链滴

spring boot 2.1.3 使用 mybatis redis cache 缓存

作者：gongxiongzhuang

原文链接：https://ld246.com/article/1553772075691

来源网站：链滴

为了减少数据库访问频率，提高查询速度，今天讲一下spring boot mybatis 怎么集成 redis缓存。

第一步：添加redis相关jar包依赖

这里我只贴出集成redis缓存所需要的jar包，至于链接数据库，和基本的项目启动请参考

Spring boot 2 mybatis配置

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

至于jar包spring-boot-starter-cache至少在版本2.1.3可以不用引用，2.1.3版本jar包spring-boot-starter-data-redis已经自带cache所需要的jar包，不用重复引用。

第二步：添加redis链接数据库配置文件

```
spring:
  # 数据库连接
  #datasource:
  #  password: 111111
  #  url: jdbc:mysql://120.77.241.43:3306/test?useSSL=false&useUnicode=true&characterEncoding=UTF-8
  #  username: root

  # redis
  redis:
    host: 127.0.0.1
    port: 6300
    password: 123
    # Redis数据库索引（默认为0）
    database: 0
    #连接超时时间（毫秒）
    timeout: 10000

    jedis: #一些常规配置
      pool:
        # 连接池中的最大空闲连接
        max-idle: 60
        # 连接池中的最小空闲连接
        min-idle: 30
        # 连接池最大阻塞等待时间（使用负值表示没有限制）
        max-wait: 60000
        # 连接池最大连接数（使用负值表示没有限制）
        max-active: 200
```

第三步：添加RedisConfig配置类

```
@Configuration
```

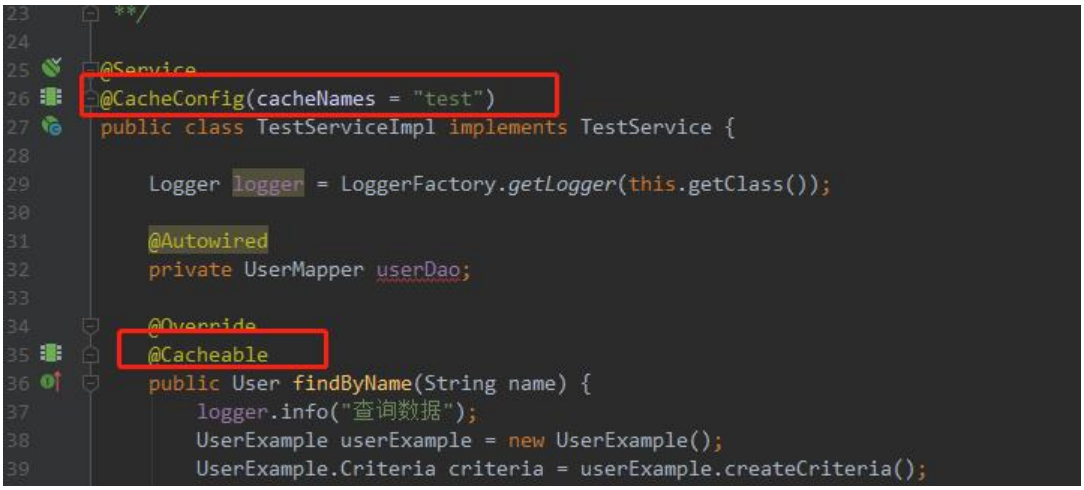
```
@EnableCaching
public class RedisConfig extends CachingConfigurerSupport {

    @Bean(name = "cacheKeyGenerator")
    @Primary//@Primary：自动装配时当出现多个Bean候选者时，被注解为@Primary的Bean将作首选者，否则将抛出异常
    public KeyGenerator keyGenerator() {
        return (target, method, params) -> CacheHashCode.of(params);
    }

    /**
     * spring boot 缓存默认配置
     * @param factory
     * @return
     */
    @Bean
    public CacheManager cacheManager(RedisConnectionFactory factory) {
        return RedisCacheManager.builder(factory)
                //默认缓存时间
                .cacheDefaults(getRedisCacheConfigurationWithTtl(60))
                .transactionAware()
                //自定义缓存时间
                .withInitialCacheConfigurations(getRedisCacheConfigurationMap())
                .build();
    }

    /**
     * 自定义缓存时间
     * @return
     */
    private Map<String, RedisCacheConfiguration> getRedisCacheConfigurationMap() {
        Map<String, RedisCacheConfiguration> redisCacheConfigurationMap = new HashMap>();
        redisCacheConfigurationMap.put("test", this.getRedisCacheConfigurationWithTtl(3000));
        return redisCacheConfigurationMap;
    }

    private RedisCacheConfiguration getRedisCacheConfigurationWithTtl(Integer seconds) {
        return RedisCacheConfiguration.defaultCacheConfig()
                .entryTtl(Duration.ofSeconds(seconds))//定义默认的cache time-to-live.(缓存存储有效间)
                .disableCachingNullValues()//静止缓存为空
                //此处定义了cache key的前缀, 避免公司不同项目之间的key名称冲突.
                .computePrefixWith(cacheName -> "api".concat(":").concat(cacheName).concat(":"))
                //定义key和value的序列化协议, 同时的hash key和hash value也被定义.
                .serializeKeysWith(RedisSerializationContext.SerializationPair.fromSerializer(new StrigRedisSerializer()))
                .serializeValuesWith(RedisSerializationContext.SerializationPair.fromSerializer(createackson2JsonRedisSerializer()))
                //自定义key的生成策略, 将方法参数转换为hashcode, 作为redis key. 需要做两个事情, 一是添加一个自定义的ConversionService, 另一个是需要自定义一个KeyGenerator.
                .withConversionService(new CacheKeyConversionService());
    }
```

原文链接：spring boot 2.1.3 使用 mybatis redis cache 缓存

```
    /**
     * 创建redisTemplate工具
     * @param factory
     * @return
     */
    @Bean
    public RedisTemplate redisTemplate(RedisConnectionFactory factory) {
        StringRedisTemplate template = new StringRedisTemplate(factory);
        template.setKeySerializer(new StringRedisSerializer());//设置key序列化类，否则key前面会
了一些乱码
        template.setValueSerializer(createJackson2JsonRedisSerializer());//设置value序列化
        template.setHashKeySerializer(createJackson2JsonRedisSerializer());//设置 hash key 序列

        template.setHashValueSerializer(createJackson2JsonRedisSerializer());//设置 hash value
列化
        template.setEnableTransactionSupport(true);//设置redis支持数据库的事务
        template.afterPropertiesSet();//初始化设置并且生效
        return template;
    }

    /**
     * 创建redis序列化
     * @return
     */
    private RedisSerializer<Object> createJackson2JsonRedisSerializer() {
        ObjectMapper objectMapper = new ObjectMapper();
        Jackson2JsonRedisSerializer<Object> jackson2JsonRedisSerializer = new Jackson2JsonR
disSerializer<>(Object.class);
        objectMapper.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
        objectMapper.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
        jackson2JsonRedisSerializer.setObjectMapper(objectMapper);
        return jackson2JsonRedisSerializer;
    }

}

public class CacheKeyConversionService implements ConversionService {
    @Override
    public boolean canConvert(@Nullable Class<?> sourceType, Class<?> targetType) {
        return true;
    }

    @Override
    public boolean canConvert(@Nullable TypeDescriptor sourceType, TypeDescriptor targetT
pe) {
        return true;
    }

    @Nullable
    @Override
    public <T> T convert(@Nullable Object source, Class<T> targetType) {
        return (T) convert(source);
    }
```

```
    @Nullable
    @Override
    public Object convert(@Nullable Object source, @Nullable TypeDescriptor sourceType, Ty
eDescriptor targetType) {
        return convert(source);
    }

    private Object convert(Object source) {
        if (source instanceof CacheHashCode) {
            return ((CacheHashCode) source).hashString();
        }
        return CacheHashCode.of(source).hashString();
    }
}
```

第四步：在需要存入redis缓存的方法上添加注解



注解介绍

- @Cacheable

获取缓存 如果有缓存 直接返回

属性	类型	功能
value	String[]	缓存的名称 和cacheNames功能一样
cacheNames	String[]	缓存的名和value功能一样
key	String	缓存key的值、默认以所有的参数作为key、也可以直接配置keyGenerator
keyGenerator	String	缓存key生成器
cacheManager	String	配置使那个缓存管理器、和cacheResolver排斥
cacheResolver	String	定义使用个拦截器、和cacheManager互斥

condition	String	根据spel表达来可以配置什么条件下进行缓存 默认全部缓存
unless	String	和condition相反
sync	boolean	是否开启同步功、默认不开启

- @CachePut

执行并且更新缓存相关 不管如何 肯定会执行方法 然后返回 这样可以更新缓存的内容

属性	**类型**	**功能**
value	String[]	缓存的名称 和cacheNames功能一样
cacheNames	String[]	缓存的名和value功能一样
key	String	缓存key的值、默认以所有的参数作为key、也可以直接配置keyGenerator
keyGenerator	String	缓存key生成器
cacheManager	String	配置使那个缓存管理器、和cacheResolver排斥
cacheResolver	String	定义使用个拦截器、和cacheManager互斥
condition	String	根据spel表达来可以配置什么条件下进行缓存 默认全部缓存
unless	String	和condition相反

- @CacheEvict

删除缓存相关

属性	**类型**	**功能**
value	String[]	缓存的名称 和cacheNames功能一样
cacheNames	String[]	缓存的名和value功能一样
key	String	缓存key的值、默认以所有的参数作为key、也可以直接配置keyGenerator
keyGenerator	String	缓存key生成器
cacheManager	String	配置使那个缓存管理器、和cacheResolver排斥
cacheResolver	String	定义使用个拦截器、和cacheManager互斥
condition	String	根据spel表达来可以配置什么条件下进行缓存 默认全部缓存

allEntries	boolean	是否删除所有键的缓存 默认不删除
beforeInvocation	boolean	是否在调用此方法前 删除缓存

- @CacheConfig

在类级别统一的配置缓存公共配置

属性	类型	功能
cacheNames	String[]	缓存的名称 和value功能一样
keyGenerator	String	缓存key生成器
cacheManager	String	配置使用那个缓存管理器、和cacheResolver排斥
cacheResolver	String	定义使用个拦截器、和cacheManager互斥

- @EnableCaching

开启缓存以及缓存的全局配置

属性	类型	功能
proxyTargetClass	boolean	是否要基于cglib生成代理去实现缓存
mode	AdviceMode	配置那种方式去实现缓存、默认是AdviceMode.PROXY 可以切换为 AdviceMode#ASPECTJ
order	int	设置缓存管理器执行顺序

- @Caching

对多个缓存组的配置

属性	类型	功能
cacheable	Cacheable	配置获取缓存相关的配置
put	CachePut	配置如何更新缓存的相关配置
evict	CacheEvict	配置如何删除缓存的相关配置

注解介绍引用： https://www.jianshu.com/p/fd950f65aec7

第五步：在启动类配置开启缓存

```
*/
@SpringBootApplication
@MapperScan("com.springboot.dao")// mapper 接口类扫描包配置
@EnableCaching //开启缓存
public class Study2019Application extends SpringBootServletInitializer {

    /**
     * 部署war包需要继承SpringApplicationBuilder，重写configure
     * @param builder
     * @return
     */
```

第六步：运行查看效果

controller

```
    }

    @ApiOperation(value = "根据用户姓名获取用户信息")
    @ApiImplicitParams({
            @ApiImplicitParam(paramType = "query",name = "name",value = "姓名",
    })
    @RequestMapping(value = "/user/name",method = RequestMethod.GET)
    public User findByName(@RequestParam("name") String name) {
        return testService.findByName(name);
    }
```

查看redis客户端存储的数据