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。

## 1）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。本文介绍Redis在Spring Boot中两个典型的应用场景。

## 2）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>

1. 添加配置文件：



1. 添加cache的配置类：

@Configuration  
@EnableCaching  
**public class** RedisConfig **extends** CachingConfigurerSupport {  
  
 */\*\*  
 \* redis key生成策略  
 \* 注：该方法只是声明了key的生成策略，还未被使用，需在@Cacheable注解中指定keyGenerator  
 \*  
 \** ***@param:*** *[]  
 \** ***@return:*** *org.springframework.cache.interceptor.KeyGenerator  
 \*/* @Bean  
 **public** KeyGenerator cacheKeyGenerator() {  
 **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();  
 }  
 };  
 }  
  
 */\*\*  
 \* redis全局默认配置  
 \*  
 \** ***@param:*** *[redisTemplate]  
 \** ***@return:*** *org.springframework.cache.CacheManager  
 \*/* @Bean  
 **public** CacheManager cacheManager(RedisConnectionFactory factory) {  
 *//创建缓存管理器对象* RedisCacheManager rcm = RedisCacheManager.*create*(factory);  
 *//设置缓存过期时间  
 //rcm.setDefaultExpiration(60);* **return** rcm;  
 }  
  
 */\*\*  
 \* RedisTemplate相关配置  
 \*  
 \** ***@param:*** *[factory]  
 \** ***@return:*** *org.springframework.data.redis.core.RedisTemplate*<*java.lang.String*,*java.lang.Object*>  
 *\*/* @Bean  
 **public** RedisTemplate<String, Object> redisTemplate() {  
 *//设置序列化* 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);  
  
 *//配置redisTemplate* RedisTemplate<String,Object> redisTemplate = **new** RedisTemplate<>();  
 redisTemplate.setValueSerializer(jackson2JsonRedisSerializer);  
 redisTemplate.afterPropertiesSet();  
  
 **return** redisTemplate;  
 }  
}

1. 测试：

@RunWith(SpringJUnit4ClassRunner.**class**)  
@SpringBootTest  
**public class** TestRedis {  
  
 @Autowired  
 **private** StringRedisTemplate **stringRedisTemplate**;  
  
 @Autowired  
 **private** RedisTemplate **redisTemplate**;  
  
 @Test  
 **public void** test() {  
 **stringRedisTemplate**.opsForValue().set(**"aaa"**, **"111"**);  
 Assert.*assertEquals*(**"111"**, **stringRedisTemplate**.opsForValue().get(**"aaa"**));  
 }  
  
 @Test  
 **public void** testObj() **throws** InterruptedException {  
 Student student = **new** Student(**"aa"**, **"aa123456"**, **"aa@126.com"**, **"aa"**, **"123"**);  
 ValueOperations<String, Student> operations = **redisTemplate**.opsForValue();  
 operations.set(**"com.wcy"**, student);  
 operations.set(**"com.wcy.f"**, student, 1, TimeUnit.***SECONDS***);  
 Thread.*sleep*(1000);  
 *//redisTemplate.delete("com.neo.f");* **boolean** exists = **redisTemplate**.hasKey(**"com.wcy.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。

## 3）自动根据方法生成缓存

@Cacheable(value = "redis", keyGenerator = "cacheKeyGenerator")

public String getRedisString(String param1, String param2) {

return param1+":"+param2;

}

@Cacheable注解生成的类型在redis中默认都是string，在每次请求的时候，都是先根据key到redis查询是否存在，如不存在则执行方法中的代码。并且将方法的返回值和指定的key关联起来，放入到缓存中。

value属性用于指定缓存存储的集合名。

keyGenerator：用于指定key生成器。

## 4）共享Session-spring-session-data-redis

分布式系统中，session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.（Spring Session提供了用于管理用户会话信息的API和实现。）

1. 引入依赖

<dependency>

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

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

</dependency>

1. 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。

1. 测试

添加测试方法获取sessionid

@Controller  
**public class** SessionController {  
   
 @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 ‘\*sessiom\*’

t<spring:session:sessions:db031986-8ecc-48d6-b471-b137a3ed6bc4t(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。

在两台或多台中共享session：

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