**一、demo**

svn://svn.club/myJavaEE/javaWeb/yjfly-study/spring-boot-web

<https://github.com/ityouknow/spring-boot-examples/tree/master/spring-boot-web>

参考：<https://www.cnblogs.com/haitao-xie/p/6323423.html>

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

spring boot对常用的数据库支持外，对nosql 数据库也进行了封装自动化。

**redis介绍**

Redis是目前业界使用最广泛的内存数据存储。相比memcached，Redis支持更丰富的数据结构，例如hashes, lists, sets等，同时支持数据持久化。除此之外，Redis还提供一些类数据库的特性，比如事务，HA，主从库。可以说Redis兼具了缓存系统和数据库的一些特性，因此有着丰富的应用场景。本文介绍Redis在Spring Boot中两个典型的应用场景。

**如何使用**

1、引入 spring-boot-starter-redis

<**dependency**>

<**groupId**>org.springframework.boot</**groupId**>

<**artifactId**>spring-boot-starter-data-redis</**artifactId**>

</**dependency**>

2、添加配置文件

*# REDIS (RedisProperties)*

*# Redis数据库索引（默认为0）*

spring.redis.database=0

*# Redis服务器地址*

spring.redis.host=192.168.0.58

*# 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=0

3、添加cache的配置类

**@Configuration**

**@EnableCaching**

**public** **class** **RedisConfig** **extends** CachingConfigurerSupport**{**

         **@Bean**

         **public** KeyGenerator **keyGenerator()** **{**

        **return** **new** **KeyGenerator()** **{**

            **@Override**

            **public** Object **generate(**Object target**,** Method method**,** Object**...** params**)** **{**

                StringBuilder sb **=** **new** StringBuilder**();**

                sb**.**append**(**target**.**getClass**().**getName**());**

                sb**.**append**(**method**.**getName**());**

                **for** **(**Object obj **:** params**)** **{**

                    sb**.**append**(**obj**.**toString**());**

                **}**

                **return** sb**.**toString**();**

            **}**

        **};**

    **}**

    **@SuppressWarnings(**"rawtypes"**)**

    **@Bean**

    **public** CacheManager **cacheManager(**RedisTemplate redisTemplate**)** **{**

        RedisCacheManager rcm **=** **new** RedisCacheManager**(**redisTemplate**);**

        *//设置缓存过期时间*

        *//rcm.setDefaultExpiration(60);//秒*

        **return** rcm**;**

    **}**

    **@Bean**

    **public** RedisTemplate**<**String**,** String**>** **redisTemplate(**RedisConnectionFactory factory**)** **{**

        StringRedisTemplate template **=** **new** StringRedisTemplate**(**factory**);**

        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer **=** **new** Jackson2JsonRedisSerializer**(**Object**.**class**);**

        ObjectMapper om **=** **new** ObjectMapper**();**

        om**.**setVisibility**(**PropertyAccessor**.**ALL**,** JsonAutoDetect**.**Visibility**.**ANY**);**

        om**.**enableDefaultTyping**(**ObjectMapper**.**DefaultTyping**.**NON\_FINAL**);**

        jackson2JsonRedisSerializer**.**setObjectMapper**(**om**);**

        template**.**setValueSerializer**(**jackson2JsonRedisSerializer**);**

        template**.**afterPropertiesSet**();**

        **return** template**;**

    **}**

**}**

3、好了，接下来就可以直接使用了

**@RunWith(**SpringJUnit4ClassRunner**.**class**)**

**@SpringApplicationConfiguration(**Application**.**class**)**

**public** **class** **TestRedis** **{**

    **@Autowired**

    **private** StringRedisTemplate stringRedisTemplate**;**

         **@Autowired**

         **private** RedisTemplate redisTemplate**;**

    **@Test**

    **public** **void** **test()** **throws** Exception **{**

        stringRedisTemplate**.**opsForValue**().**set**(**"aaa"**,** "111"**);**

        Assert**.**assertEquals**(**"111"**,** stringRedisTemplate**.**opsForValue**().**get**(**"aaa"**));**

    **}**

    **@Test**

    **public** **void** **testObj()** **throws** Exception **{**

        User user**=new** User**(**"aa@126.com"**,** "aa"**,** "aa123456"**,** "aa"**,**"123"**);**

        ValueOperations**<**String**,** User**>** operations**=**redisTemplate**.**opsForValue**();**

        operations**.**set**(**"com.neox"**,** user**);**

        operations**.**set**(**"com.neo.f"**,** user**,**1**,**TimeUnit**.**SECONDS**);**

        Thread**.**sleep**(**1000**);**

        *//redisTemplate.delete("com.neo.f");*

        **boolean** exists**=**redisTemplate**.**hasKey**(**"com.neo.f"**);**

        **if(**exists**){**

          System**.**out**.**println**(**"exists is true"**);**

        **}else{**

          System**.**out**.**println**(**"exists is false"**);**

        **}**

       *// Assert.assertEquals("aa", operations.get("com.neo.f").getUserName());*

    **}**

**}**

以上都是手动使用的方式，如何在查找数据库的时候自动使用缓存呢，看下面；

4、自动根据方法生成缓存

**@RequestMapping(**"/getUser"**)**

**@Cacheable(**value**=**"user-key"**)**

**public** User **getUser()** **{**

    User user**=**userRepository**.**findByUserName**(**"aa"**);**

    System**.**out**.**println**(**"若下面没出现“无缓存的时候调用”字样且能打印出数据表示测试成功"**);**

    **return** user**;**

**}**

其中value的值就是缓存到redis中的key

**共享Session-spring-session-data-redis**

分布式系统中，sessiong共享有很多的解决方案，其中托管到缓存中应该是最常用的方案之一，

**Spring Session官方说明**

Spring Session provides an API and implementations for managing a user’s session information.

**如何使用**

1、引入依赖

<dependency>

    <groupId>org.springframework.session</groupId>

    <artifactId>spring-session-data-redis</artifactId>

</dependency>

2、Session配置：

**@Configuration**

**@EnableRedisHttpSession(**maxInactiveIntervalInSeconds **=** 86400**\***30**)**

**public** **class** **SessionConfig** **{**

**}**

maxInactiveIntervalInSeconds: 设置Session失效时间，使用Redis Session之后，原Boot的server.session.timeout属性不再生效

好了，这样就配置好了，我们来测试一下

3、测试

添加测试方法获取sessionid

**@RequestMapping(**"/uid"**)**

    String **uid(**HttpSession session**)** **{**

        UUID uid **=** **(**UUID**)** session**.**getAttribute**(**"uid"**);**

        **if** **(**uid **==** **null)** **{**

            uid **=** UUID**.**randomUUID**();**

        **}**

        session**.**setAttribute**(**"uid"**,** uid**);**

        **return** session**.**getId**();**

    **}**

登录redis 输入 keys '\*sessions\*'

t<spring:session:sessions:db031986-8ecc-48d6-b471-b137a3ed6bc4

t(spring:session:expirations:1472976480000

其中 1472976480000为失效时间，意思是这个时间后session失效，db031986-8ecc-48d6-b471-b137a3ed6bc4 为sessionId,登录http://localhost:8080/uid 发现会一致，就说明session 已经在redis里面进行有效的管理了。

**如何在两台或者多台中共享session**

其实就是按照上面的步骤在另一个项目中再次配置一次，启动后自动就进行了session共享。

[**示例代码-github**](https://github.com/ityouknow/spring-boot-examples)

[**示例代码-码云**](https://gitee.com/ityouknow/spring-boot-examples)

**参考**

[Redis的两个典型应用场景](http://emacoo.cn/blog/spring-redis)

[SpringBoot应用之分布式会话](https://segmentfault.com/a/1190000004358410)

**作者：纯洁的微笑**

**出处：**[**www.ityouknow.com**](http://www.ityouknow.com/)

**版权所有，欢迎保留原文链接进行转载：)**