## 环境配置

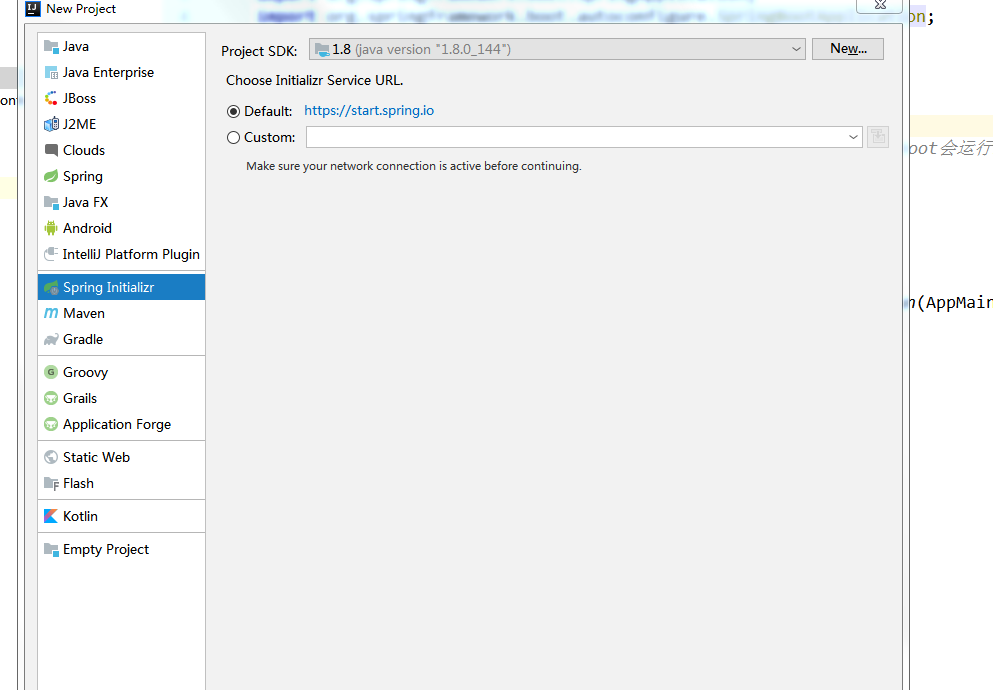
### 1.1手动配置

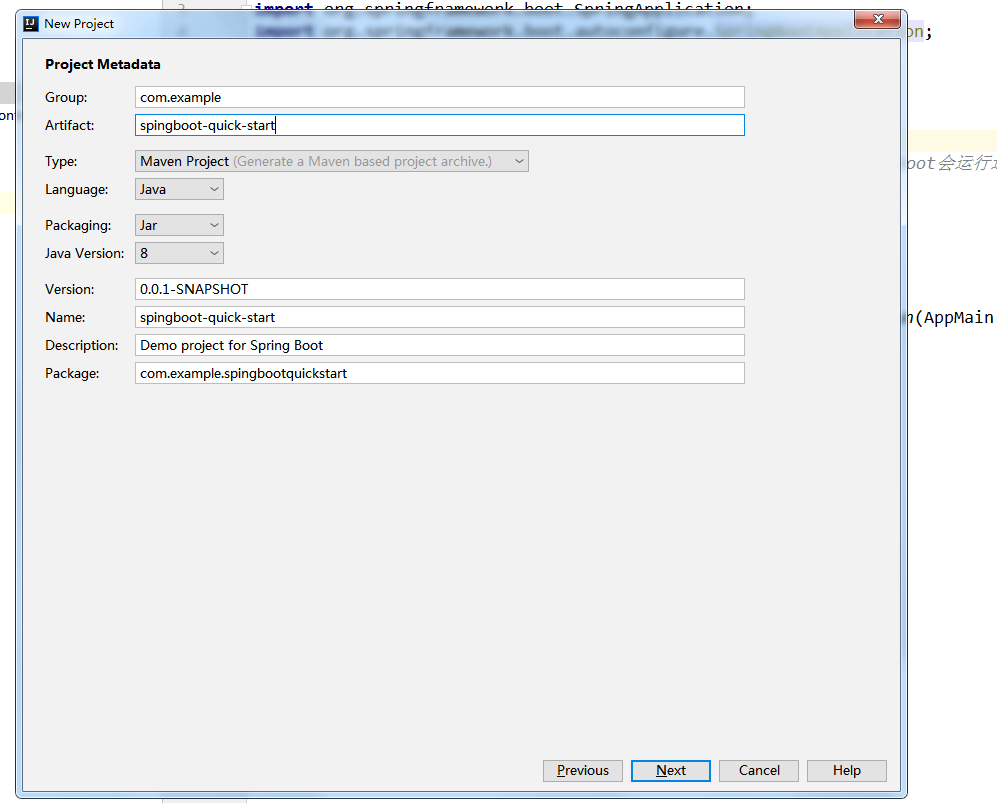
见 sping boot pdf

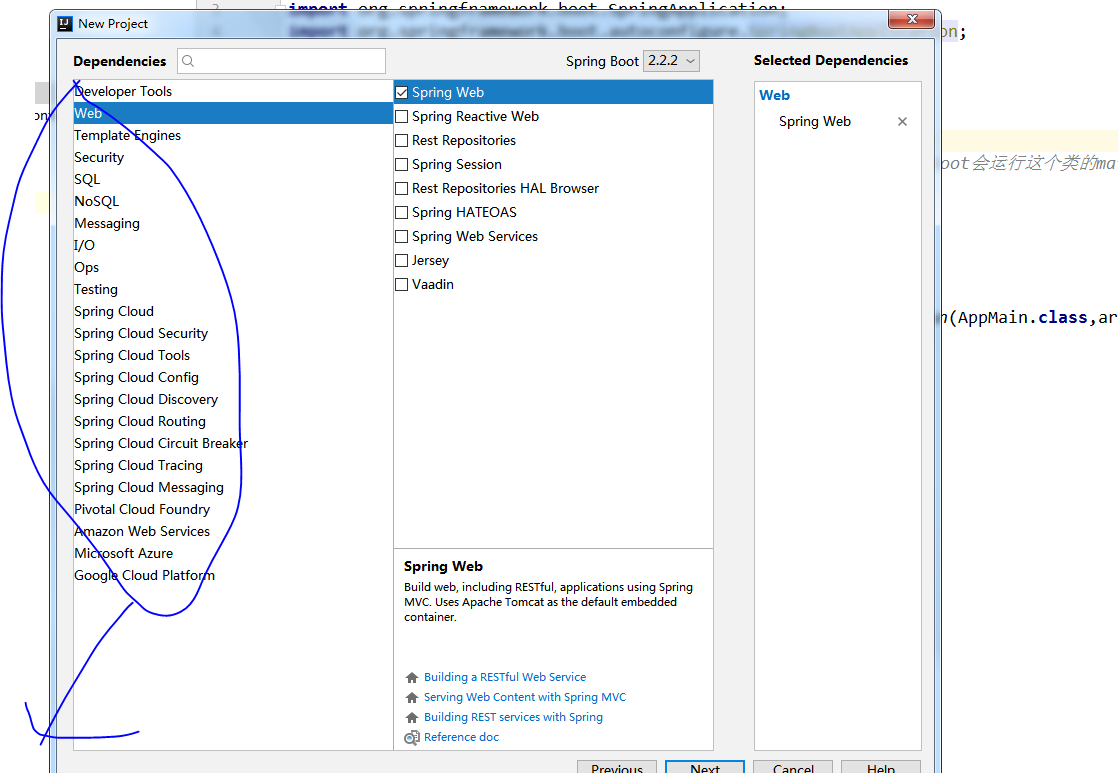
Poml jar包依赖

http://mvnrepository.com/

### 1.2快速配置







模块功能

## 构建系统

### 2.1

使用该设置，您还可以通过覆盖自己项目中的属性来覆盖各个依赖项。例如，要升级到另一个Spring Data发布系列，您可以将以下内容添加到您的pom.xml：

<properties>

<spring-data-releasetrain.version>Fowler-SR2</spring-data-releasetrain.version></properties>

## 数据库

### Jdbc---深入了解之后

jdbc是java与数据库连接的桥梁，用来连接数据库

### Druid

druid数据库连接池

### Mybatis

### springdata jpa

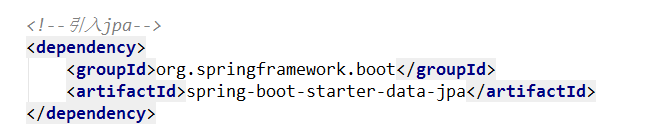
官方文档：

[https://docs.spring.io/spring-data/jpa/docs/2.2.3.RELEASE/reference/html/#jpa.query-methods](https://docs.spring.io/spring-data/jpa/docs/2.2.3.RELEASE/reference/html/" \l "jpa.query-methods)

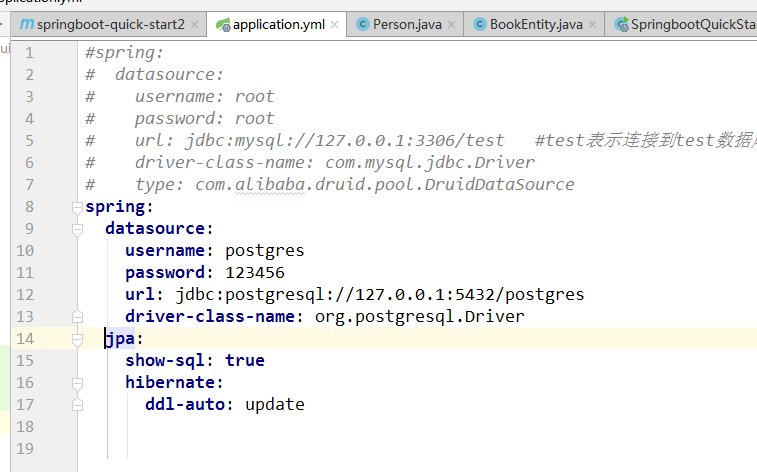
<https://blog.csdn.net/fly910905/article/details/78557110>

4.1配置与使用

#### 4.1引入jpaPom.xml

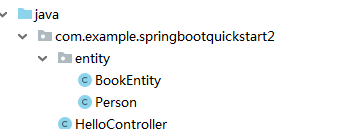


#### 4.2配置jpaYml

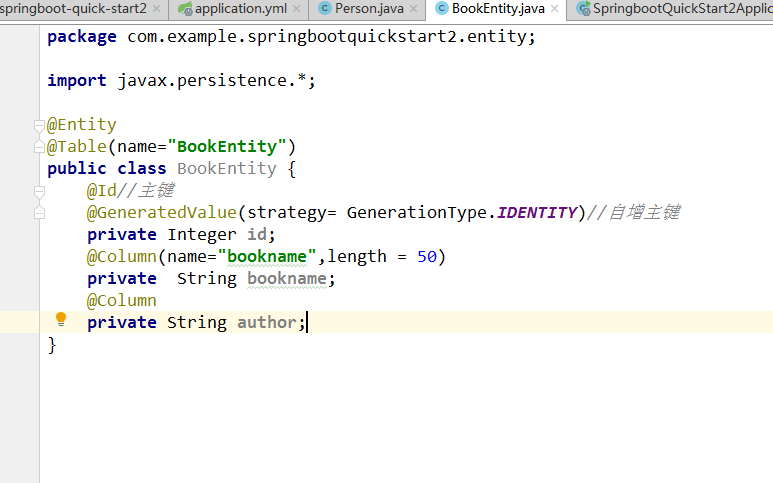


空格一定要对齐！！！！！！

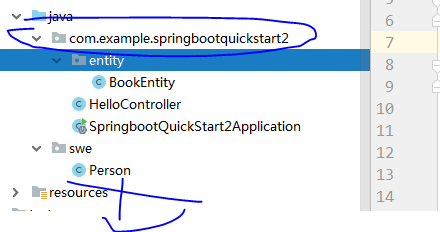
#### 4.3创建实体



自定义的jpa repository的使用



注意：当创建的实体类不在启动类所在包下或子目录下时



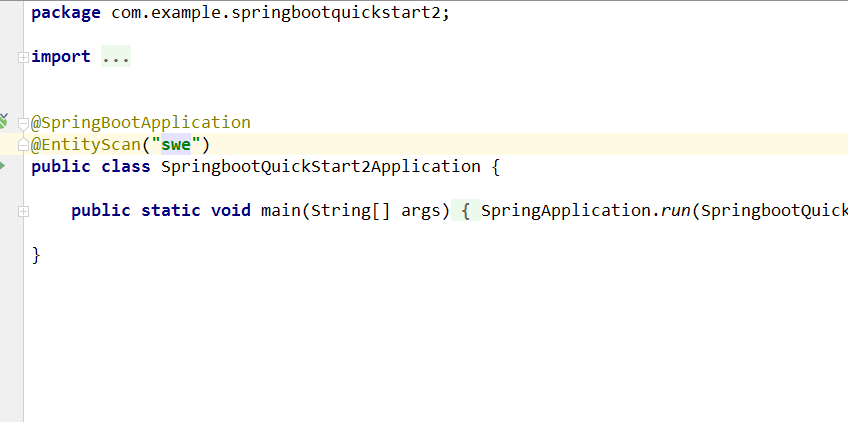
图中的person实体不在启动类所在包下或子目录下

在启动类添加 @EntityScan注解

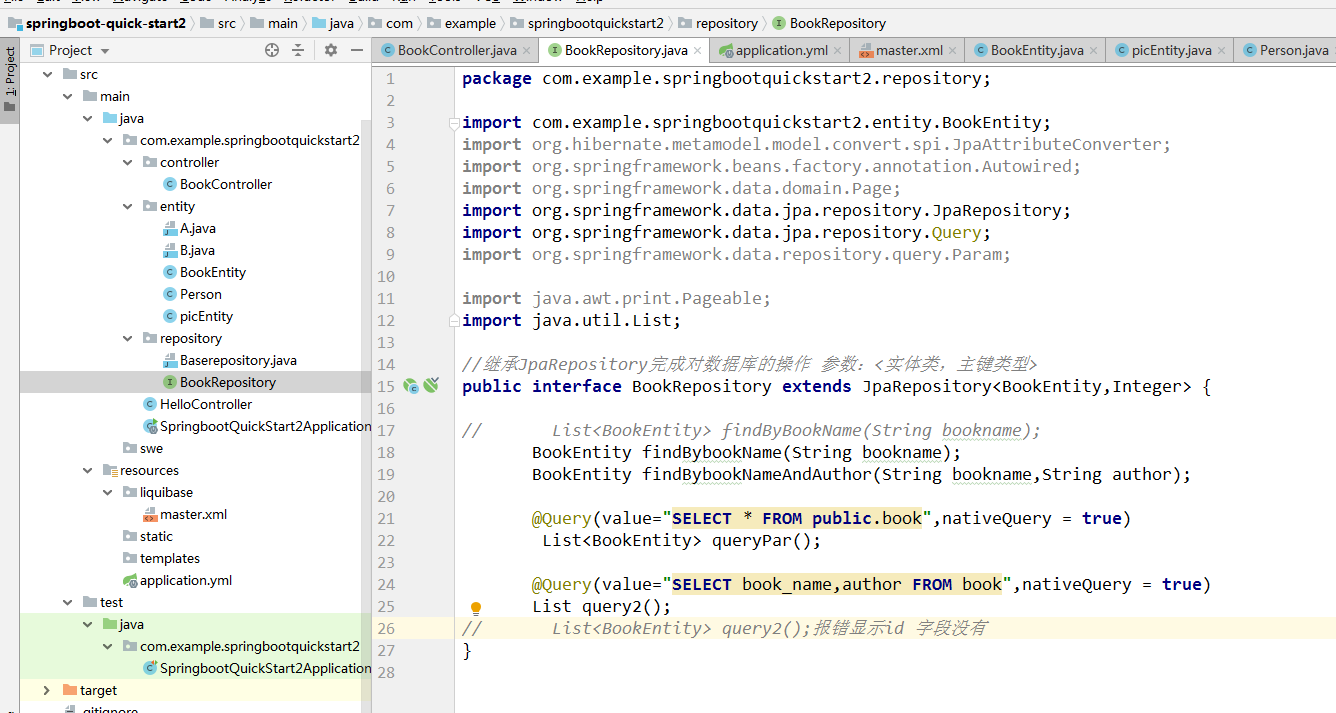
实体类注解使用说明

<https://www.kancloud.cn/cxr17618/springboot/435779>

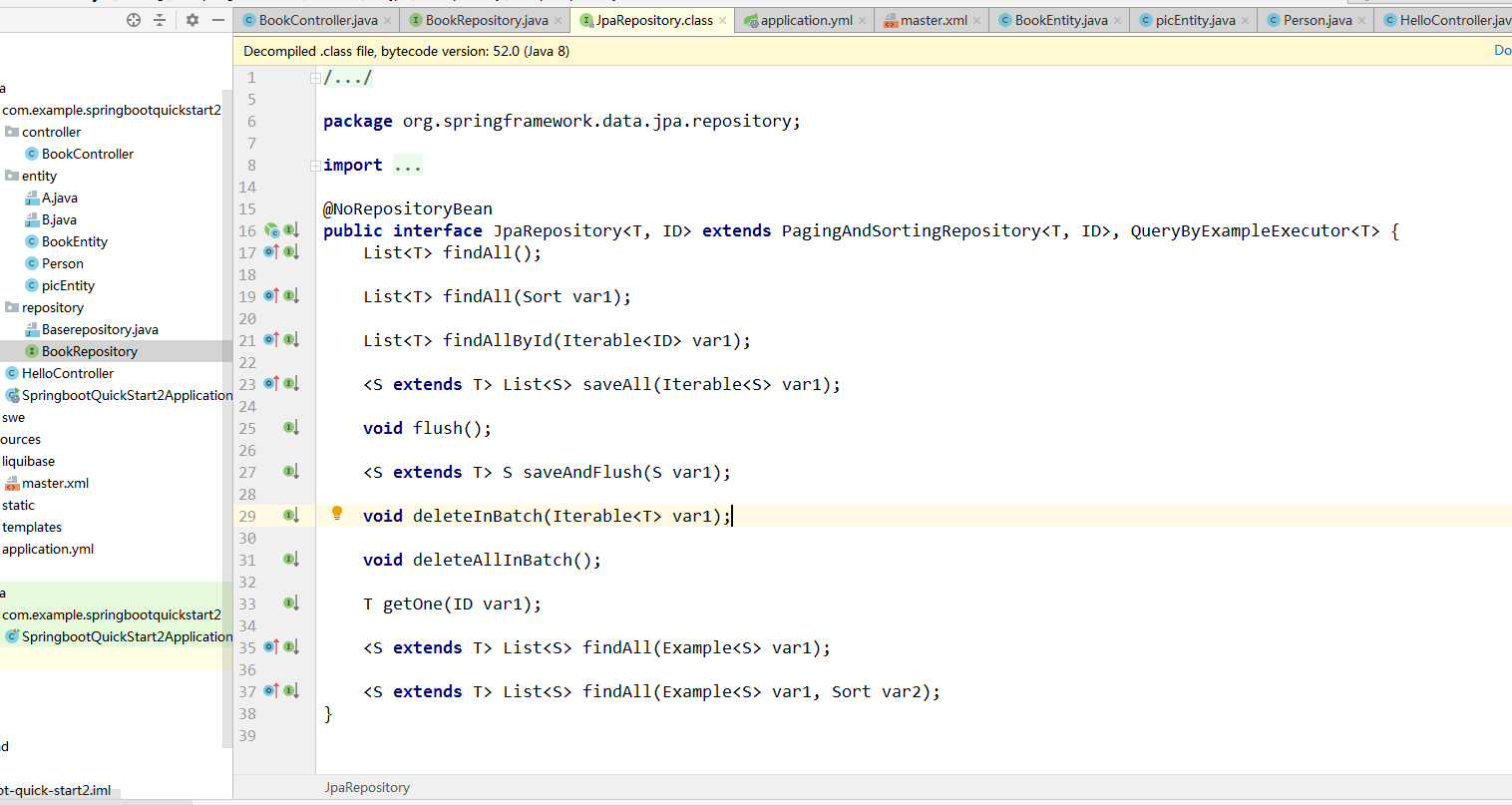
实体类添加字段注释？？？？



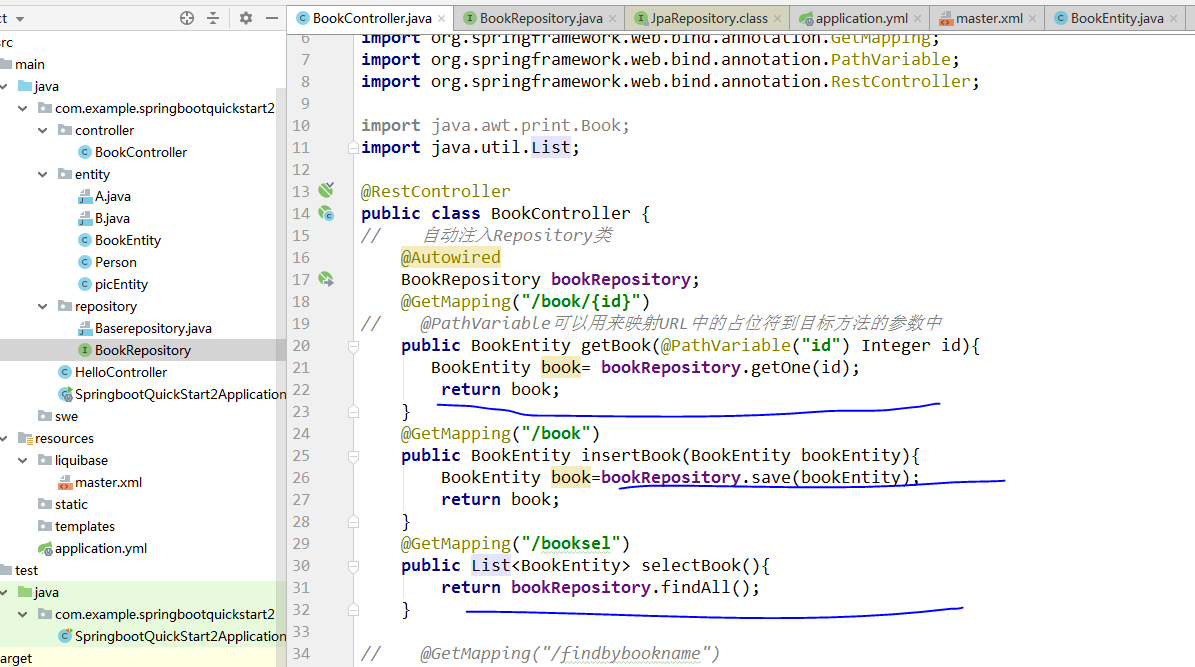
#### 4.4dao层创建repository仓库类



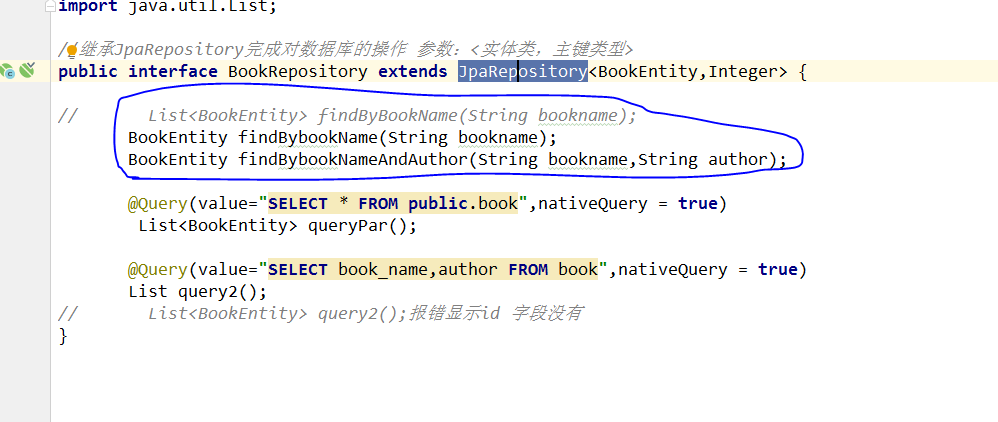
JpaRepository里面封装了一些方法，



##### 我们可以在使用的时候直接调用里面的接口



##### 还可以在repository仓库类中自定义方法



需要注意的是：自定义查询时findBybookName====>字段book\_name===实体属性==》bookName

findByAothor==>字段author

##### Jpa repository自定义方法命名规范

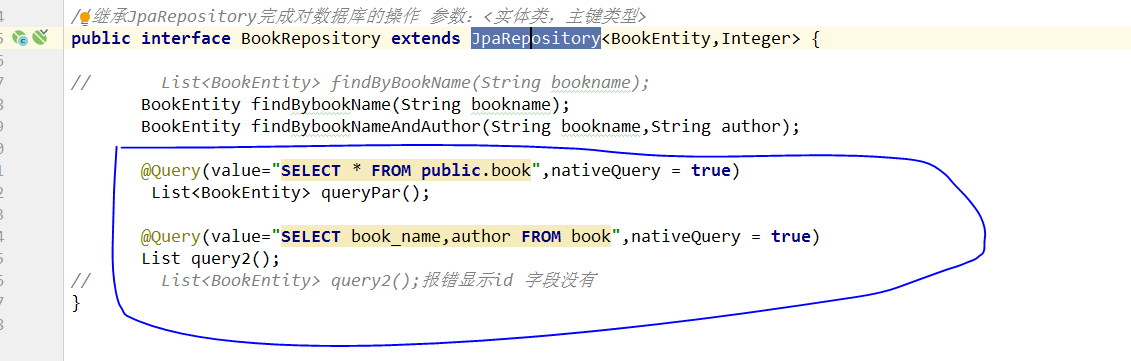
<https://blog.csdn.net/liyang_nash/article/details/80704089>

jpa默认只有save(Entity)方法，如果数据库中没有记录就新增，如果数据库中有记录就更新记录。

Jpa 增删改查使用方法

<https://www.cnblogs.com/lsl1314520/p/10598346.html>

##### [还可以利用@Query进行查询](https://www.cnblogs.com/powerwu/articles/10717970.html)



@Query使用方法

<https://blog.csdn.net/yingxiake/article/details/51016234>

<https://blog.csdn.net/qq_26434357/article/details/95513073>

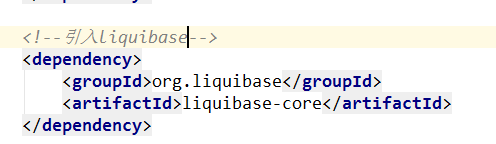


### Liquibase

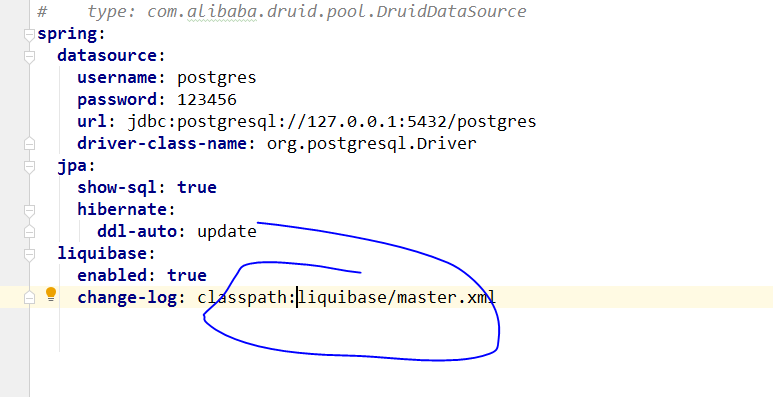
官方文档

<http://www.liquibase.org/documentation/index.html>

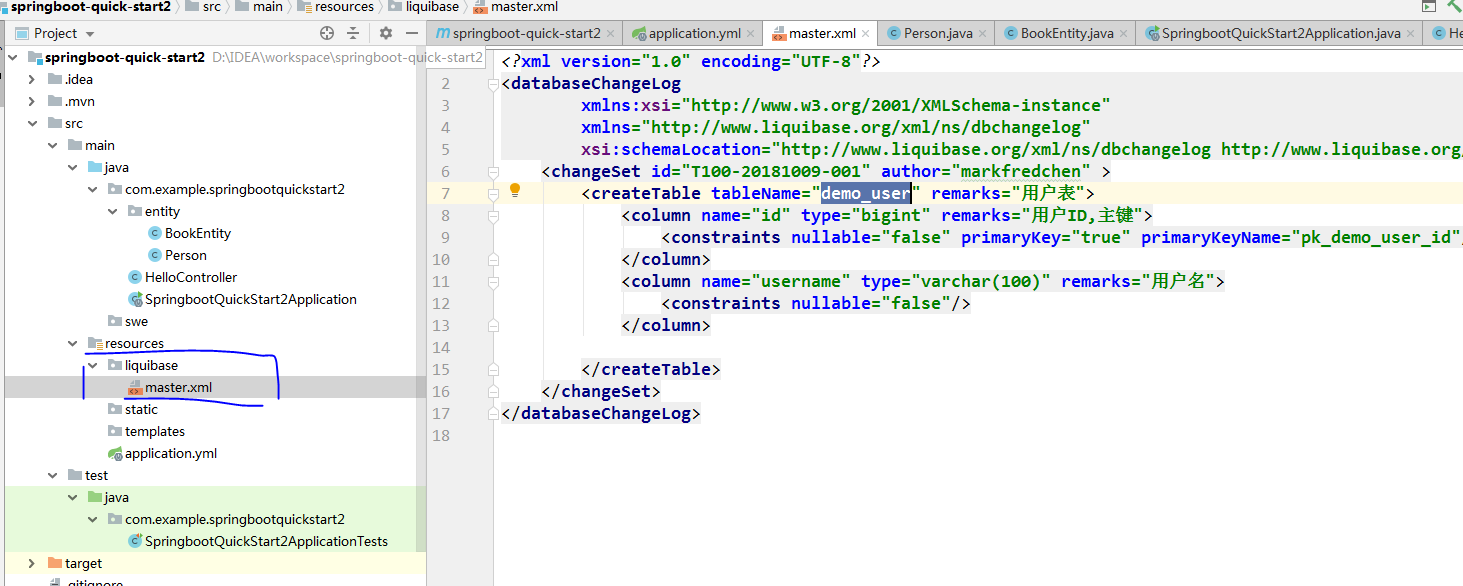
#### 配置：



Yml

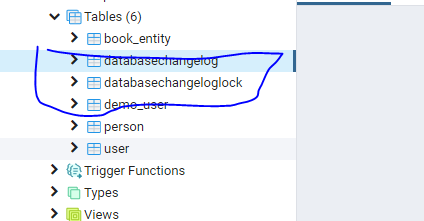


标记部分没有空格



以xml文件生成数据表

运行时自动生成



#### 简介

Liquibase是一个用于数据库重构和迁移的开源工具，类似于php 中使用artisan

自动创建生成数据表，artisan将数据表的变化放在php文件中，liquibase放在XMl文件中。可实现代码的迁移和回滚。

Liquibase是一个用于跟踪、管理和应用数据库变化的开源数据库重构工具。它将所有数据库的变化保存在XML文件中，便于版本控制和项目部署升级。在快速搭建项目的JHipster框架中集成了该工具，使用该工具可以稳定、高效率地实现代码迁移与回滚。

本文使用Java Hipster快速自动生成的项目目录，简要说明liquibase使用方法。XML文件所在目录：resources\config\liquibase。

<https://blog.csdn.net/solocoder/article/details/99696715>

## JDBC与 JdbcTemplate

JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，

可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。

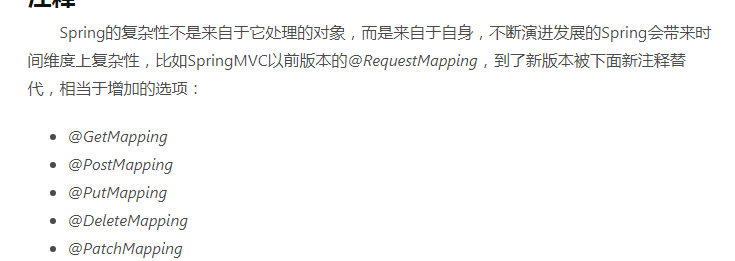
DBCTemplate就是Spring对JDBC的封装,通俗点说就是Spring对jdbc的封装的模板

## 注解

@RequestMapping与@GetMapping和@PostMapping

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

@GetMapping用于处理请求方法的GET类型，@ PostMapping用于处理请求方法的POST类型



## 日志

### 4.1配置

<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring‐boot‐starter</**artifactId**>  
</**dependency**>  
 *<!--引入日志-->*<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-logging</**artifactId**>  
</**dependency**>



### 4.2使用

//日志的级别；

//由低到高 trace<debug<info<warn<error

//可以调整输出的日志级别；日志就只会在这个级别以以后的高级别生效

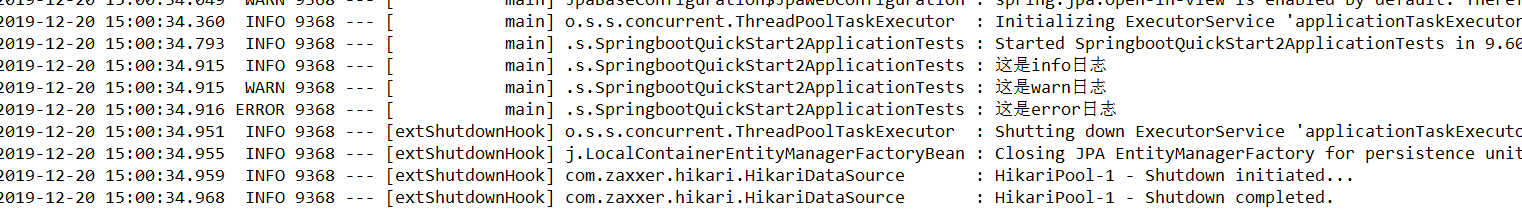
Logger logger= LoggerFactory.*getLogger*(getClass());  
logger.trace(**"这是trace日志"**);  
logger.debug(**"这是debug日志"**);

//SpringBoot默认给我们使用的是info级别的，没有指定级别的就用SpringBoot默认规定的级别；root

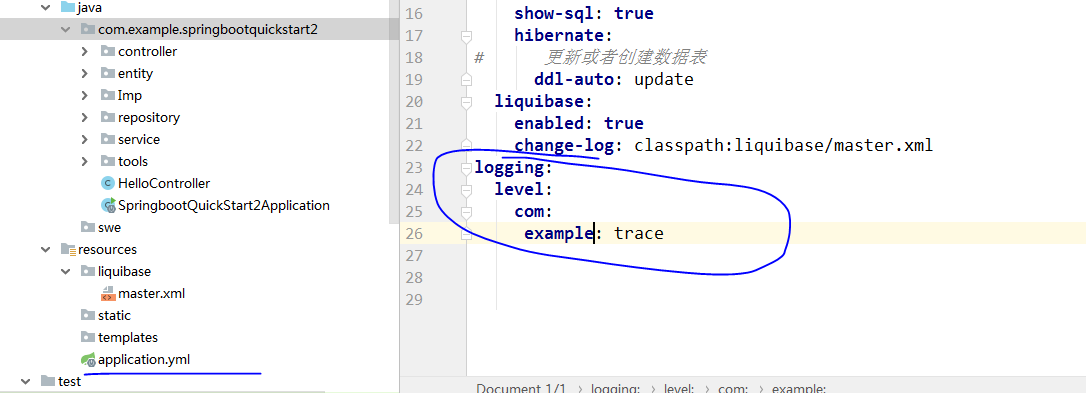
级别

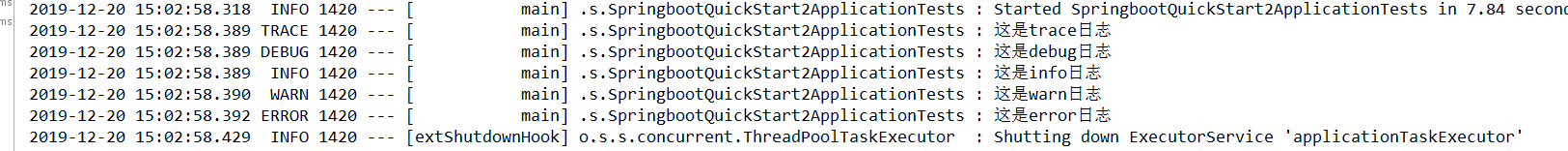
控制台只输出info 及以上日志

logger.info(**"这是info日志"**);  
logger.warn(**"这是warn日志"**);  
logger.error(**"这是error日志"**);



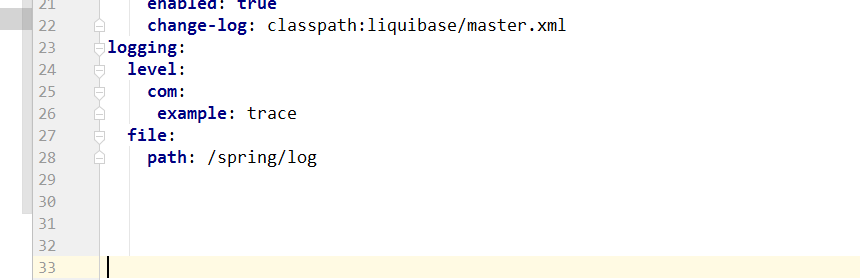
修改日志级别

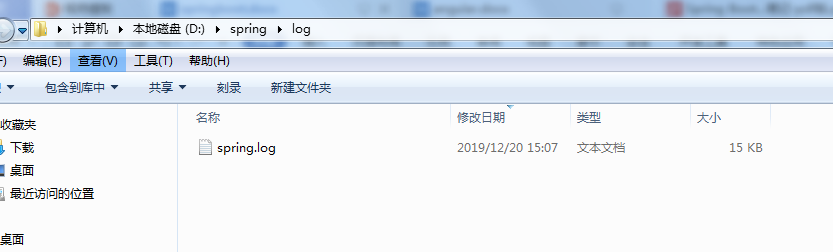




指定日志文件路径

/spring/log 项目根路径下的文件





### 4.3修改日志配置

logging.level.com.atguigu=trace

#logging.path=

# 不指定路径在当前项目下生成springboot.log日志

# 可以指定完整的路径；

#logging.file=G:/springboot.log

# 在当前磁盘的根路径下创建spring文件夹和里面的log文件夹；使用 spring.log 作为默认文件 logging.path=/spring/log

## 五．Docker

### 5.1介绍

Docker让开发者将应用或依赖包打包到一个轻量级，可移植的容器中，然后发布到任何流行的Linux机器上

## Security登录认证

官网：

<https://docs.spring.io/spring-security/site/docs/5.2.2.BUILD-SNAPSHOT/reference/htmlsingle/#prerequisites>

### 6.1引入依赖

*<!--引入spring security-->*<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-security</**artifactId**>  
</**dependency**>

## 6.2编写springsecurity配置类

## 七．Spring boot mvc 的理解

Service

## 八．拦截器

### 8.1创建拦截器



### 8.2注入拦截器



## @注解

### 9.1@Component

把简单的java对象实例化到spring容器中

Pojo(简单的java对象)

### 9.2@Configuration

个人理解就是加上这个注解之后项目启动时自动解析你的配置

### 9.3@Cacheable

<https://www.cnblogs.com/MagicAsa/p/10762356.html>

原理

@Cacheable(cacheNames = {"emp"},keyGenerator = "myKeyGenerator",condition = "#a0>1",unless = "#a0==2")   
意思为 放在的缓存名称为emp中，key的生成方式为配置的myKeyGenerator类，当第一个参数大于1时候缓存，当第一个参数等于2时不缓存

@Cacheable(value={"users"}, key="#user.id", condition="#user.id%2==0")



## sprinboot其他

### 10.1session使用

request.getSession().setAttribute(ConstantUtil.***USER\_SESSION\_KEY***, userEntity.getUserName());

request.getSession().getAttribute(ConstantUtil.***USER\_SESSION\_KEY***)

## 自动配置

## Bean

Bean是Spring框架中****最核心****的两个概念之一（另一个是****面向切面编程AOP****）

官方翻译：

在 Spring 中，构成应用程序****主干****并由****Spring IoC容器****管理的****对象****称为****bean****。bean是一个由Spring IoC容器实例化、组装和管理的对象

概念简单明了，我们提取处关键的信息：

1. bean是对象，一个或者多个不限定
2. bean由Spring中一个叫IoC的东西管理
3. 我们的应用程序由一个个bean构成

理解Spring Boot的自动配置先要理解@Conditional注解

@Bean: 将当前对象放入到Spring的IOC容器中来管理   
@Conditional：满足特定条件时才会创建一个Bean放入到IOC容器，SpringBoot就是利用这个特性进行自动配置的。

## Redis

Redis中文文档

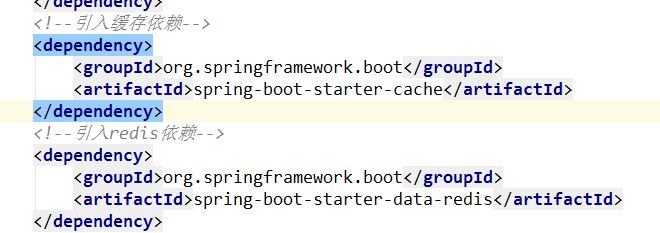
### <http://www.redis.cn/documentation.html>

### 12.1安装配置

windows下载

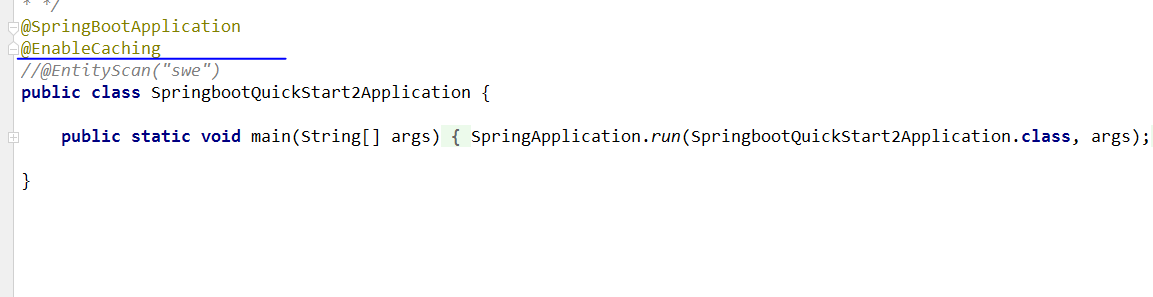
<https://github.com/microsoftarchive/redis/releases>

引入依赖

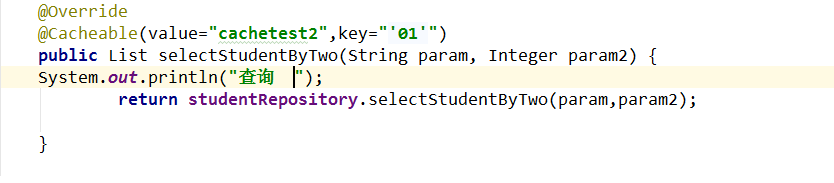




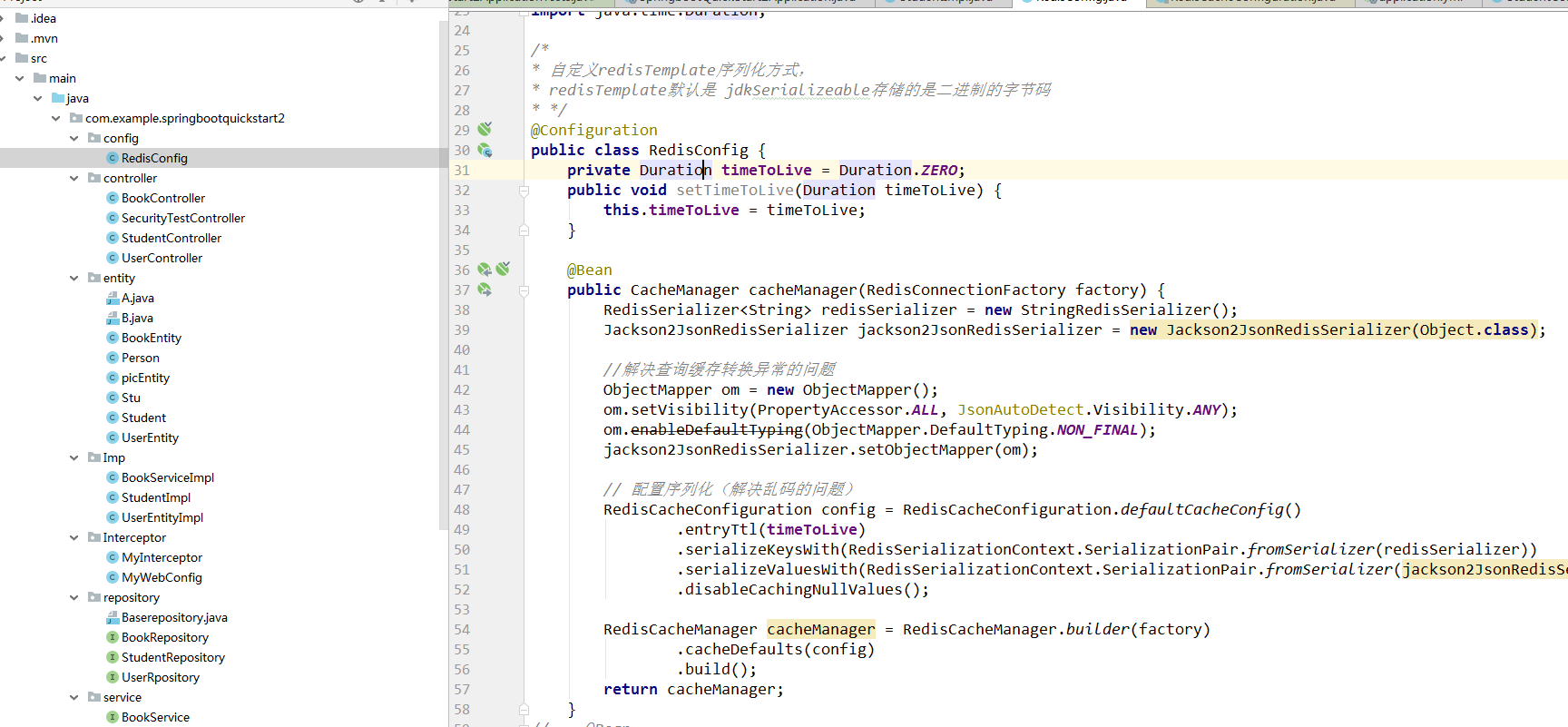
**开启缓存**



**在方法上使用@Cacheable注解**



设置序列化



@Bean  
**public** CacheManager cacheManager(RedisConnectionFactory factory) {  
 RedisSerializer<String> redisSerializer = **new** StringRedisSerializer();  
 Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = **new** Jackson2JsonRedisSerializer(Object.**class**);  
  
 *//解决查询缓存转换异常的问题* ObjectMapper om = **new** ObjectMapper();  
 om.setVisibility(PropertyAccessor.***ALL***, JsonAutoDetect.Visibility.***ANY***);  
 om.enableDefaultTyping(ObjectMapper.DefaultTyping.***NON\_FINAL***);  
 jackson2JsonRedisSerializer.setObjectMapper(om);  
  
 *// 配置序列化（解决乱码的问题）* RedisCacheConfiguration config = RedisCacheConfiguration.*defaultCacheConfig*()  
 .entryTtl(**timeToLive**)  
 .serializeKeysWith(RedisSerializationContext.SerializationPair.*fromSerializer*(redisSerializer))  
 .serializeValuesWith(RedisSerializationContext.SerializationPair.*fromSerializer*(jackson2JsonRedisSerializer))  
 .disableCachingNullValues();  
  
 RedisCacheManager cacheManager = RedisCacheManager.*builder*(factory)  
 .cacheDefaults(config)  
 .build();  
 **return** cacheManager;  
}

### 12.2使用

序列化

<https://www.cnblogs.com/wslook/p/9401134.html>

# Java

Iterator，list，arraylist ，map

aop面向方面编程

Oop面向对象编程

Final关键字

用final修饰一个类，表示该类不能被继承

被final修饰的方法不能被重写

 如果父类中有final修饰的方法，那么子类不能去重写

Final修饰成员变量必须要赋初始值，而且是只能初始化一次

static修饰的属性强调它们只有一个，final修饰的属性表明是一个常数（创建后不能被修改）。static final修饰的属性表示一旦给值，就不可修改，并且可以通过类名访问

## 重载！！！

Java允许同名函数不同参数或者参数顺序不同

## Idea 快捷键使用

*//ctrl+o打开重写方法*