目录

[1、 redis环境准备 1](#_Toc521435484)

[(1)下载安装msi包 1](#_Toc521435485)

[(2)安装教程&&注意事项 1](#_Toc521435486)

[（3）下载redis-desktop-manager图形化客户端 3](#_Toc521435487)

[(4)docker 运行redis容器其实更简单 3](#_Toc521435488)

[2、了解redis自身的知识体系，基本的key、map、list等的操作命令 3](#_Toc521435489)

[3、 springboot 集成redis（使用RedisTemplate） 3](#_Toc521435490)

[4、 springboot 集成redis（使用注解） 4](#_Toc521435491)

[（1） 引用依赖 4](#_Toc521435492)

[（2）配置application.properties，包含如下信息 4](#_Toc521435493)

[（3）创建User实体类，一定要实现Serializable序列化 4](#_Toc521435494)

[（4）创建个repository 6](#_Toc521435495)

[（5）加个service和 impl 7](#_Toc521435496)

[（6）简单测试一下 8](#_Toc521435497)

[5、 springboot + redis (使用jedis) 9](#_Toc521435498)

1. redis环境准备

## (1)下载安装msi包

Redis window版下载地址

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

下载msi

## (2)安装教程&&注意事项

默认端口：6379

win10 redis教程：

<https://blog.csdn.net/u012754211/article/details/72773166>

如果出现下图：

|  |
| --- |
| C:\Users\zj\Desktop\2345截图20180805235658.png |

证明成功安装

因为msi安装包装完默认自动将redis-server作为windows服务自动启动。我已经把它停止了

以后每次需输入开启命令：

redis-server.exe redis.windows.conf

如果要关闭的话：

* 关闭 redis  redis-server --service-stop

出现下图，表明开启成功

|  |
| --- |
| C:\Users\zj\Desktop\2345截图20180806001508.png |

然后启动 redis 客户端

redis-cli.exe -h 127.0.0.1 -p 6379

|  |
| --- |
| C:\Users\zj\Desktop\2345截图20180805235658.png |

这样就可以执行redis服务命令

## （3）下载redis-desktop-manager图形化客户端

<https://www.oschina.net/news/78736/redis-desktop-manager-0-9-0-alpha3>

## (4)docker 运行redis容器其实更简单

# 2、了解redis自身的知识体系，基本的key、map、list等的操作命令

Redis 教程| 菜鸟教程

<http://www.runoob.com/redis/redis-commands.html>

1. springboot 集成redis（使用RedisTemplate）

Spring Boot整合JPA+MySQL+Redis - CSDN博客

<https://blog.csdn.net/icecaptain/article/details/79200388>

SpringBoot整合Redis及Redis工具类撰写 - zeng1994 - 博客园

<https://www.cnblogs.com/zeng1994/p/03303c805731afc9aa9c60dbbd32a323.html>

1. springboot 集成redis（使用注解）
2. 引用依赖

|  |
| --- |
| <!--Redis依赖--> <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-redis</artifactId> </dependency> |

（2）配置application.properties，包含如下信息

|  |
| --- |
| # REDIS (RedisProperties)redis 配置信息 # Redis数据库索引（默认为0） spring.redis.database=0 # Redis服务器地址 spring.redis.host=127.0.0.1 # Redis服务器连接端口 spring.redis.port=6379 # Redis服务器连接密码（默认为空） spring.redis.password= # 连接池最大连接数（使用负值表示没有限制） spring.redis.pool.max-active=8 # 连接池最大阻塞等待时间（使用负值表示没有限制） spring.redis.pool.max-wait=-1 # 连接池中的最大空闲连接 spring.redis.pool.max-idle=8 # 连接池中的最小空闲连接 spring.redis.pool.min-idle=0 # 连接超时时间（毫秒） spring.redis.timeout=1000 |

## （3）创建User实体类，一定要实现Serializable序列化

|  |
| --- |
| package jit.hf.agriculture.domain;  import javax.persistence.Entity; import javax.persistence.GeneratedValue; import javax.persistence.Id; import java.io.Serializable;  */\*\*  \** ***@Auther:*** *zj  \** ***@Date:*** *2018/8/6 17:10  \** ***@Description:用于redis*** *的测试用户集  \*/* @Entity public class TestUser implements Serializable{   //private static final long serialVersionUID = 1L;   @Id  @GeneratedValue  private Integer id;   private String name;   private String address;    public Integer getId() {  return id;  }   public void setId(Integer id) {  this.id = id;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public String getAddress() {  return address;  }   public void setAddress(String address) {  this.address = address;  } } |

## （4）创建个repository

|  |
| --- |
| package jit.hf.agriculture.Repository;  import jit.hf.agriculture.domain.TestUser; import org.springframework.cache.annotation.CacheConfig; import org.springframework.cache.annotation.CacheEvict; import org.springframework.cache.annotation.CachePut; import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.cache.annotation.Cacheable; import org.springframework.data.jpa.repository.Modifying; import org.springframework.transaction.annotation.Transactional;   */\*\*  \** ***@Auther:*** *zj  \** ***@Date:*** *2018/8/6 19:54  \** ***@Description:自定义方法，将mysql与redis操作结合起来（mysql通过jpa,redis通过注解）*** *\*/* @CacheConfig(cacheNames = "TestUser") public interface TestUserRepository extends JpaRepository<TestUser, Integer> {   @Cacheable(key = "#p0")// #p0代表第一个参数，也就是id。这句话加上之后，当你在调用findById时，  // 就会先从redis的TestUser(@CacheConfig(cacheNames = "TestUser")通过这个注解指定)缓存对象里去查询key等于传过来的id的值。  // 如果没有，就去查表。  TestUser findById(int id);   */\*\*  \* 新增或修改时  \*/* @CachePut(key = "#p0.id")//在save方法上加了个CachePut，代表往缓存里添加值，key为参数post的id属性，  // 这样当我们save一个Post对象时，redis就会新增一个以id为key的Post对象；  // 如果是update操作，那同样，redis会覆盖id相同的Post对象的值，也完成一次更新。  @Override  TestUser save(TestUser testUser);   @Transactional  @Modifying  @CacheEvict(key = "#p0")//该注解的作用是：同时删除mysql和redis的数据；不然mysql删掉了但redis不会删除  int deleteById(int id);   } |

## （5）加个service和 impl

|  |
| --- |
| package jit.hf.agriculture.Service;  import jit.hf.agriculture.domain.TestUser;  */\*\*  \** ***@Auther:*** *zj  \** ***@Date:*** *2018/8/7 13:55  \** ***@Description:*** *\*/* public interface TestUserService {   TestUser findById(int id);   TestUser save(TestUser testUser);   int delete(int id);  } |
| package jit.hf.agriculture.Service;  import jit.hf.agriculture.Repository.TestUserRepository; import jit.hf.agriculture.domain.TestUser; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;  */\*\*  \** ***@Auther:*** *zj  \** ***@Date:*** *2018/8/7 13:57  \** ***@Description:*** *\*/* @Service public class TestUserServiceImpl implements TestUserService{   @Autowired  private TestUserRepository testUserRepository;   @Override  public TestUser findById(int id) {  return testUserRepository.findById( id );  }   @Override  public TestUser save(TestUser testUser) {  return testUserRepository.save( testUser );  }   @Override  public int delete(int id) {  return testUserRepository.deleteById( id );  } } |

## （6）简单测试一下

启动后访问 http://localhost:8080/save?content=1&weight=1

这样就添加了一条记录，控制台有一个insert语句。  


然后访问查询，http://localhost:8080/query/1

会发现查询到了id为1的这条记录，并且控制台没有走select查询语句，也就是根本没访问数据库，直接从redis缓存拿的值。

下面做一个更新操作，看看是否会同步到redis里。http://localhost:8080/save?content=1&weight=2&id=1

把weight改为2，访问地址看看结果。

控制台打印了两条语句

Hibernate: select post0\_.id as id1\_0\_0\_, post0\_.content as content2\_0\_0\_, post0\_.weight as weight3\_0\_0\_ from post post0\_ where post0\_.id=?  
Hibernate: update post set content=?, weight=? where id=?

说明数据已经被更新了。然后再查询http://localhost:8080/query/1

发现查到的数据已经改变，并且控制台没有走select语句，说明在update时，redis已经更新了。

下面做删除操作，可以直接在数据库里删这条记录，或者通过浏览器访问来删除。http://localhost:8080/delete/1

控制台走了删除delete语句。再访问查询地址。发现不能查到这条记录，也就是db的删除成功了，但redis也删除。

访问http://localhost:8080/delete/1

然后再查询就会发现id为1的值已经查不到了。

这样我们就完成了一个最简单的整合redis的demo。包括了对单个对象的增删改查的缓存。

参考：

1 Springboot中使用redis，自动缓存、更新、删除 - CSDN博客

<https://blog.csdn.net/tianyaleixiaowu/article/details/70314277>

2 Springboot中使用redis，配置redis的key value生成策略 - CSDN博客

<https://blog.csdn.net/tianyaleixiaowu/article/details/70595073>

3 Springboot中使用redis，redis自动缓存异常处理 - CSDN博客

<https://blog.csdn.net/tianyaleixiaowu/article/details/70670329>

1. springboot + redis (使用jedis)

<https://segmentfault.com/a/1190000009165070#articleHeader9>

<https://blog.csdn.net/xiao______xin/article/details/73896165>

<https://blog.csdn.net/catoop/article/details/71275331>

参考：

Spring Boot整合JPA+MySQL+Redis - CSDN博客

<https://blog.csdn.net/icecaptain/article/details/79200388>

Redis 教程| 菜鸟教程

<http://www.runoob.com/redis/redis-commands.html>

Springboot整合redis - 简书

<https://www.jianshu.com/p/7f6a180e6ee7>

SpringBoot整合Redis - 掘金

<https://juejin.im/post/5ad6acb4f265da239e4e9906>

springboot(三)：Spring boot中Redis的使用 - 纯洁的微笑博客

<http://www.ityouknow.com/springboot/2016/03/06/spring-boot-redis.html>

使用 SpringBoot 之 JPA 整合 Redis 实现缓存 - CSDN博客

<https://blog.csdn.net/larger5/article/details/79696562>

Spring Boot中使用Redis数据库 | 程序猿DD

<http://blog.didispace.com/springbootredis/>

Spring Boot---(6)SpringBoot整合Redis - CSDN博客

<https://blog.csdn.net/weixin_39800144/article/details/79113202>

<https://blog.csdn.net/catoop/article/details/71275331>