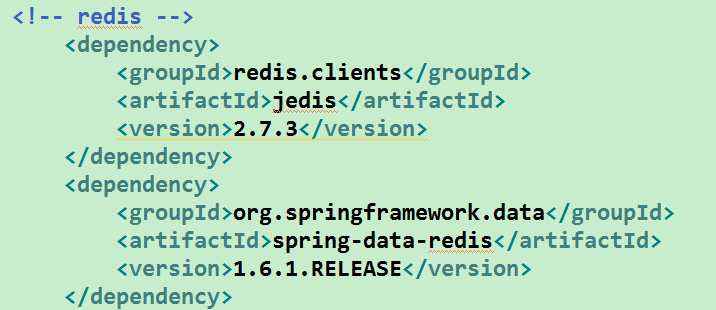
Redis在spring中的应用

1. 要想用redis，首先导入jar到pom中：



1. 进行redis的配置（applicationContext-redis.xml）：

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:jee=*"http://www.springframework.org/schema/jee"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-4.0.xsd"*>  <context:property-placeholder location=*"classpath:resouces/db.properties"* />  <!-- jedis 配置 redis 的连接池 -->  <bean id=*"poolConfig"* class=*"redis.clients.jedis.JedisPoolConfig"*>  <property name=*"maxIdle"* value=*"${redis.maxIdle}"*/>  <property name=*"maxWaitMillis"* value=*"${redis.maxWait}"*/>  <property name=*"testOnBorrow"* value=*"${redis.testOnBorrow}"*/>  </bean>    <!-- redis的连接工厂 -->  <bean id=*"connectionFactory"* class=*"org.springframework.data.redis.connection.jedis.JedisConnectionFactory"*>  <property name=*"poolConfig"* ref=*"poolConfig"*/>  <property name=*"port"* value=*"${redis.port}"*/>  <property name=*"hostName"* value=*"${redis.host}"*/>  <property name=*"password"* value=*"${redis.pass}"*/>  <property name=*"database"* value=*"${redis.db}"*/>  <property name=*"timeout"* value=*"${redis.timeout}"*></property>  </bean>  <!-- redis操作模板，这里采用尽量面向对象的模板 -->  <bean id=*"redisTemplate"* class=*"org.springframework.data.redis.core.RedisTemplate"*>  <property name=*"connectionFactory"* ref=*"connectionFactory"* />  <!-- 如果不配置Serializer，那么存储的时候只能使用String，如果用对象类型存储，那么会提示错误 can't cast to String！！！ -->  <property name=*"keySerializer"*>  <bean  class=*"org.springframework.data.redis.serializer.StringRedisSerializer"* />  </property>  <property name=*"valueSerializer"*>  <bean  class=*"org.springframework.data.redis.serializer.JdkSerializationRedisSerializer"* />  </property>  </bean>    <bean id=*"studentCache"* class=*"com.sunshine.cache.impl.StudentCache"*>  <property name=*"cache"* value=*"redis:student"*/>  </bean>    <bean id=*"cacheManager"* class=*"com.sunshine.cache.impl.CacheManagerImpl"*>  <property name=*"caches"*>  <list>  <ref bean=*"studentCache"*/>  </list>  </property>  </bean>    <bean id=*"jedisClient"* class=*"com.sunshine.dao.impl.JedisClientImpl"*/>    <!-- cache配置 -->  <bean id=*"methodCacheInterceptor"* class=*"com.sunshine.intercept.MethodCacheInterceptor"*>  <property name=*"redisUtil"* ref=*"redisUtil"* />  </bean>  <bean id=*"redisUtil"* class=*"com.sunshine.util.RedisUtil"*>  <property name=*"redisTemplate"* ref=*"redisTemplate"* />  </bean>  <bean id=*"dataSourceExchange"* class=*"com.sunshine.util.DataSourceExchange"*/>  <!--配置切面拦截方法 -->  <aop:config proxy-target-class=*"true"*>  <!--将com.crossoverJie.service包下的所有select开头的方法加入拦截 去掉select则加入所有方法w -->  <aop:pointcut id=*"controllerMethodPointcut"*  expression=*"*  *execution(\* com.sunshine.service.\*.\*(..))"* />  <aop:advisor advice-ref=*"methodCacheInterceptor"*  pointcut-ref=*"controllerMethodPointcut"* />    <!--所有数据库操作的方法加入切面-->  <aop:aspect ref=*"dataSourceExchange"*>  <aop:pointcut id=*"dataSourcePointcut"* expression=*"execution(\* com.sunshine.service.\*.\*(..))"*/>  <aop:before pointcut-ref=*"dataSourcePointcut"* method=*"before"*/>  <aop:after pointcut-ref=*"dataSourcePointcut"* method=*"after"*/>  </aop:aspect>  </aop:config>  </beans> |

1. 要想用redis需要创建一个jedis对象（因为用的是spring连接工厂连接redis，所以用JedisConnectionFactory对象创建）：

|  |
| --- |
| 1.@Autowired  **private** JedisConnectionFactory jedisConnectionFactory;  2.Jedis jedis = jedisConnectionFactory.getShardInfo().createResource(); |

1. 创建redis的接口和实现类，方便使用：

1）接口：

|  |
| --- |
| **package** com.sunshine.dao;  **import** java.util.Map;  **import** java.util.Set;  **public** **interface** JedisClient {    String setValue(String key, String value);  String getValue(String key);  String hgetValue(String hkey, String key);  String hmsetValue(String key, Map<String, String> map);  Set<String> hkeysValue(String key);  **long** hsetValue(String hkey, String key, String value);  String get(String key); //得到redis中的key值，直接用get key的名字在redis中查询  String set(String key, String value);//王redis中加入key值和value值  String hget(String hkey, String key);//用hash的方式得到值，  **long** hset(String hkey, String key, String value);  **long** incr(String key);  **long** expire(String key,**int** second);  **long** ttl(String key);  String hmset(String key, Map<String, String> map);  Set<String> hkeys(String key);  String hmsetValue(String key, Object value);  } |

1. 实现类：

|  |
| --- |
| **package** com.sunshine.dao.impl;  **import** java.util.HashMap;  **import** java.util.Map;  **import** java.util.Set;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.data.redis.connection.jedis.JedisConnectionFactory;  **import** org.springframework.data.redis.serializer.RedisSerializer;  **import** org.springframework.stereotype.Service;  **import** com.sunshine.dao.JedisClient;  **import** com.sunshine.util.JsonUtils;  **import** redis.clients.jedis.Jedis;  **import** redis.clients.jedis.JedisPool;  @Service  **public** **class** JedisClientImpl **implements** JedisClient {    **private** Logger logger = LoggerFactory.*getLogger*(getClass());    // @Autowired  **private** JedisPool jedisPool;    @Autowired  **private** JedisConnectionFactory jedisConnectionFactory;    **protected** RedisSerializer<String> strSerializer;    **public** String setValue(String key,String value){  Jedis jedis = jedisConnectionFactory.getShardInfo().createResource();  String string = jedis.set(key, value);  jedis.close();  **return** string;  }    **public** String get(String key) {  Jedis jedis = jedisPool.getResource();  String string = jedis.get(key);  jedis.close();  **return** string;  }  **public** String set(String key, String value) {  Jedis jedis = jedisPool.getResource();  String string = jedis.set(key, value);  jedis.close();  **return** string;  }  **public** String hget(String hkey, String key) {  Jedis jedis = jedisPool.getResource();  String string = jedis.hget(hkey, key);  jedis.close();  **return** string;  }  **public** **long** hset(String hkey, String key, String value) {  Jedis jedis = jedisPool.getResource();  Long result = jedis.hset(hkey, key, value);  jedis.close();  **return** result;  }  **public** **long** incr(String key) {  Jedis jedis = jedisPool.getResource();  Long result = jedis.incr(key);  jedis.close();  **return** result;  }  **public** **long** expire(String key, **int** second) {  Jedis jedis = jedisPool.getResource();  Long result = jedis.expire(key, second);  jedis.close();  **return** result;  }  **public** **long** ttl(String key) {  Jedis jedis = jedisPool.getResource();  Long result = jedis.ttl(key);  jedis.close();  **return** result;  }    **public** String hmset(String key, Map<String, String> map) {  Jedis jedis = jedisPool.getResource();  String string = jedis.hmset(key, map);  jedis.close();  **return** string;  }    **public** Set<String> hkeys(String key) {  Jedis jedis = jedisPool.getResource();  Set<String> set = jedis.hkeys(key);  jedis.close();  **return** set;  }  @Override  **public** String getValue(String key) {  Jedis jedis = jedisConnectionFactory.getShardInfo().createResource();  String string = jedis.get(key);  jedis.close();  **return** string;  }  @Override  **public** String hgetValue(String hkey, String key) {  Jedis jedis = jedisConnectionFactory.getShardInfo().createResource();  String string = jedis.hget(hkey, key);  jedis.close();  **return** string;  }  @Override  **public** String hmsetValue(String key, Object value) {  Jedis jedis = jedisConnectionFactory.getShardInfo().createResource();  String string = JsonUtils.*objectToJson*(value);  **byte**[] buff = key.getBytes();  **byte**[] buffValue = string.getBytes();  Map<**byte**[], **byte**[]> tuple = **new** HashMap<>();  tuple.put(buff, buffValue);  jedis.hmset(buff, tuple);  jedis.close();  **return** string;  }  @Override  **public** Set<String> hkeysValue(String key) {  Jedis jedis = jedisConnectionFactory.getShardInfo().createResource();  Set<String> set = jedis.hkeys(key);  jedis.close();  **return** set;  }  @Override  **public** **long** hsetValue(String hkey, String key, String value) {  Jedis jedis = jedisConnectionFactory.getShardInfo().createResource();  Long result = jedis.hset(hkey, key, value);  jedis.close();  **return** result;  }  @Override  **public** String hmsetValue(String key, Map<String, String> map) {  Jedis jedis = jedisConnectionFactory.getShardInfo().createResource();  String string = jedis.hmset(key, map);  jedis.close();  **return** string;  }  } |

1. 使用spring的set注入，进行一部分数据在项目启动的时候就加进redis中，比如：前台一些内容、图片地址等等，在前台出现频率高的数据：
2. 在redis配置文件中加入set注入的bean：

|  |
| --- |
|  |

1. 建立bean所对应的java类

2.1）CacheManager.java

|  |
| --- |
| **package** com.sunshine.cache;  **public** **interface** CacheManager {  /\*\*  \* 根据 name 获取 Cache.  \*  \* **@param** name  \* cache name  \* **@param** <T>  \* cache 元素类型  \* **@return** Cache, may be null  \*/  <T> Cache getCache(String name);  /\*\*  \* 注册 cache.  \* <p>  \* 根据 cache 的 name 注册.  \*  \* **@param** cache  \* cache  \*/  **void** addCache(Cache cache);  } |

2.2）CacheManagerImpl.java

|  |
| --- |
| package com.sunshine.cache.impl;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  import org.apache.commons.lang3.StringUtils;  import org.springframework.context.ApplicationContext;  import com.sunshine.cache.Cache;  import com.sunshine.cache.CacheManager;  import com.sunshine.core.AppContextInitListener;  public class CacheManagerImpl/\* implements CacheManager, AppContextInitListener \*/{  private HashMap<String, Cache> cacheMap = new HashMap<>();  private List<Cache> caches;  public void setCaches(List<Cache> caches) {  this.caches = caches;  if (caches != null) {  for (Cache c : caches) {  // cacheMap.put(c.getName(), c);  c.init();  }  }  }  public List<Cache> getCaches() {  return caches;  }  // @Override  // public <T> Cache getCache(String name) {  // return cacheMap.get(name);  // }  //  // @Override  // public void addCache(Cache cache) {  // if (caches == null) {  // caches = new ArrayList<>();  // }  // if (!caches.contains(cache)) {  // caches.add(cache);  // }  //// cacheMap.put(cache.getName(), cache);  // }  //  // @Override  // public void contextInitialized(ApplicationContext applicationContext) {  // Map<String, Cache> cacheBeans = applicationContext.getBeansOfType(Cache.class);  // if (cacheBeans != null) {  // cacheBeans.forEach((k, v) -> {  // if (!caches.contains(v)) {  //// if (StringUtils.isEmpty(v.getName())) {  //// throw new RuntimeException(v + " cacheName is empty");  //// }  //// addCache(v);  // v.init();  // }  // });  // }  // }  } |

2.3）Cache.java

|  |
| --- |
| **package** com.sunshine.cache;  **public** **interface** Cache {    **void** init();  } |

2.4）StudentCace.java

|  |
| --- |
| **package** com.sunshine.cache.impl;  **import** java.util.HashMap;  **import** java.util.Iterator;  **import** java.util.List;  **import** java.util.Map;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.data.redis.serializer.RedisSerializer;  **import** com.sunshine.cache.Cache;  **import** com.sunshine.dao.JedisClient;  **import** com.sunshine.mapper.StudentMapper;  **import** com.sunshine.pojo.Student;  **import** com.sunshine.util.JsonUtils;  **public** **class** StudentCache **implements** Cache {  @Autowired  **private** StudentMapper studentMapper;    @Autowired  **private** JedisClient jedisClient;    **private** String cache;    **public** String getCache() {  **return** cache;  }  **public** **void** setCache(String cache) {  **this**.cache = cache;  }  @Override  **public** **void** init() {  updateStudent();  }    **protected** RedisSerializer<String> strSerializer;    **public** **void** setValue(String key, Object value){  String string = JsonUtils.*objectToJson*(value);  Map<**byte**[], **byte**[]> tuple = **new** HashMap<>();  tuple.put(strSerializer.serialize(key), strSerializer.serialize(string));  // jedisClient.hmsetValue(strSerializer.serialize(key), tuple);  }  **public** **void** updateStudent() {  List<Student> list = studentMapper.getAll();  Iterator studentList = list.iterator();  **while**(studentList.hasNext()){  Student studentNext = (Student) studentList.next();  Map<String, Object> maps = **new** HashMap<>();  maps.put("id", studentNext.getId());  maps.put("name", studentNext.getName());  jedisClient.hmsetValue(cache+":"+studentNext.getId()+"."+studentNext.getAbc(), maps);  }  }  } |