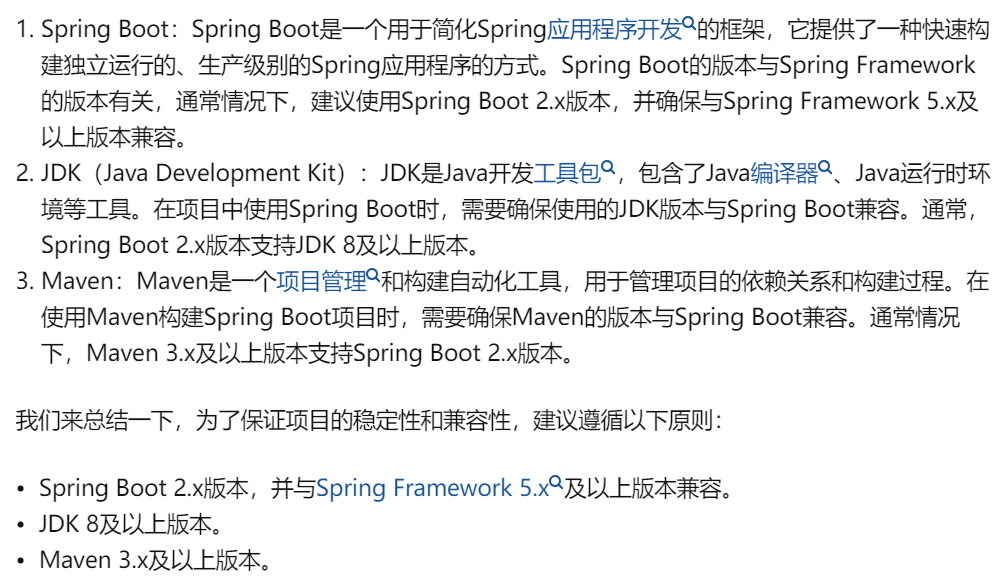
* **项目版本：**

Springboot 2.7.6

Maven 3.6.1

Java 11

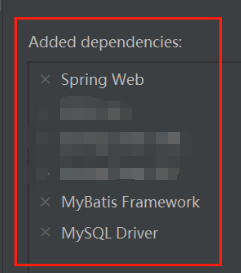
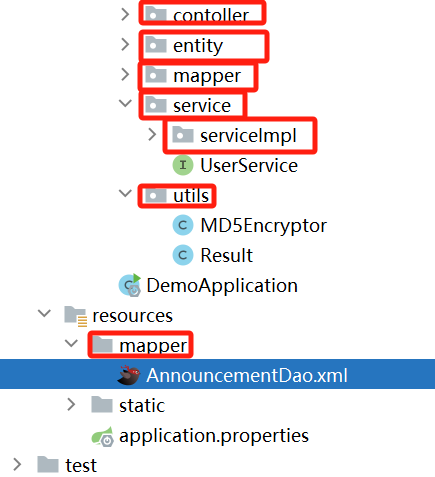
JDK 11



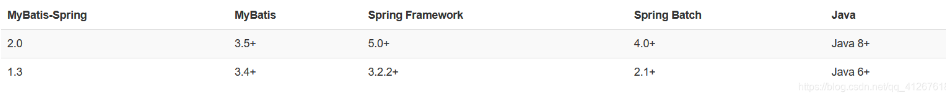
* **项目结构:**

**Controller-->service接口-->serviceImpl-->dao接口-->daoImpl-->mapper-->db**

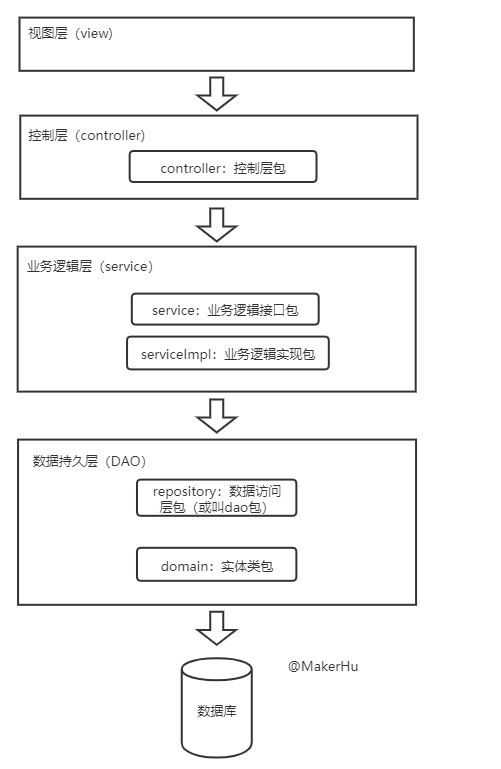
1. **实体类(entity)：数据库表的实体对象，该文件包含实体类的属性和对应属性的set、get方法**
2. **数据访问层(mapper，有些也叫dao层,我们统一使用mapper)：mapper层会调用entity层，mapper中会定义实际使用到的方法，比如增删改查。mapper层的数据源和数据库连接的参数都是在配置文件中进行配置的，配置文件一般在同层的XML文件夹中。数据持久化操作就是指，把数据放到持久化的介质中，同时提供增删改查操作。**
3. **数据服务层(service和serviceImpl两个文件夹，serviceimpl为service 子文件分接口和实现类，接口在service目录下，接口实现类在serviceimpl目录下):负责业务模块的逻辑应用设计。先设计放接口的类，再创建实现的类，然后在配置文件中进行配置其实现的关联。service层调用mapper层接口，接收mapper层返回的数据，完成项目的基本功能设计。封装Service层的业务逻辑有利于业务逻辑的独立性和重复利用性。**
4. **前端控制器(controller):负责具体的业务模块流程的控制，controller层负责前后端交互，接受前端请求，调用service层，接收service层返回的数据，最后返回具体的页面和数据到客户端**
5. **工具类(utils)：通常是一些工具类:一个多功能、基于工具的包。如[字符串处理](https://so.csdn.net/so/search?q=%E5%AD%97%E7%AC%A6%E4%B8%B2%E5%A4%84%E7%90%86&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/Restarting2019/article/details/_blank)、日期处理等，(建立数据库之间的连接），是通用的、与业务无关的，可以独立出来，可供其他项目使用**
6. 配置信息类(config)
7. exception：自定义异常，异常错误码







### 启动类的位置



数据持久层是的目的是在java对象与数据库之间建立映射，也就是说它的作用是将某一个Java类对应到数据库中的一张表。在我们的项目中，就将创建一个实体类User映射到数据库的user表，表中的每个字段对应于实体类的每个属性。而之前配置的JPA的作用就是帮助我们完成类到数据表的映射。

repository: 存放一些数据访问类（也就是一些能操纵数据库的类）的包，比如存放能对user表进行增删改查的类

domain：存放实体类的包，比如User类，其作为对应数据库user表的一个实体类

业务逻辑层的作用是处理业务逻辑。比如在本项目中，我们就在业务逻辑层实现登录注册的逻辑，像是判断是否有用户名重复，密码是否正确等逻辑

service: 存放业务逻辑接口的包

serviceImpl: 存放业务逻辑实现类的包，其中的类实现service中的接口

控制层的作用是接收视图层的请求并调用业务逻辑层的方法。比如视图层请求登录并发来了用户的账号和密码，那么控制层就调用业务逻辑层的登录方法，并将账号密码作为参数传入，在将结果返回给视图层。

controller: 存放控制器的包。比如UserController

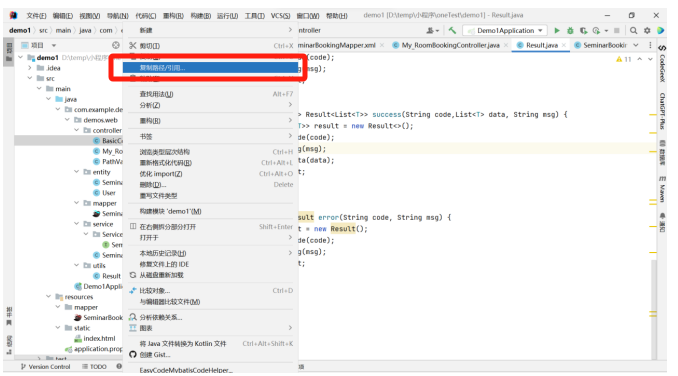
视图层的作用是展现数据，由于本项目写的是纯后端，就不展开解释视图层了。

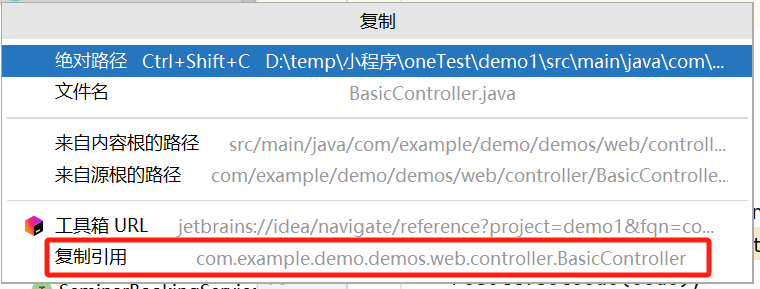
@RestController是一个组合注解，它包含了@Controller和@ResponseBody两个注解的功能。

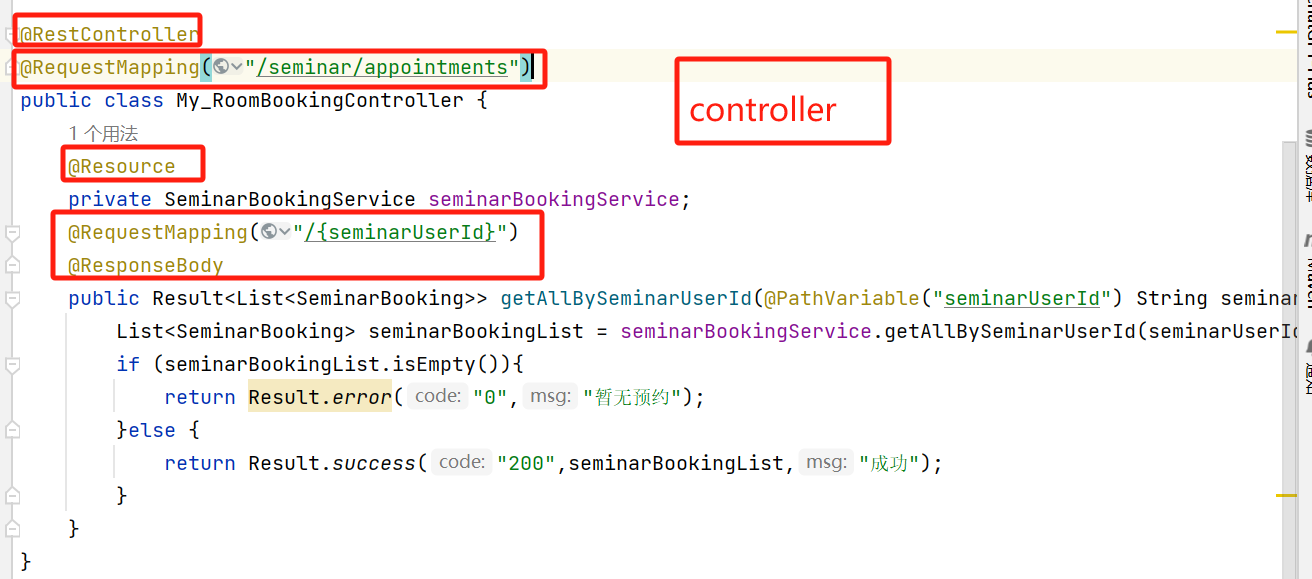
用@RestController标记的类表示这是一个RESTful风格的控制器，它可以处理HTTP请求并返回JSON格式的响应。 @RestController注解在处理请求时，会自动将方法的返回值转换为JSON格式的响应体，并返回给客户端。

因此，使用@RestController可以省去在每个方法上都加@ResponseBody注解的麻烦。

@RestController也支持@RequestMapping注解，用于映射请求







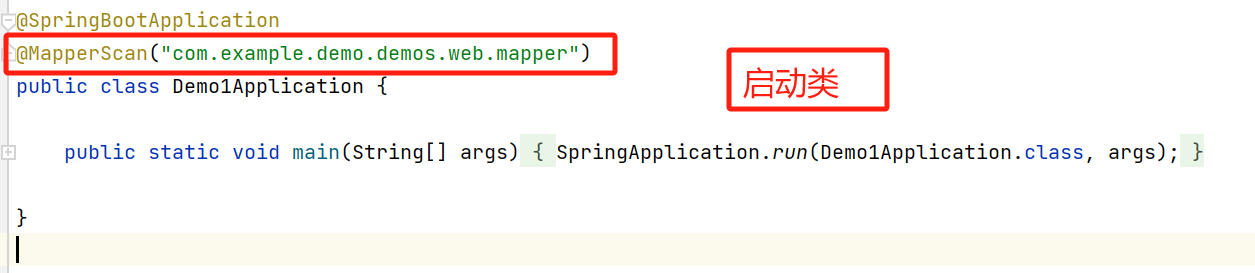












（application.properties）代码版：  
server.port=8081  
mybatis.mapper-locations=classpath:\*\*/\*xml  
mybatis.type-aliases-package=com.example.demo.demos.web.entity  
spring.datasource.url=jdbc:mysql://8.134.201.230:3306/reservationsys?characterEncoding=UTF-8  
spring.datasource.username=ReservationSys  
spring.datasource.password=ydyd3154  
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver