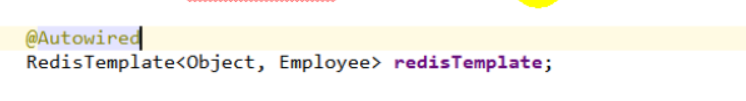
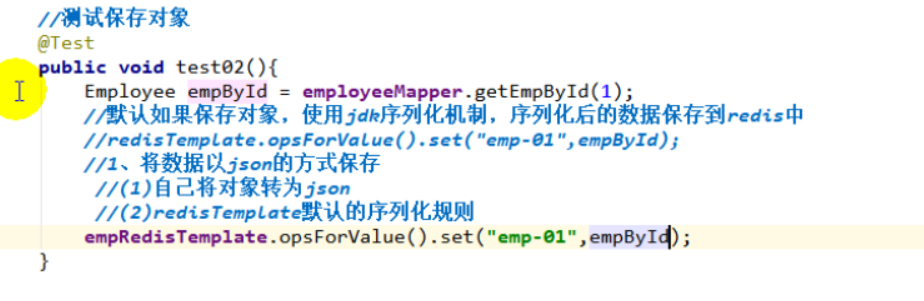
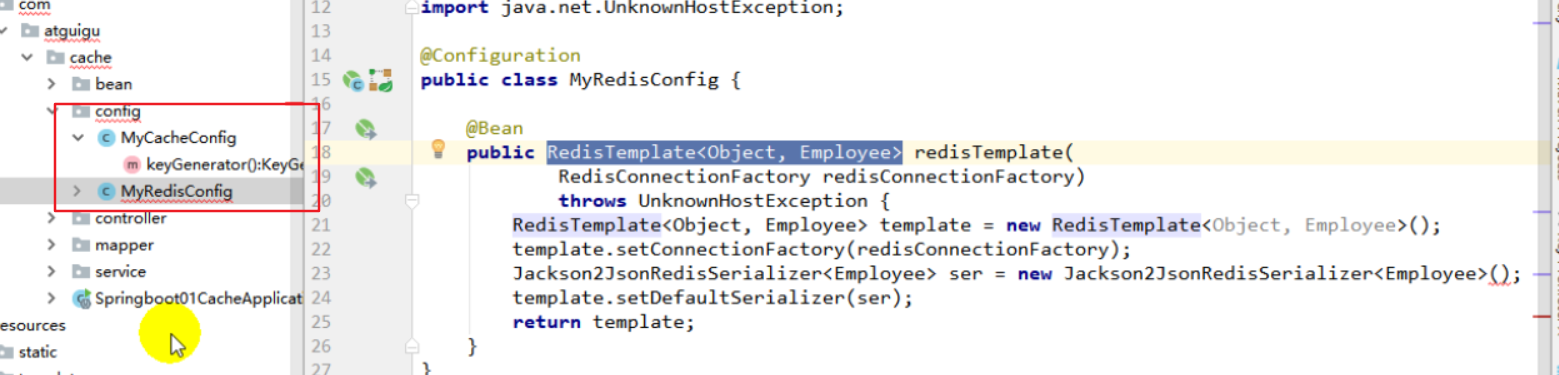


实现自己的序列化：



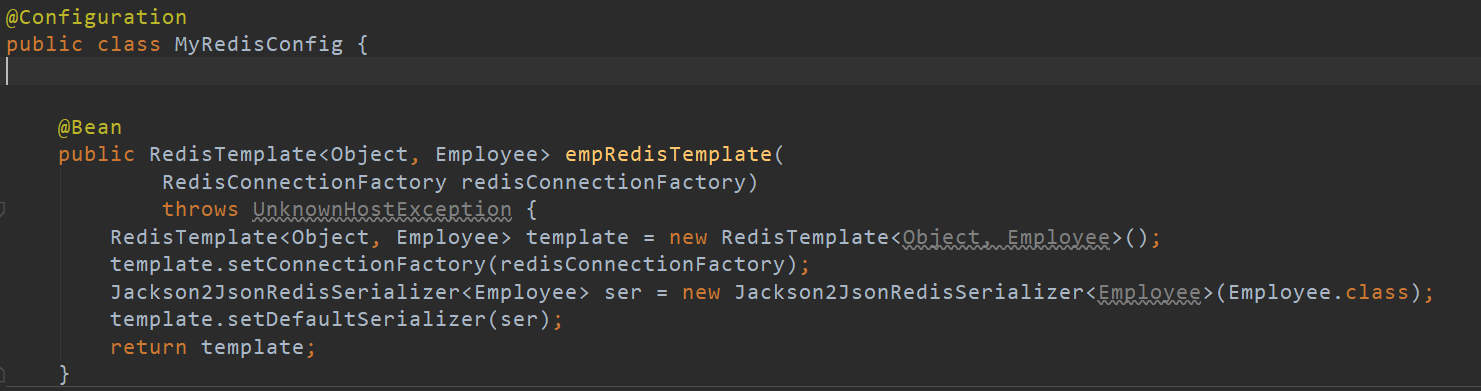


1. 只要在配置类中

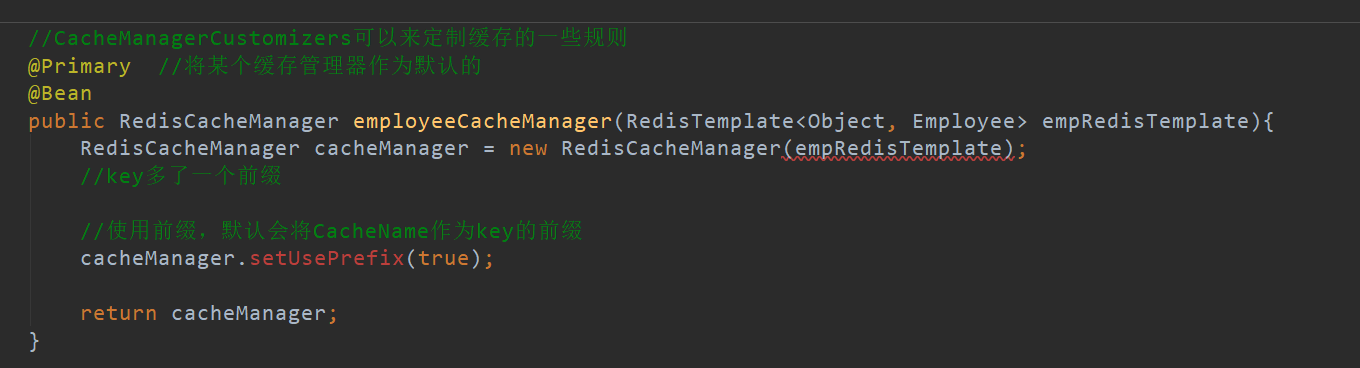


Redis缓存自定义：

1. 在配置类中首先是先序列化目标类，这样往Redis缓存中存入数据才不会是系列化的字符



1. 如果是redis缓存中取出数据则：如果多个管理器，则应该添加一个@Primary 注解，制定一个默认的。反序列化



1. 在service层：

@Qualifier("deptCacheManager")

@Autowired

RedisCacheManager deptCacheManager;

缓存的数据能存入redis；

第二次从缓存中查询就不能反序列化回来；

存的是dept的json数据;CacheManager默认使用RedisTemplate<Object, Employee>操作Redis

注解版

代码： cacheManage: 指定一个缓存管理器

@Cacheable(cacheNames = "dept",cacheManager = "deptCacheManager")

public Department getDeptById(Integer id){

System.out.println("查询部门"+id);

Department department = departmentMapper.getDeptById(id);

return department;

}

// 使用缓存管理器得到缓存，进行api调用

public Department getDeptById(Integer id){

System.out.println("查询部门"+id);

Department department = departmentMapper.getDeptById(id);

//获取某个缓存

Cache dept = deptCacheManager.getCache("dept");

dept.put("dept:1",department);

return department;

}

完整步骤：

/\*\*

\* 一、搭建基本环境

\* 1、导入数据库文件 创建出department和employee表

\* 2、创建javaBean封装数据

\* 3、整合MyBatis操作数据库

\* 1.配置数据源信息

\* 2.使用注解版的MyBatis；

\* 1）、@MapperScan指定需要扫描的mapper接口所在的包

\* 二、快速体验缓存

\* 步骤：

\* 1、开启基于注解的缓存 @EnableCaching

\* 2、标注缓存注解即可

\* @Cacheable

\* @CacheEvict

\* @CachePut

\* 默认使用的是ConcurrentMapCacheManager==ConcurrentMapCache；将数据保存在 ConcurrentMap<Object, Object>中

\* 开发中使用缓存中间件；redis、memcached、ehcache；

\* 三、整合redis作为缓存

\* Redis 是一个开源（BSD许可）的，内存中的数据结构存储系统，它可以用作数据库、缓存和消息中间件。

\* 1、安装redis：使用docker；

\* 2、引入redis的starter

\* 3、配置redis

\* 4、测试缓存

\* 原理：CacheManager===Cache 缓存组件来实际给缓存中存取数据

\* 1）、引入redis的starter，容器中保存的是 RedisCacheManager；

\* 2）、RedisCacheManager 帮我们创建 RedisCache 来作为缓存组件；RedisCache通过操作redis缓存数据的

\* 3）、默认保存数据 k-v 都是Object；利用序列化保存；如何保存为json

\* 1、引入了redis的starter，cacheManager变为 RedisCacheManager；

\* 2、默认创建的 RedisCacheManager 操作redis的时候使用的是 RedisTemplate<Object, Object>

\* 3、RedisTemplate<Object, Object> 是 默认使用jdk的序列化机制

\* 4）、自定义CacheManager；

\*

\*/