此文章已于 9:31:43 2017/12/7 重新发布到 HealerJean梦想博客

6.2、redis集群和单片机spring

类别 redis

# 1、单片机 JedisPool Jedis

## 1、我没有用maven项目，所以引入了jar包

|  |
| --- |
| <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-redis</artifactId>  <version>1.7.2.RELEASE</version>  </dependency>  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.9.0</version>  </dependency>  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-pool2</artifactId>  <version>2.4.3</version>  </dependency> |

## 2、spring 配置文件，添加JedisPool

|  |
| --- |
| <?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:mvc=*"http://www.springframework.org/schema/mvc"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.2.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.2.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-3.2.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx-3.2.xsd "*>    <!-- 连接池配置 -->  <bean id=*"jedisPoolConfig"* class=*"redis.clients.jedis.JedisPoolConfig"*>  <!-- 最大连接数 -->  <property name=*"maxTotal"* value=*"30"* />  <!-- 最大空闲连接数 -->  <property name=*"maxIdle"* value=*"10"* />  <!-- 每次释放连接的最大数目 -->  <property name=*"numTestsPerEvictionRun"* value=*"1024"* />  <!-- 释放连接的扫描间隔（毫秒） -->  <property name=*"timeBetweenEvictionRunsMillis"* value=*"30000"* />  <!-- 连接最小空闲时间 -->  <property name=*"minEvictableIdleTimeMillis"* value=*"1800000"* />  <!-- 连接空闲多久后释放, 当空闲时间>该值 且 空闲连接>最大空闲连接数 时直接释放 -->  <property name=*"softMinEvictableIdleTimeMillis"* value=*"10000"* />  <!-- 获取连接时的最大等待毫秒数,小于零:阻塞不确定的时间,默认-1 -->  <property name=*"maxWaitMillis"* value=*"1500"* />  <!-- 在获取连接的时候检查有效性, 默认false -->  <property name=*"testOnBorrow"* value=*"true"* />  <!-- 在空闲时检查有效性, 默认false -->  <property name=*"testWhileIdle"* value=*"true"* />  <!-- 连接耗尽时是否阻塞, false报异常,ture阻塞直到超时, 默认true -->  <property name=*"blockWhenExhausted"* value=*"false"* />  </bean>    <!-- redis单机 通过连接池 -->  <bean id=*"jedisPool"* class=*"redis.clients.jedis.JedisPool"* destroy-method=*"close"*>  <constructor-arg name=*"poolConfig"* ref=*"jedisPoolConfig"*/>  <constructor-arg name=*"host"* value=*"127.0.0.1"*/>  <constructor-arg name=*"port"* value=*"6379"*/>  </bean>    </beans> |

## 3、测试JedisPool

|  |
| --- |
| **public** **class** JedisPoolMain {    **static** Jedis *jedis* = **null**;    **public** **static** **void** main(String[] args) {    ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext-redis.xml");    JedisPool pool = (JedisPool) context.getBean("jedisPool");  **try** {  *jedis* = pool.getResource();  *jedis*.set("name", "spring jedis pool");  String name = *jedis*.get("name");  System.*out*.println(name);  }**catch**(Exception ex){  ex.printStackTrace();  }**finally**{  **if**(*jedis* != **null**){  //关闭连接  *jedis*.close();  }  }  }  } |

# 2、集群准备：

## 1、首先，要明确一个非常重要的一点

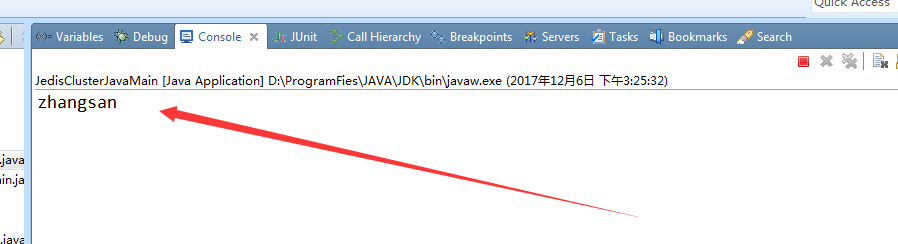
### 1、搭建集群命令，不使用127.0.0.1，因为我使用了127.0.0.1，但是这里不想动干戈，所以在虚拟机nat下进行了端口的映射。（正常情况下下面的代码中连接redis的ip应该为 192.168.189.130）

### redis-trib.rb create --replicas 1 192.168.189.130:7000 192.168.189.130:7001 192.168.189.130:7002 192.168.189.130:7003 192.168.189.130:7004 192.168.189.130:7005

## 2、JedisCluster java

|  |
| --- |
| **public** **class** JedisClusterJavaMain {  **public** **static** **void** main(String[] args) {  JedisPoolConfig config = **new** JedisPoolConfig();  // 最大连接数  config.setMaxTotal(30);  // 最大连接空闲数  config.setMaxIdle(2);  //集群结点  Set<HostAndPort> jedisClusterNode = **new** HashSet<HostAndPort>();  jedisClusterNode.add(**new** HostAndPort("127.0.0.1", 7000));  jedisClusterNode.add(**new** HostAndPort("127.0.0.1", 7001));  jedisClusterNode.add(**new** HostAndPort("127.0.0.1", 7002));  jedisClusterNode.add(**new** HostAndPort("127.0.0.1", 7003));  jedisClusterNode.add(**new** HostAndPort("127.0.0.1", 7004));  jedisClusterNode.add(**new** HostAndPort("127.0.0.1", 7005));  JedisCluster jc = **new** JedisCluster(jedisClusterNode, config);    JedisCluster jcd = **new** JedisCluster(jedisClusterNode);  jcd.set("name", "zhangsan");  String value = jcd.get("name");  System.*out*.println(value);  }  } |

#### 2、测试成功



## 3、spring搭建集群连接

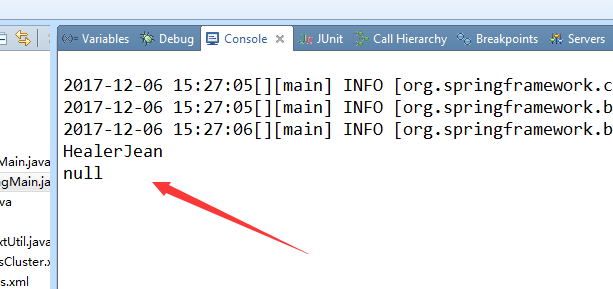
### 1、spring配置文件

|  |
| --- |
| <?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:mvc=*"http://www.springframework.org/schema/mvc"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.2.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.2.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-3.2.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx-3.2.xsd "*>      <!-- 连接池配置 -->  <bean id=*"jedisPoolConfig"* class=*"redis.clients.jedis.JedisPoolConfig"*>  <!-- 最大连接数 -->  <property name=*"maxTotal"* value=*"30"* />  <!-- 最大空闲连接数 -->  <property name=*"maxIdle"* value=*"10"* />  <!-- 每次释放连接的最大数目 -->  <property name=*"numTestsPerEvictionRun"* value=*"1024"* />  <!-- 释放连接的扫描间隔（毫秒） -->  <property name=*"timeBetweenEvictionRunsMillis"* value=*"30000"* />  <!-- 连接最小空闲时间 -->  <property name=*"minEvictableIdleTimeMillis"* value=*"1800000"* />  <!-- 连接空闲多久后释放, 当空闲时间>该值 且 空闲连接>最大空闲连接数 时直接释放 -->  <property name=*"softMinEvictableIdleTimeMillis"* value=*"10000"* />  <!-- 获取连接时的最大等待毫秒数,小于零:阻塞不确定的时间,默认-1 -->  <property name=*"maxWaitMillis"* value=*"1500"* />  <!-- 在获取连接的时候检查有效性, 默认false -->  <property name=*"testOnBorrow"* value=*"true"* />  <!-- 在空闲时检查有效性, 默认false -->  <property name=*"testWhileIdle"* value=*"true"* />  <!-- 连接耗尽时是否阻塞, false报异常,ture阻塞直到超时, 默认true -->  <property name=*"blockWhenExhausted"* value=*"false"* />  </bean>    <!-- redis集群 -->  <bean id=*"jedisCluster"* class=*"redis.clients.jedis.JedisCluster"*>  <constructor-arg index=*"0"*>  <set>  <bean class=*"redis.clients.jedis.HostAndPort"*>  <constructor-arg index=*"0"* value=*"127.0.0.1"*></constructor-arg>  <constructor-arg index=*"1"* value=*"7000"*></constructor-arg>  </bean>    <bean class=*"redis.clients.jedis.HostAndPort"*>  <constructor-arg index=*"0"* value=*"127.0.0.1"*></constructor-arg>  <constructor-arg index=*"1"* value=*"7001"*></constructor-arg>  </bean>  <bean class=*"redis.clients.jedis.HostAndPort"*>  <constructor-arg index=*"0"* value=*"127.0.0.1"*></constructor-arg>  <constructor-arg index=*"1"* value=*"7002"*></constructor-arg>  </bean>  <bean class=*"redis.clients.jedis.HostAndPort"*>  <constructor-arg index=*"0"* value=*"127.0.0.1"*></constructor-arg>  <constructor-arg index=*"1"* value=*"7003"*></constructor-arg>  </bean>  <bean class=*"redis.clients.jedis.HostAndPort"*>  <constructor-arg index=*"0"* value=*"127.0.0.1"*></constructor-arg>  <constructor-arg index=*"1"* value=*"7004"*></constructor-arg>  </bean>  <bean class=*"redis.clients.jedis.HostAndPort"*>  <constructor-arg index=*"0"* value=*"127.0.0.1"*></constructor-arg>  <constructor-arg index=*"1"* value=*"7005"*></constructor-arg>  </bean>    </set>  </constructor-arg>  <constructor-arg index=*"1"* ref=*"jedisPoolConfig"*></constructor-arg>  </bean>    </beans> |

### 2、JedisCluster

|  |
| --- |
| **public** **class** JedisClusterSpringMain {  **public** **static** **void** main(String[] args) {  ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext-jedisCluster.xml");  JedisCluster jedisCluster = (JedisCluster) context.getBean("jedisCluster");    jedisCluster.set("name", "HealerJean");  String value = jedisCluster.get("name");  System.*out*.println(value);    jedisCluster.del("name");  String delvalue = jedisCluster.get("name");  System.*out*.println(delvalue);  }  } |

### 3、测试成功



# 3、淘淘商城缓存 (记得将对象变成json放进去）

|  |
| --- |
| @Override  **public** TaotaoResult getContentList(**long** cid) **throws** Exception{  //缓存逻辑，先判断缓存中是否有内容  **try** {  String contentStr = cluster.hget(TB\_CONTENT\_KEY, cid + "");  **if** (!StringUtils.*isBlank*(contentStr)) {  //把json字符串转换成对象列表  List<TbContent> list = JsonUtils.*jsonToList*(contentStr, TbContent.**class**);  //返回结果  **return** TaotaoResult.*ok*(list);  }  } **catch** (Exception e) {  e.printStackTrace();  //缓存不能影响正常逻辑  }  //从数据库中加载数据  TbContentExample example = **new** TbContentExample();  //添加条件  Criteria criteria = example.createCriteria();  criteria.andCategoryIdEqualTo(cid);  List<TbContent> list = contentMapper.selectByExample(example);    //把结果添加到redis数据库中  **try** {  cluster.hset(TB\_CONTENT\_KEY, cid + "", JsonUtils.*objectToJson*(list));  } **catch** (Exception e) {  e.printStackTrace();  //缓存不能影响正常逻辑  }    //返回结果  **return** TaotaoResult.*ok*(list);  } |