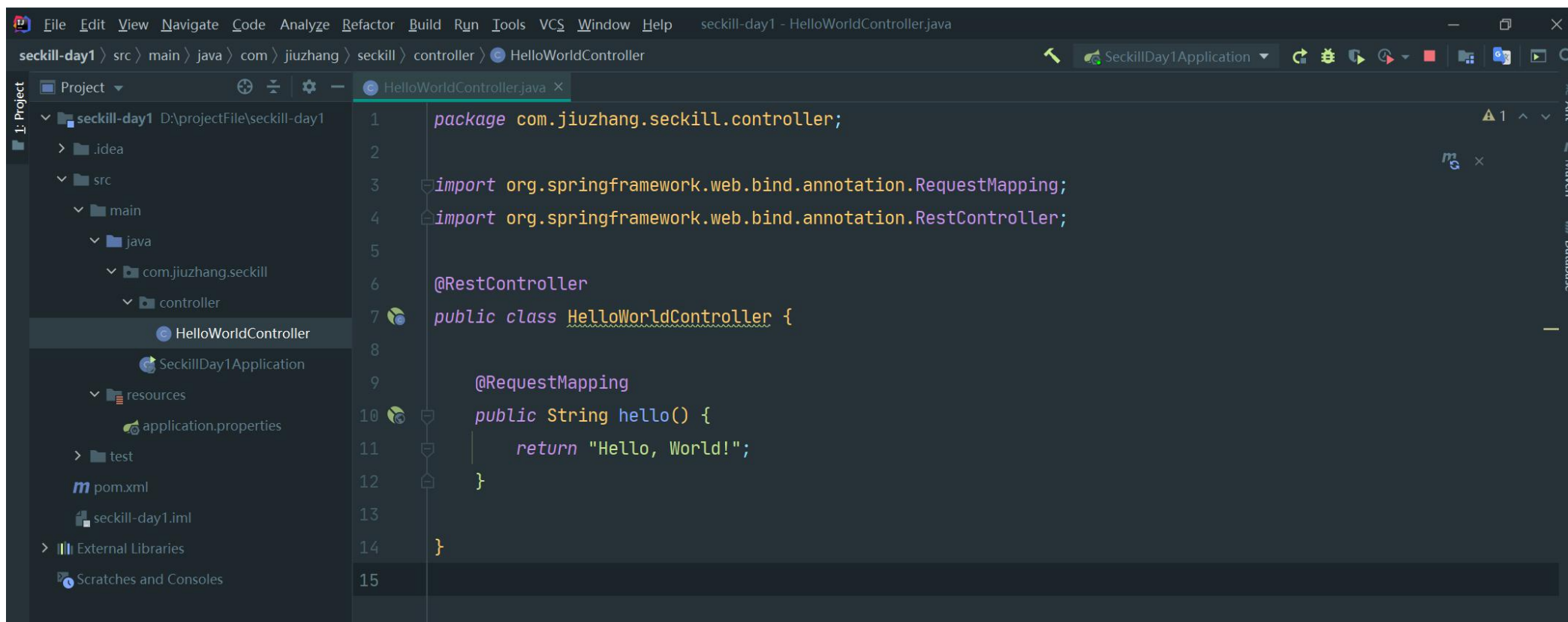
3. 添加必备依赖

我们需要提供 Web 服务，所以 选择 Web 依赖



编写我们的 Hello, world! 控制器

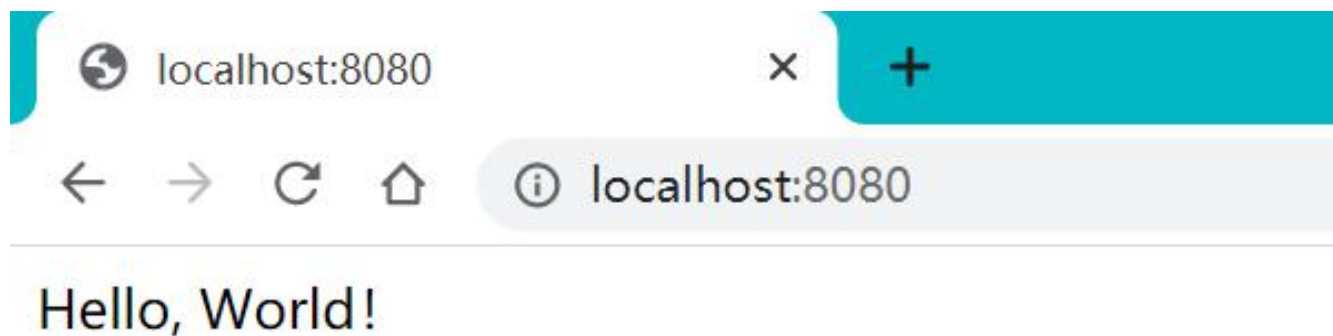
运行项目，看看是否有报错信息



```
1 package com.jiuzhang.seckill.controller;
2
3 import org.springframework.web.bind.annotation.RequestMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class HelloWorldController {
8
9     @RequestMapping
10    public String hello() {
11        return "Hello, World!";
12    }
13
14 }
```

Hello, world! 效果展示

访问 <http://localhost:8080> 查看运行效果



项目成功运行了吗？

如果有启动报错，可以在问答区或微信群中提问

我们来看一看项目的配置文件

pom.xml 与 properties 配置分别是做什么的

pom.xml 文件有什么作用?

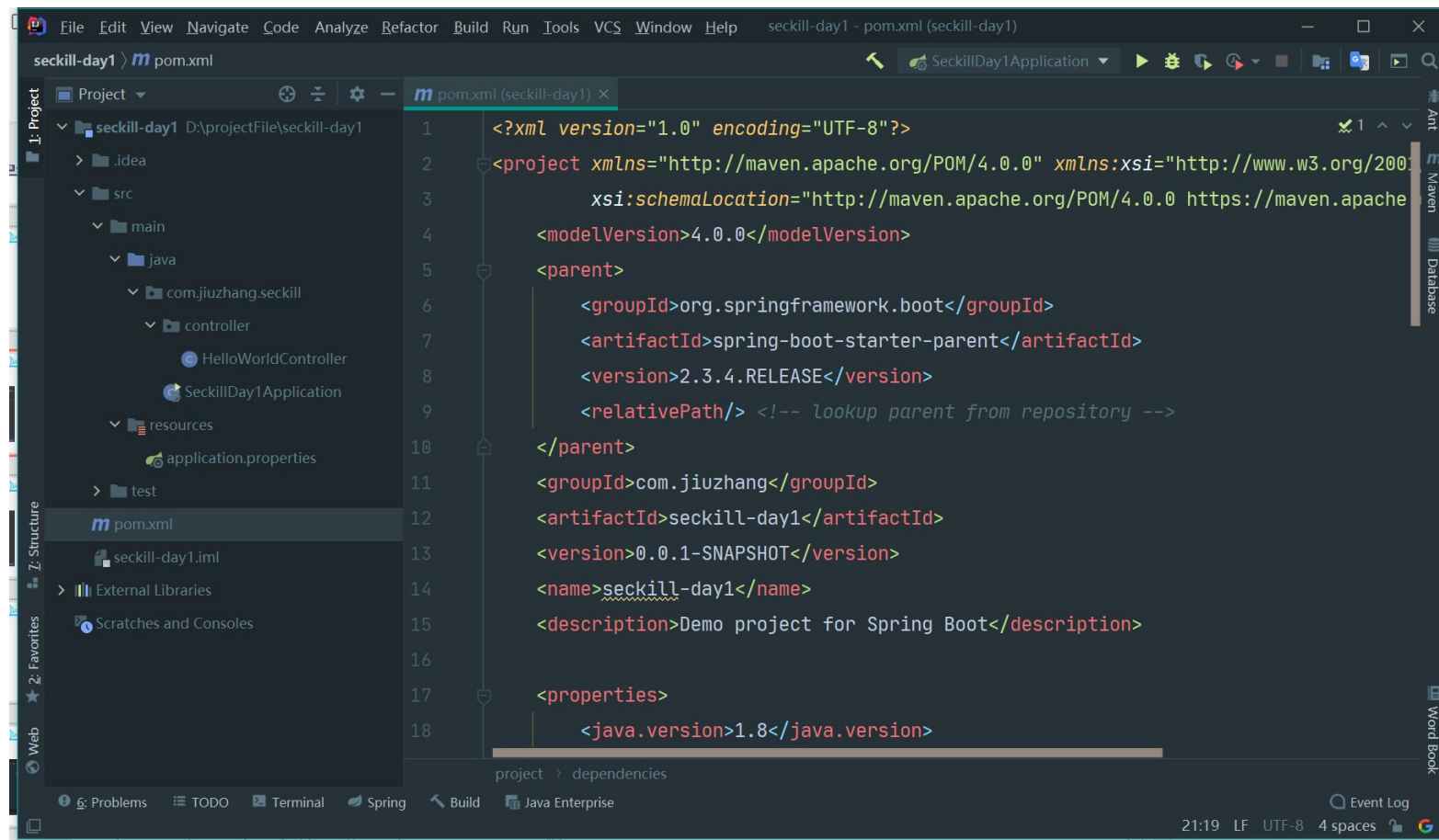
涉及到了哪些概念?

POM = Project Object Model

项目对象模型

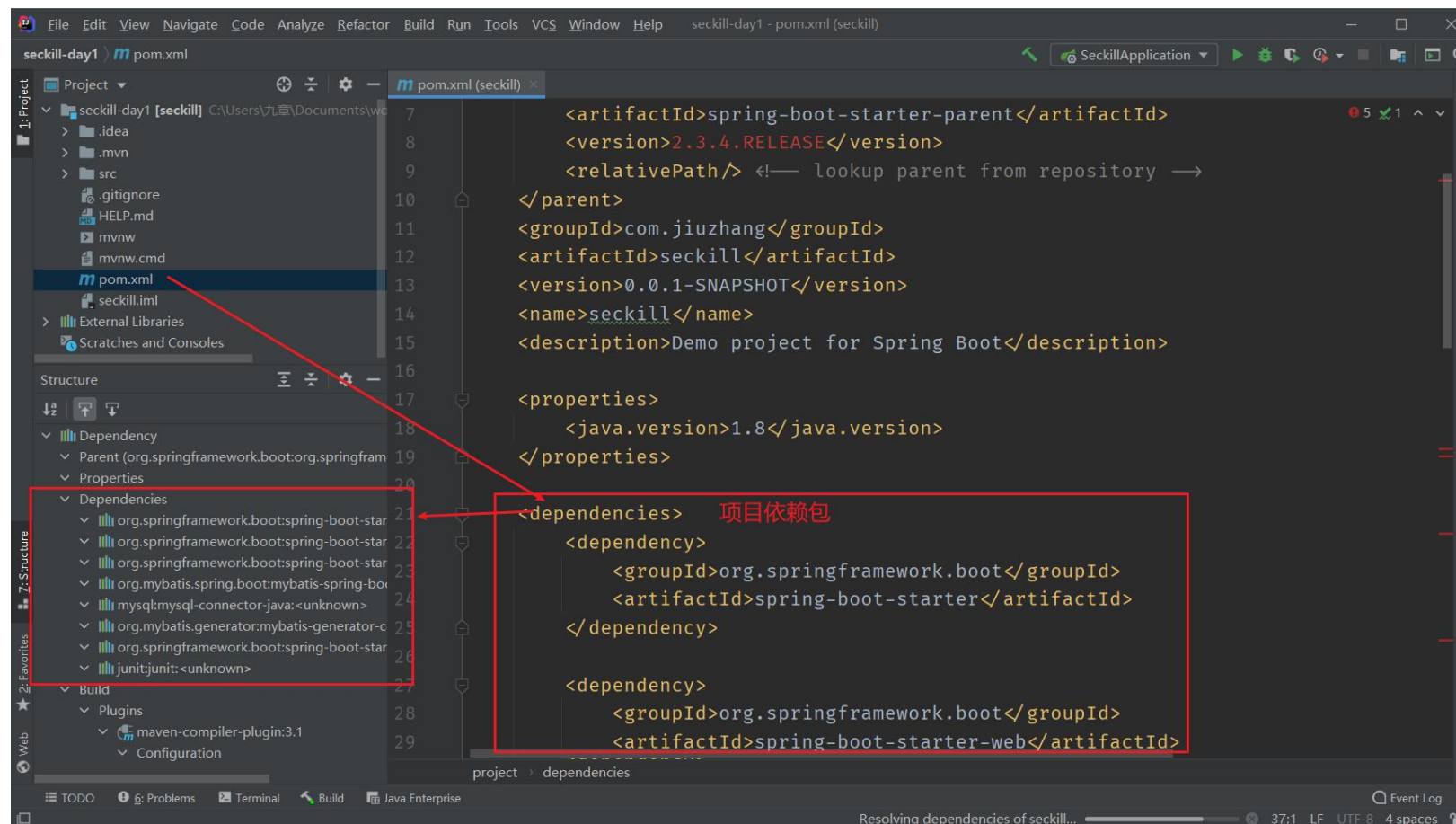
pom.xml 文件有什么作用?

涉及到了哪些概念?



到目前为止，我们引入了哪些依赖？

分别有什么作用？



什么是启动类？有什么用处？

启动类承担什么功能？

```
@SpringBootApplication
    @MapperScan("com.jiuzhang.seckill.db.mappers")
    @ComponentScan(basePackages = {"com.jiuzhang"})
    public class SeckillApplication {

        public static void main(String[] args) {
            SpringApplication.run(SeckillApplication.class, args);
        }

    }
```

3. 秒杀活动发布功能

设计并实现相关数据表及功能

秒杀发布功能要实现什么？

我们通过哪些途径去实现？

秒杀活动发布页面

01

02

怎么设计数据表

03

04

提交哪些信息

后端怎么处理

哪些字段是必须的？

如何通过需求反推数据库表的设计？

新增秒杀活动信息

秒杀活动名称:

商品ID:

秒杀价格:

商品原价:

秒杀商品数量:

开始时间:

结束时间:

秒杀活动表应该有哪些对应的字段？

结合刚才的活动表单来设计

字段 column	类型 type	解释 explain
id	BIGINT	秒杀活动ID
name	VARCHAR	秒杀活动名称
commodity_id	BIGINT	商品ID
old_price	DECIMAL	商品原价
seckill_price	DECIMAL	秒杀价格
start_time	DATETIME	秒杀开始时间
end_time	DATETIME	秒杀结束时间
total_stock	BIGINT	秒杀商品的库存数量

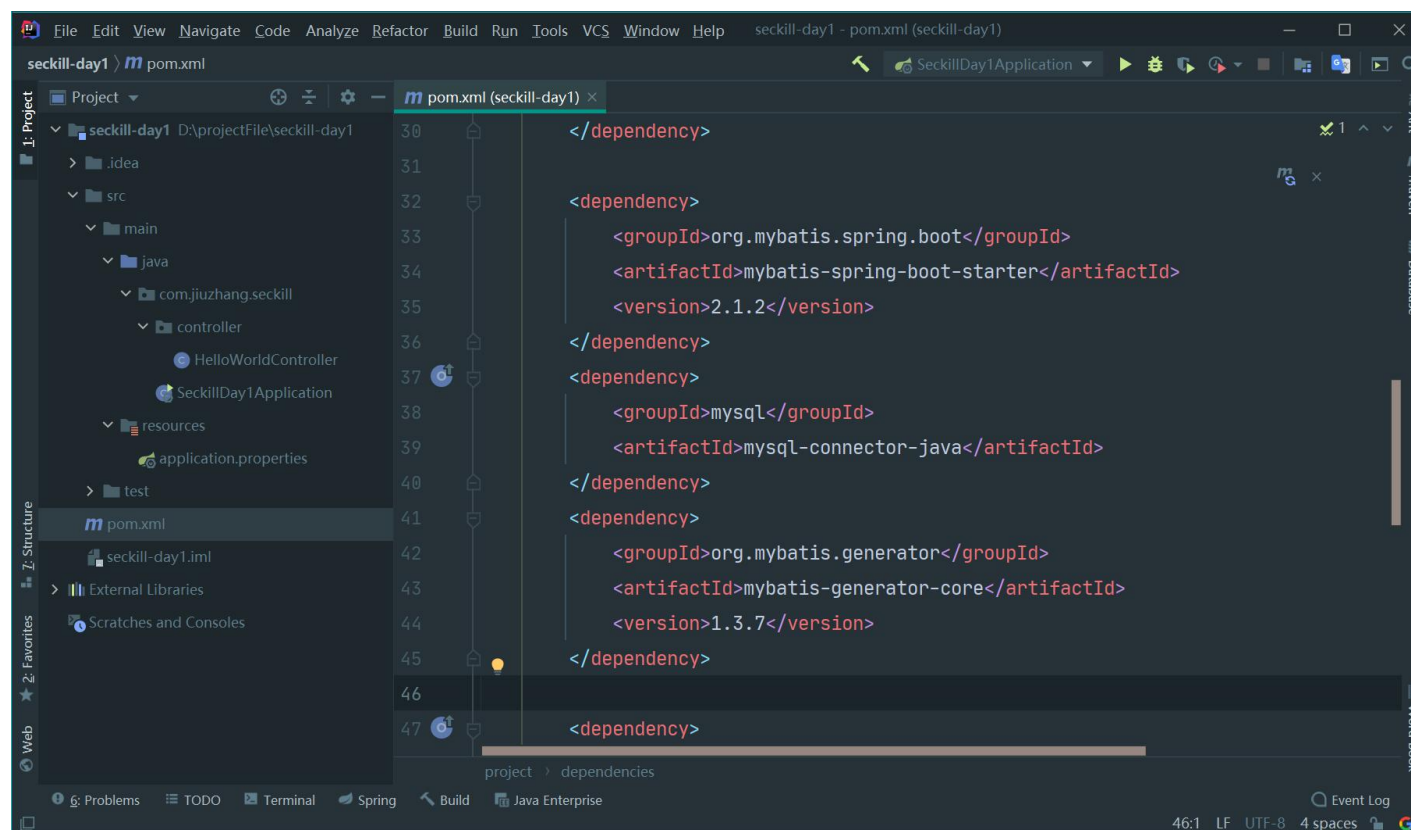
商品表应该怎么设计？

与秒杀活动有什么关联？

字段 column	类型 type	解释 explain
id	BIGINT	商品ID
name	VARCHAR	商品名称
desc	VARCHAR	商品描述信息
price	DECIMAL	商品价格

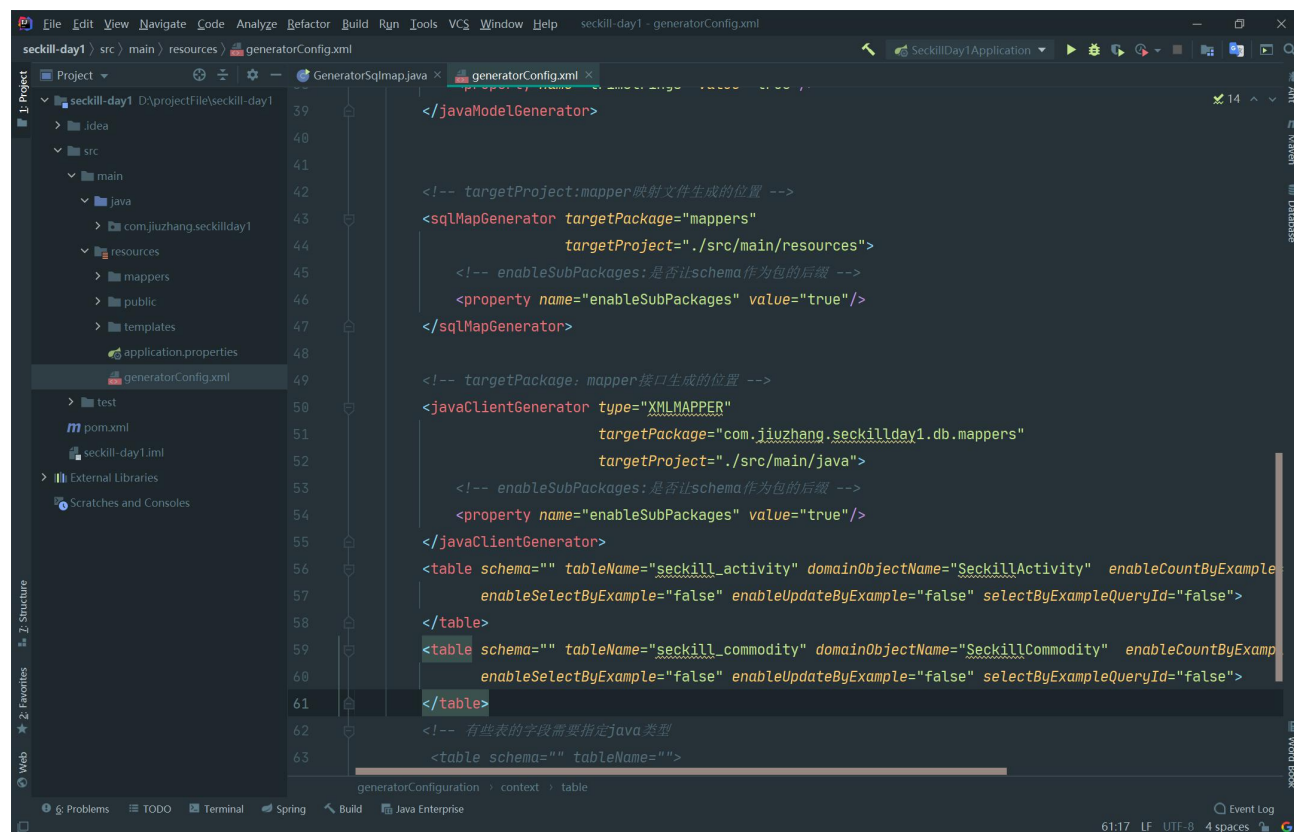
还记得 pom.xml 的用途吗？

修改 pom.xml 添加 MyBatis 相关依赖



MyBatis 的逆向生成配置

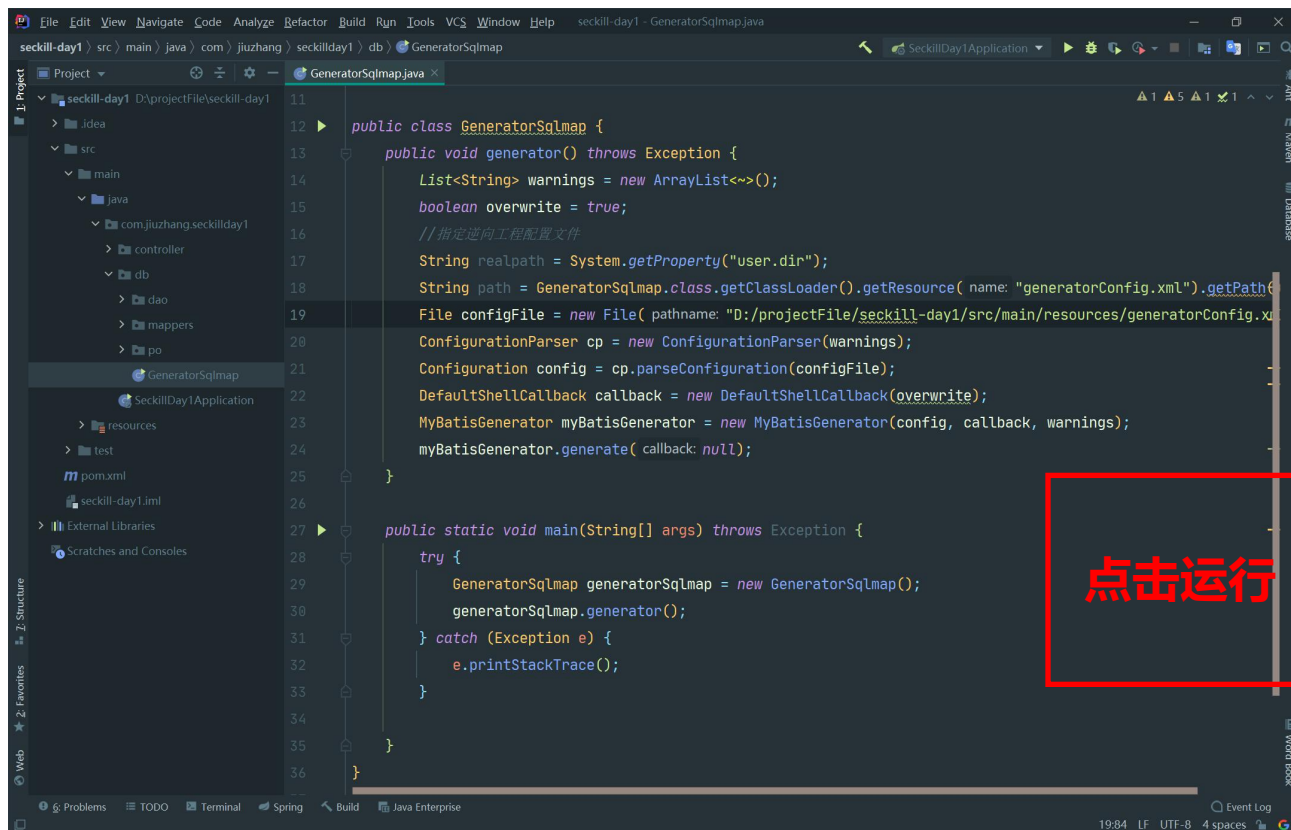
配置数据库连接 (与 properties 配置不同)



```
39 </javaModelGenerator>
40
41
42 <!-- targetProject: mapper 映射文件生成的位置 -->
43 <sqlMapGenerator targetPackage="mappers"
44                 targetProject="./src/main/resources">
45     <!-- enableSubPackages: 是否让 schema 作为包的后缀 -->
46     <property name="enableSubPackages" value="true"/>
47 </sqlMapGenerator>
48
49 <!-- targetPackage: mapper 接口生成的位置 -->
50 <javaClientGenerator type="XMLMAPPER"
51                    targetPackage="com.jiuzhang.seckillday1.db.mappers"
52                    targetProject="./src/main/java">
53     <!-- enableSubPackages: 是否让 schema 作为包的后缀 -->
54     <property name="enableSubPackages" value="true"/>
55 </javaClientGenerator>
56 <table schema="" tableName="seckill_activity" domainObjectName="SeckillActivity" enableCountByExample="false"
57       enableSelectByExample="false" enableUpdateByExample="false" selectByExampleQueryId="false">
58 </table>
59 <table schema="" tableName="seckill_commodity" domainObjectName="SeckillCommodity" enableCountByExample="false"
60       enableSelectByExample="false" enableUpdateByExample="false" selectByExampleQueryId="false">
61 </table>
62 <!-- 有些表的字段需要指定 java 类型 -->
63 <table schema="" tableName="">
```

使用 MyBatis 逆向生成 Mapper 相关代码

运行 main 方法逆向生成 Mapper 相关代码

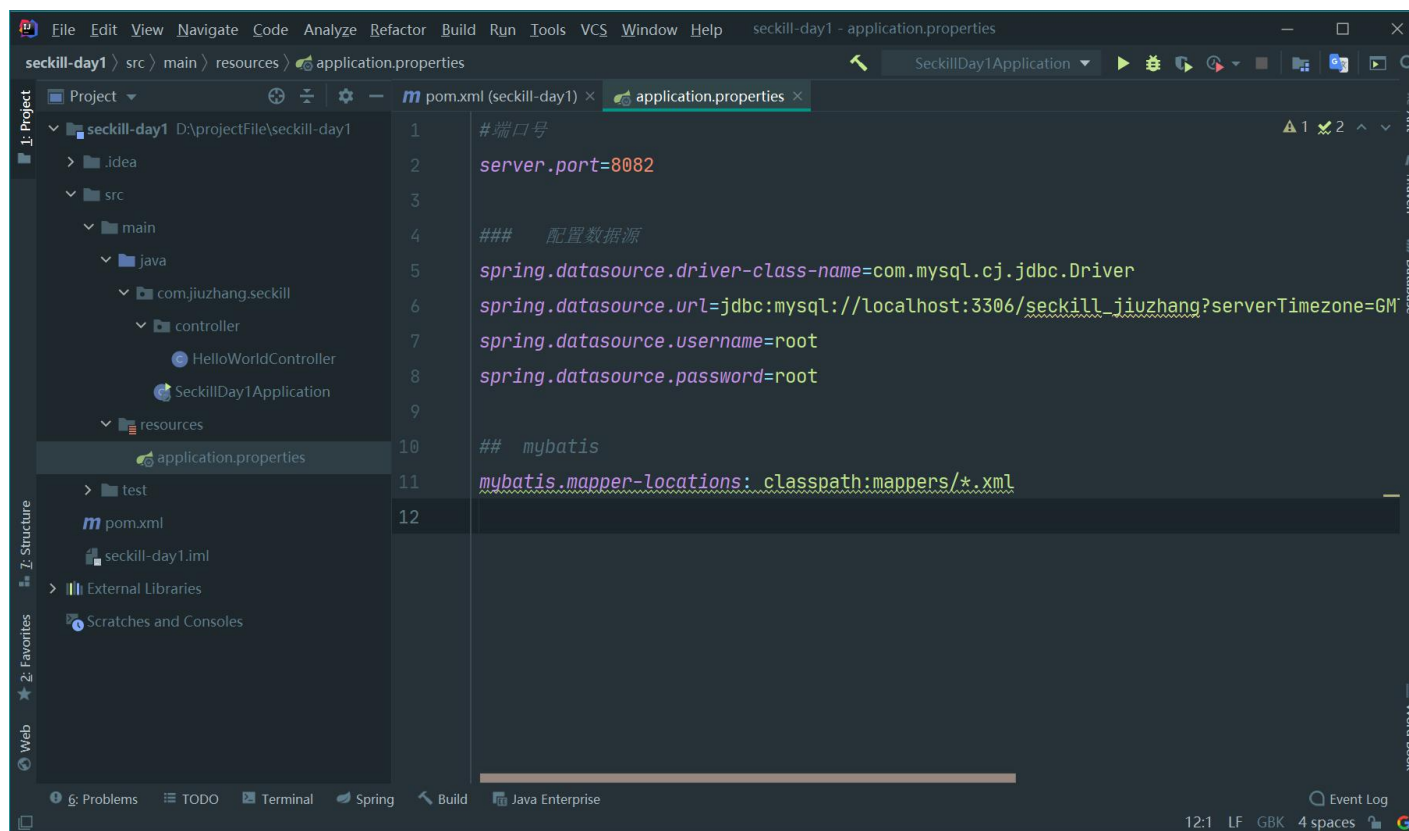


```
11  
12 public class GeneratorSqlmap {  
13     public void generator() throws Exception {  
14         List<String> warnings = new ArrayList<>();  
15         boolean overwrite = true;  
16         // 指定逆向工程配置文件  
17         String realpath = System.getProperty("user.dir");  
18         String path = GeneratorSqlmap.class.getClassLoader().getResource( name: "generatorConfig.xml").getPath();  
19         File configFile = new File( pathname: "D:/projectFile/seckill-day1/src/main/resources/generatorConfig.xml");  
20         ConfigurationParser cp = new ConfigurationParser(warnings);  
21         Configuration config = cp.parseConfiguration(configFile);  
22         DefaultShellCallback callback = new DefaultShellCallback(overwrite);  
23         MyBatisGenerator myBatisGenerator = new MyBatisGenerator(config, callback, warnings);  
24         myBatisGenerator.generate( callback: null);  
25     }  
26  
27     public static void main(String[] args) throws Exception {  
28         try {  
29             GeneratorSqlmap generatorSqlmap = new GeneratorSqlmap();  
30             generatorSqlmap.generator();  
31         } catch (Exception e) {  
32             e.printStackTrace();  
33         }  
34     }  
35 }  
36 }
```

点击运行 main 方法即可生成

配置数据库连接

驱动、数据源、用户密码等



The screenshot shows an IDE window titled 'seckill-day1 - application.properties'. The left sidebar displays the project structure for 'seckill-day1', including 'src/main/resources' where 'application.properties' is located. The main editor area shows the following configuration:

```
1 #端口号
2 server.port=8082
3
4 ### 配置数据源
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
6 spring.datasource.url=jdbc:mysql://localhost:3306/seckill_jiuzhang?serverTimezone=GMT
7 spring.datasource.username=root
8 spring.datasource.password=root
9
10 ## mybatis
11 mybatis.mapper-locations: classpath:mappers/*.xml
12
```

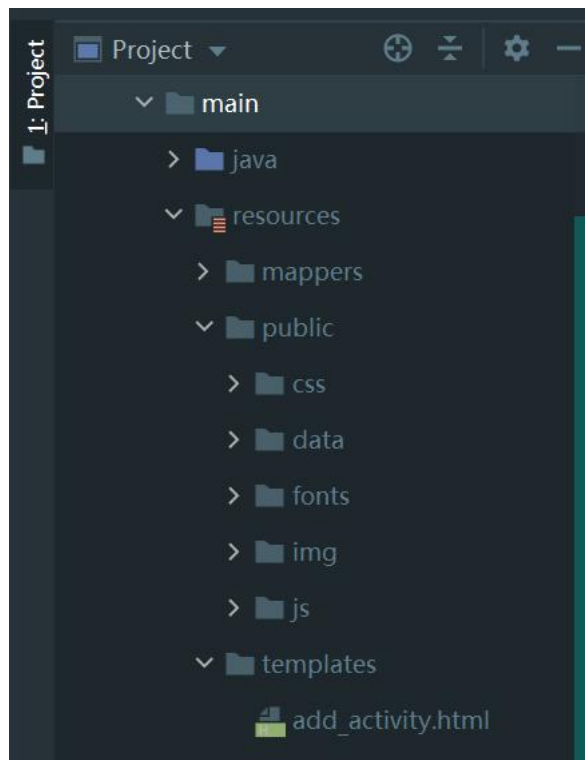
测试 MyBatis 配置与连接

编写测试类并运行测试

```
@Test
void SeckillActivityTest() {
    SeckillActivity seckillActivity = new SeckillActivity();
    seckillActivity.setName("测试");
    seckillActivity.setCommodityId(999L);
    seckillActivity.setTotalStock(100L);
    seckillActivity.setOldPrice(new BigDecimal( val: 100));
    seckillActivity.setSeckillPrice(new BigDecimal( val: 99));
    seckillActivity.setActivityStatus(new Short( s: "10"));
    seckillActivity.setAvailableStock(10L);
    seckillActivity.setLockStock(0L);
    seckillActivityMapper.insert(seckillActivity);
    System.out.println("====>>>" + seckillActivityMapper.selectByPrimaryKey( id: 1L));
}
```

templates 模板资源文件夹

加入发布页面



在 Controller 中编写 RequestMapping

返回渲染好的模板文件

```
@RequestMapping("/addSeckillActivity")
public String addSeckillActivity() {
    return "add_activity";
}
```


秒杀活动发布页

点击提交后，数据会被 Spring Boot 处理并写入数据库

请登录 | 免费注册

我的订单 | 我的购物车 | 企业采购 | 合作招商 | 商家后台 | 网站导航

九章算法

搜索

新增秒杀活动信息

秒杀活动名称:

商品ID:

秒杀价格:

商品原价:

秒杀商品数量:

开始时间:

年 / 月 / 日 ----:--

结束时间:

年 / 月 / 日 ----:--

提交

正品保障
正品保障，提供发票

正品保障
正品保障，提供发票

正品保障
正品保障，提供发票

正品保障
正品保障，提供发票

正品保障
正品保障，提供发票

如何在 Controller 处理请求?

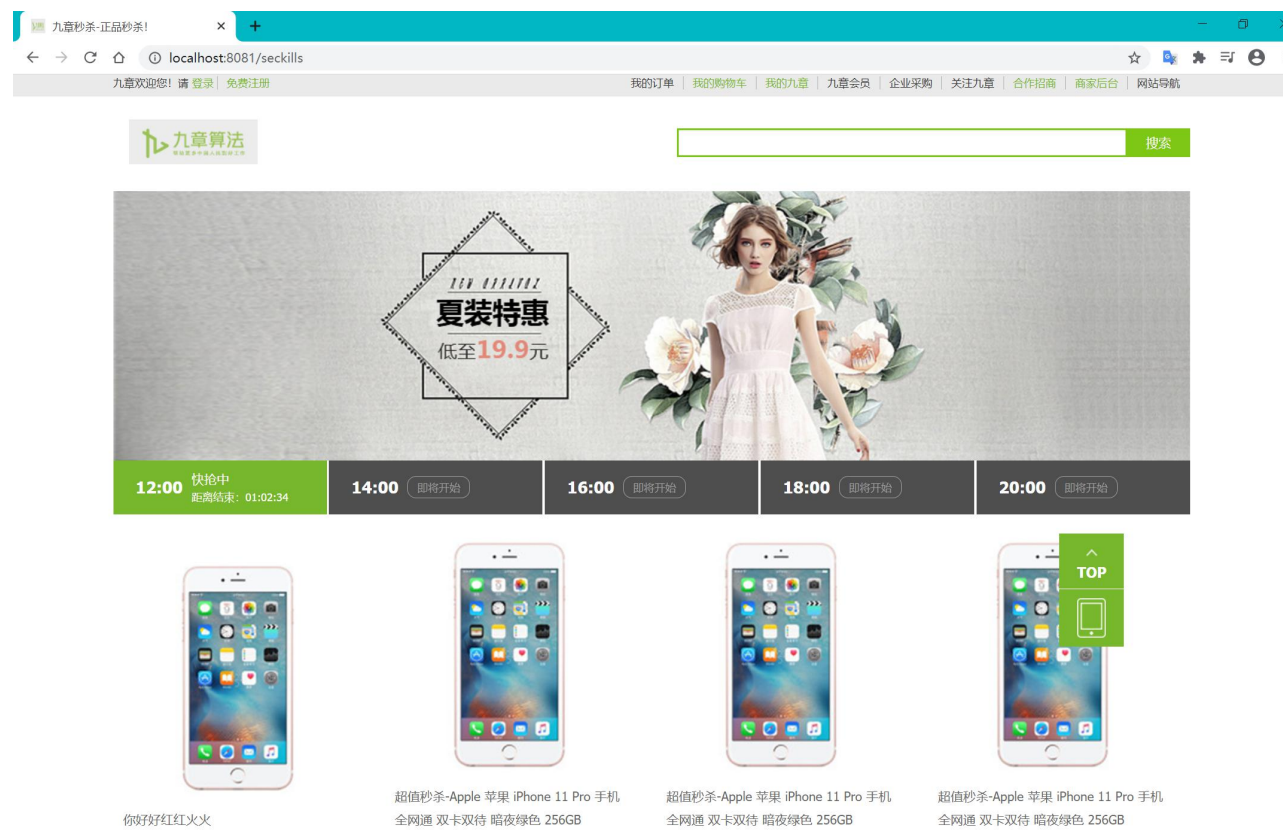
如何在 Controller 处理请求

```
@ResponseBody
@RequestMapping("/addSeckillActivityAction")
public String addSeckillActivityAction(
    @RequestParam("name") String name,
    @RequestParam("commodityId") long commodityId,
    @RequestParam("seckillPrice") BigDecimal seckillPrice,
    @RequestParam("oldPrice") BigDecimal oldPrice,
    @RequestParam("seckillNumber") long seckillNumber
) {
    SeckillActivity seckillActivity = new SeckillActivity();
    seckillActivity.setName(name);
    seckillActivity.setCommodityId(commodityId);
    seckillActivity.setSeckillPrice(seckillPrice);
    seckillActivity.setOldPrice(oldPrice);
    seckillActivity.setTotalStock(seckillNumber);
    seckillActivity.setAvailableStock(seckillNumber);
    seckillActivity.setLockStock(0L);
    seckillActivity.setActivityStatus(1);
    seckillActivityDao.inertSeckillActivity(seckillActivity);
    return seckillActivity.toString();
}
```

4. 秒杀活动列表页面

4.1 功能介绍

一个活动列表要有哪些属性和功能



4.2 Spring Boot 跳转活动列表页

1. Spring Boot 跳转活动列表页

```
/**
 * 查询秒杀活动的列表
 *
 * @param resultMap
 * @return
 */
@RequestMapping("/seckill")
public String successTest(Map<String, Object> resultMap) {
    List<SeckillActivity> seckillActivities = seckillActivityDao.querySeckillActivitiysByStatus( activityStatus: 1);
    resultMap.put("seckillActivities", seckillActivities);
    return "seckill_activity_list";
}
```

4.2 Spring Boot 跳转活动列表页

2. 秒杀活动列表

```
<tr th:each="seckillActivity : ${seckillActivities}">
  <li class="seckill-item" >
    <div class="pic">
      
    </div>
    <div class="intro">
      <span th:text="${seckillActivity.name}">活动名称</span>
    </div>
    <div class="price">
      <b class="sec-price" th:text="'¥'+${seckillActivity.seckillPrice}"></b>
      <b class="ever-price" th:text="'¥'+${seckillActivity.oldPrice}"></b>
    </div>
    <div class="num">
      <div>已售999</div>
      <div class="progress">
        <div class="sui-progress progress-danger">
          <span style='...' class="bar"></span>
        </div>
      </div>
      <div>剩余
        <b class="owned">999</b>件</div>
    </div>
    <a class="sui-btn btn-block btn-buy" th:href="@{'/seckill/detail/'+${seckillActivity.id}}" target='_blank'>立即抢购</a>
  </li>
</tr>
```

4.3 功能测试与演示

发布秒杀活动，并在活动列表页面检查结果

4.4 秒杀商品详情页

详情页请求

```
/**
 * 秒杀商品详情
 *
 * @param resultMap
 * @param seckillActivityId
 * @return
 */
@RequestMapping("/seckill/detail/{seckillActivityId}")
public String itemPage(Map<String, Object> resultMap, @PathVariable long seckillActivityId) {
    SeckillActivity seckillActivity = seckillActivityDao.querySeckillActivityById(seckillActivityId);
    SeckillCommodity seckillCommodity = seckillCommodityDao.querySeckillCommodityById(seckillActivity.getCommodityId());

    resultMap.put("seckillActivity", seckillActivity);
    resultMap.put("seckillCommodity", seckillCommodity);
    resultMap.put("seckillPrice", seckillActivity.getSeckillPrice());
    resultMap.put("oldPrice", seckillActivity.getOldPrice());
    resultMap.put("commodityId", seckillActivity.getCommodityId());
    resultMap.put("commodityName", seckillCommodity.getCommodityName());
    resultMap.put("commodityDesc", seckillCommodity.getCommodityDesc());
    return "seckill_detail";
}
```

回顾与总结

回顾并总结本节主要的知识点

本节课学习总结

2. 创建项目



- ☐ Spring Boot 介绍
- ☐ 新建项目
- ☐ Hello, world!
- ☐ pom.xml 依赖配置
- ☐ 启动类配置

3. 秒杀活动发布功能



- ☐ 功能介绍
- ☐ 表单字段
- ☐ 设计秒杀活动表
- ☐ Spring Boot 整合 MyBatis
- ☐ MyBatis 逆向生成
- ☐ properties 配置数据库连接
- ☐ Spring Boot 跳转发布页
- ☐ 处理发布页面的表单请求

4. 秒杀活动列表页面



- ☐ 功能介绍
- ☐ Spring Boot 跳转活动列表页
- ☐ 功能测试与演示

2. 库存扣减功能

3. 库存超卖问题

4. 解决超卖问题



☐ 扣减库存简单处理

☐ Jmeter 并发请求测试



☐ 超卖原因分析

☐ 超卖问题解决方案

☐ 数据库乐观锁方案

☐ Redis Lua 脚本方案



☐ 项目中引入 Redis

☐ 库存扣减 Lua 脚本

☐ 秒杀库存扣减流程开发

☐ Jmeter测试是否解决并发超卖问题

下节课计划预览

下节课环境准备

JMeter 5.0

[下载地址](#)

[历史版本下载地址](#)

[Jmeter教程\(一\) - 入门](#)

Redis

[Redis 下载地址](#)

[Redis 教程](#)