此文章已于 9:21:39 2017/12/11 发布到 HealerJean梦想博客

spring redis 注解开发 单片机 集群 主从复制

类别 redis

# 失败记录 ，虽然最终没有成功，但是原理还是知道的

# 1、spring reids简单实现注解

<http://blog.csdn.net/fighterandknight/article/details/53432276/>

## 1、导入相应依赖

|  |
| --- |
| <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.8.1</version>  </dependency>  <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-redis</artifactId>  <version>1.7.2.RELEASE</version>  </dependency> |

## 2、redis配置信息properties

|  |
| --- |
| redis.host=127.0.0.1  redis.port=6379  redis.password=  redis.maxIdle=100  redis.maxActive=300  redis.maxWait=1000  redis.testOnBorrow=true  redis.timeout=10000 |

## 3、spring redis 配置文件

### 1、poolConfig ，导入reids信息，注意新旧版本参数名字不一样哦

### 2、JedisConnectionFactory，redis连接工厂，将poolConfig导入，并配置host，port，ip password等

### 3、redis处理类 JedisTemplate,将连接工厂导入进来，（spring boot中用来处理key的序列化）

### 4、缓存管理器，配置cache（这个时候，需要自己写一个RedisCach类，将JedisTemplate 导入，并设置缓存位置名称name，（这里就是相当于Ehcache 管理管理工厂）这个时候还可以配置多个redis缓存名称）

|  |
| --- |
| <?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:mvc="http://www.springframework.org/schema/mvc"  xmlns:cache="http://www.springframework.org/schema/cache"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-4.2.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd  http://www.springframework.org/schema/cache  http://www.springframework.org/schema/cache/spring-cache-4.2.xsd">    <context:property-placeholder location="classpath:redis-config.properties" />  <!-- 启用缓存注解功能，这个是必须的，否则注解不会生效，另外，该注解一定要声明在spring主配置文件中才会生效 -->  <cache:annotation-driven cache-manager="cacheManager" />    <!-- redis 相关配置 -->  <bean id="poolConfig" class="redis.clients.jedis.JedisPoolConfig">  <property name="maxIdle" value="${redis.maxIdle}" />  <property name="maxWaitMillis" value="${redis.maxWait}" />  <property name="testOnBorrow" value="${redis.testOnBorrow}" />  </bean>  <bean id="JedisConnectionFactory" class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory"  p:host-name="${redis.host}"  p:port="${redis.port}"  p:password="${redis.pass}"  p:pool-config-ref="poolConfig"/>    <bean id="redisTemplate" class="org.springframework.data.redis.core.RedisTemplate">  <property name="connectionFactory" ref="JedisConnectionFactory" />  </bean>    <!-- spring自己的缓存管理器，这里定义了缓存位置名称 ，即注解中的value 名称和Ehcache有点像，是吧，哈哈-->  <bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">  <property name="caches">  <set>  <!-- 这里可以配置多个redis -->  <bean class="com.cn.util.RedisCache">  <property name="redisTemplate" ref="redisTemplate" />  <property name="name" value="default"/>  </bean>  <bean class="com.cn.util.RedisCache">  <property name="redisTemplate" ref="redisTemplate" />  <property name="name" value="common"/>  <!-- common名称要在类或方法的注解中使用 -->  </bean>  </set>  </property>  </bean>    </beans> |

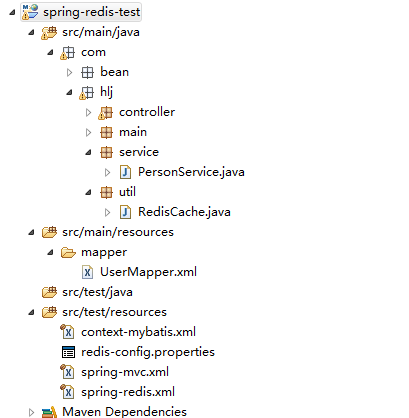
## 4、RedisCache为使用注解开发，缓存类

|  |
| --- |
| **public** **class** RedisCache **implements** Cache{  **private** RedisTemplate<String, Object> redisTemplate;  **private** String name;  **public** RedisTemplate<String, Object> **getRedisTemplate**() {  **return** redisTemplate;  }    **public** **void** **setRedisTemplate**(RedisTemplate<String, Object> redisTemplate) {  **this**.redisTemplate = redisTemplate;  }    **public** **void** **setName**(String name) {  **this**.name = name;  }    @Override  **public** String **getName**() {    **return** **this**.name;  }  @Override  **public** Object **getNativeCache**() {    **return** **this**.redisTemplate;  }    @Override  **public** ValueWrapper **get**(Object key) {  System.out.println("get key");  **final** String keyf = key.toString();  Object object = **null**;  object = redisTemplate.execute(**new** RedisCallback<Object>() {  **public** Object **doInRedis**(RedisConnection connection)  **throws** DataAccessException {  **byte**[] key = keyf.getBytes();  **byte**[] value = connection.get(key);  **if** (value == **null**) {  **return** **null**;  }  **return** toObject(value);  }  });  **return** (object != **null** ? **new** SimpleValueWrapper(object) : **null**);  }    @Override  **public** **void** **put**(Object key, Object value) {  System.out.println("put key");  **final** String keyf = key.toString();  **final** Object valuef = value;  **final** **long** liveTime = 86400;  redisTemplate.execute(**new** RedisCallback<Long>() {  **public** Long **doInRedis**(RedisConnection connection)  **throws** DataAccessException {  **byte**[] keyb = keyf.getBytes();  **byte**[] valueb = toByteArray(valuef);  connection.set(keyb, valueb);  **if** (liveTime > 0) {  connection.expire(keyb, liveTime);  }  **return** 1L;  }  });  }  **private** **byte**[] **toByteArray**(Object obj) {  **byte**[] bytes = **null**;  ByteArrayOutputStream bos = **new** ByteArrayOutputStream();  **try** {  ObjectOutputStream oos = **new** ObjectOutputStream(bos);  oos.writeObject(obj);  oos.flush();  bytes = bos.toByteArray();  oos.close();  bos.close();  }**catch** (IOException ex) {  ex.printStackTrace();  }  **return** bytes;  }  **private** Object **toObject**(**byte**[] bytes) {  Object obj = **null**;  **try** {  ByteArrayInputStream bis = **new** ByteArrayInputStream(bytes);  ObjectInputStream ois = **new** ObjectInputStream(bis);  obj = ois.readObject();  ois.close();  bis.close();  } **catch** (IOException ex) {  ex.printStackTrace();  } **catch** (ClassNotFoundException ex) {  ex.printStackTrace();  }  **return** obj;  }    @Override  **public** **void** **evict**(Object key) {  System.out.println("del key");  **final** String keyf = key.toString();  redisTemplate.execute(**new** RedisCallback<Long>() {  **public** Long **doInRedis**(RedisConnection connection)  **throws** DataAccessException {  **return** connection.del(keyf.getBytes());  }  });  }    @Override  **public** **void** **clear**() {  System.out.println("clear key");  redisTemplate.execute(**new** RedisCallback<String>() {  **public** String **doInRedis**(RedisConnection connection)  **throws** DataAccessException {  connection.flushDb();  **return** "ok";  }  });  }  **public** <**T**> **T** **get**(Object key, Class<**T**> type) {  **return** **null**;  }    **public** ValueWrapper **putIfAbsent**(Object key, Object value) {  **return** **null**;  }  } |

## 5、service 使用注解开发

|  |
| --- |
| @Service  **public** **class** PersonService{  @Autowired  **public** PersonMapper personMapper;    @Cacheable(value="common",key="'id\_'+#id")  **public** Person **selectByPrimaryKey**(**long** id) {  System.out.println("======================");  System.out.println("======================");  System.out.println("======================");  **return** personMapper.selectByPrimaryKey(id);  }    @CachePut(value="common",key="#person.getName()")  **public** **void** **insertSelective**(Person person) {  personMapper.insertSelective(person);  System.out.println("########################");  System.out.println("########################");  System.out.println("########################");  }    @CacheEvict(value="common",key="'id\_'+#id")  **public** **void** **deleteByPrimaryKey**(**long** id) {  personMapper.deleteByPrimaryKey(id);  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  }  } |

## 6、代码位置



# 2、spring 集群 注解

## 1、导入依赖包

## 2、集群 信息 redis-cluster.properties

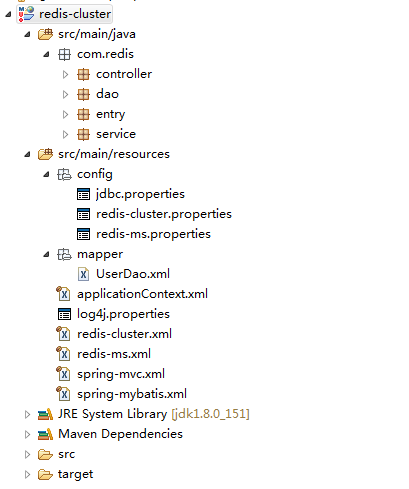
|  |
| --- |
| redis.host0=192.168.1.235  redis.port0=7000  redis.host1=192.168.1.235  redis.port1=7001  redis.host2=192.168.1.235  redis.port2=7002  redis.host3=192.168.1.235  redis.port3=7003  redis.host4=192.168.1.235  redis.port4=7004  redis.host5=192.168.1.235  redis.port5=7005  redis.maxRedirects=3  redis.maxIdle=30  redis.maxTotal=100  redis.minIdle=5  redis.maxWaitMillis=30000  redis.testOnBorrow=true  redis.testOnReturn=true  redis.testWhileIdle=true  redis.timeout=3000 |

## 3、编辑spring redis 配置文件

### 1、 相当于jedis连接工厂中添加的是一个集群信息，其他的和单片机其实是一样的

|  |
| --- |
| <?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:mvc="http://www.springframework.org/schema/mvc"  xmlns:cache="http://www.springframework.org/schema/cache"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-4.2.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd  http://www.springframework.org/schema/cache  http://www.springframework.org/schema/cache/spring-cache-4.2.xsd">  <!-- 引入配置文件 -->  <bean id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  <property name="location" value="classpath:properties/redis-cluster.properties" />  </bean>    <!-- 启用缓存注解功能，这个是必须的，否则注解不会生效，另外，该注解一定要声明在spring主配置文件中才会生效 -->  <cache:annotation-driven cache-manager="cacheManager" />    <bean id="jedisPoolConfig" class="redis.clients.jedis.JedisPoolConfig">  <property name="maxTotal" value="${redis.maxTotal}" />  <!--最大空闲数-->  <property name="maxIdle" value="${redis.maxIdle}" />  <!--最大建立连接等待时间-->  <property name="maxWaitMillis" value="${redis.maxWaitMillis}" />  <!--是否在从池中取出连接前进行检验,如果检验失败,则从池中去除连接并尝试取出另一个-->  <property name="testOnBorrow" value="${redis.testOnBorrow}" />  </bean>    <bean id="redisClusterConfiguration" class="org.springframework.data.redis.connection.RedisClusterConfiguration">  <property name="maxRedirects" value="${redis.maxRedirects}"></property>  <property name="clusterNodes">  <set>  <bean class="org.springframework.data.redis.connection.RedisNode">  <constructor-arg name="host"  value="${redis.host0}">  </constructor-arg>  <constructor-arg name="port"  value="${redis.port0}">  </constructor-arg>  </bean>  <bean class="org.springframework.data.redis.connection.RedisNode">  <constructor-arg name="host"  value="${redis.host1}">  </constructor-arg>  <constructor-arg name="port"  value="${redis.port1}">  </constructor-arg>  </bean>  >  <bean class="org.springframework.data.redis.connection.RedisNode">  <constructor-arg name="host"  value="${redis.host2}">  </constructor-arg>  <constructor-arg name="port"  value="${redis.port2}">  </constructor-arg>  </bean>  <bean class="org.springframework.data.redis.connection.RedisNode">  <constructor-arg name="host"  value="${redis.host2}">  </constructor-arg>  <constructor-arg name="port"  value="${redis.port2}">  </constructor-arg>  </bean>  <bean class="org.springframework.data.redis.connection.RedisNode">  <constructor-arg name="host"  value="${redis.host3}">  </constructor-arg>  <constructor-arg name="port"  value="${redis.port3}">  </constructor-arg>  </bean>  <bean class="org.springframework.data.redis.connection.RedisNode">  <constructor-arg name="host"  value="${redis.host4}">  </constructor-arg>  <constructor-arg name="port"  value="${redis.port5}">  </constructor-arg>  </bean>  </set>  </property>  </bean>      <bean id="jeidsConnectionFactory" class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory" >  <constructor-arg name="clusterConfig" ref="redisClusterConfiguration"/>  <constructor-arg name="poolConfig" ref="jedisPoolConfig"/>  <property name="password" value="${redis.password}" />  <property name="timeout" value="${redis.timeout}" />  </bean>      <bean id="redisTemplate" class="org.springframework.data.redis.core.RedisTemplate">  <!-- 如果不配置Serializer，那么存储的时候缺省使用String，  如果用User类型存储，那么会提示错误User can't cast to String！！  //在spring boot的时候学的，key序列化方式;（不然会出现乱码;）,但是如果方法上有Long等非String类型的话，会报类型转换错误-->  <property name="connectionFactory" ref="jeidsConnectionFactory" />  <property name="keySerializer" >  <bean class="org.springframework.data.redis.serializer.StringRedisSerializer" />  </property>  <property name="valueSerializer" >  <bean class="org.springframework.data.redis.serializer.JdkSerializationRedisSerializer" />  </property>  <property name="hashKeySerializer">  <bean class="org.springframework.data.redis.serializer.StringRedisSerializer"/>  </property>  <property name="hashValueSerializer">  <bean class="org.springframework.data.redis.serializer.JdkSerializationRedisSerializer"/>  </property>    <!-- spring自己的缓存管理器，这里定义了缓存位置名称 ，即注解中的value -->  <bean id="cacheManager" class="org.springframework.cache.support.SimpleCacheManager">  <property name="caches">  <set>  <!-- 这里可以配置多个redis -->  <bean class="com.redis.redis.RedisCache">  <property name="redisTemplate" ref="redisTemplate" />  <property name="name" value="defaultCache"/>  </bean>  </set>  </property>  </bean>    </beans> |

## 4、代码位置



# 3、主从复制，哨兵模式

## 1.、主从复制配置文件 （这里是添加了两个redis ，一个是主，一个是slave，下面的mymaster 具体看主从复制那个教程）

|  |
| --- |
| # Redis settings  #sentinel1的IP和端口  sentinel1.host=192.168.1.233  sentinel1.port=26379  #sentinel2的IP和端口  sentinel2.host=192.168.1.233  sentinel2.port=26378  im.hs.server.redis.maxIdle=500  #最大连接数，超过此连接时操作redis会报错  im.hs.server.redis.maxTotal=5000  im.hs.server.redis.maxWaitTime=1000  im.hs.server.redis.testOnBorrow=true  #最小闲置连接数，spring启动的时候自动建立该数目的连接供应用程序使用，不够的时候会申请。  im.hs.server.redis.minIdle=300  im.hs.server.redis.sentinel.masterName=mymaster |

## 2、spring redis 配置文件（这里没有添加缓存管理器）

|  |
| --- |
| <?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:mvc="http://www.springframework.org/schema/mvc"  xmlns:tx="http://www.springframework.org/schema/tx"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.0.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx-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/mvc  http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">        <bean id="poolConfig" class="redis.clients.jedis.JedisPoolConfig">  <property name="maxTotal" value="${im.hs.server.redis.maxTotal}" />  <property name="minIdle" value="${im.hs.server.redis.minIdle}" />  <property name="maxWaitMillis" value="${im.hs.server.redis.maxWaitTime}" />  <property name="maxIdle" value="${im.hs.server.redis.maxIdle}" />  <property name="testOnBorrow" value="${im.hs.server.redis.testOnBorrow}" />  <property name="testOnReturn" value="true" />  <property name="testWhileIdle" value="true" />  </bean>    <bean id="redisSentinelConfiguration" class="org.springframework.data.redis.connection.RedisSentinelConfiguration">  <property name="master">  <bean class="org.springframework.data.redis.connection.RedisNode">  <property name="name" value="${im.hs.server.redis.sentinel.masterName}"/>  </bean>  </property>  <property name="sentinels">  <set>  <bean class="org.springframework.data.redis.connection.RedisNode">  <constructor-arg name="host" value="${sentinel1.host}"></constructor-arg>  <constructor-arg name="port" value="${sentinel1.port}"></constructor-arg>  </bean>  <bean class="org.springframework.data.redis.connection.RedisNode">  <constructor-arg name="host" value="${sentinel2.host}"></constructor-arg>  <constructor-arg name="port" value="${sentinel2.port}"></constructor-arg>  </bean>  </set>  </property>  </bean>  <bean id="jeidsConnectionFactory"  class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory">  <constructor-arg name="sentinelConfig"  ref="redisSentinelConfiguration">  </constructor-arg>  <constructor-arg name="poolConfig" ref="poolConfig"></constructor-arg>  </bean>  <bean id="redisTemplate" class="org.springframework.data.redis.core.RedisTemplate">  <property name="connectionFactory" ref="jeidsConnectionFactory"/>  </bean>    </beans> |

## 3、代码位置

