Spring Cache使用详解

# 集成memcached

## 依赖

|  |
| --- |
| <dependency>  <groupId>com.googlecode.xmemcached</groupId>  <artifactId>xmemcached</artifactId>  <version>2.2.0</version>  </dependency> |

## spring配置

|  |
| --- |
| #在属性文件中添加如下配置  memcached.server.ip=172.16.130.169  memcached.server.port=11211  memcached.connection.pool.size=10  #sping配置  <bean id="memcachedClientBuilder" class="net.rubyeye.xmemcached.XMemcachedClientBuilder">  <constructor-arg>  <list>  <bean class="java.net.InetSocketAddress">  <constructor-arg value="${memcached.server.ip}" />  <constructor-arg value="${memcached.server.port}" />  </bean>  </list>  </constructor-arg>  <property name="connectionPoolSize" value="${memcached.connection.pool.size}" />  <property name="commandFactory">  <bean class="net.rubyeye.xmemcached.command.BinaryCommandFactory" />  </property>  <property name="transcoder">  <bean class="net.rubyeye.xmemcached.transcoders.SerializingTranscoder" />  </property>  </bean>  <bean id="memcachedClient" factory-bean="memcachedClientBuilder"  factory-method="build" destroy-method="shutdown" />  <bean id="cacheManager"  class="com.imooc.common.memcache.MemcachedCacheManager">  <property name="memcachedClient" ref="memcachedClient" />  <property name="configMap">  <map>  <!-- key：@Cacheable、@CachePut、@CacheEvict等的value属性。value：缓存过期时间(单位：秒)，默认值：0 -->  <entry key="defaultCache" value="0" /><!-- 默认缓存区 不过期 -->  <entry key="inTimeCache" value="3600" /><!-- 实时缓存区 一小时过期 -->  </map>  </property>  </bean>  <cache:annotation-driven cache-manager="cacheManager" /> |

注意：编写cacheManager实现。

# 集成spring data redis

## maven依赖

|  |
| --- |
| <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.8.0</version>  </dependency>  <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-redis</artifactId>  <version>1.7.5.RELEASE</version>  </dependency> |

## spring配置

# 通过注解使用缓存

spring提供了四个注解来声明缓存规则：

|  |  |
| --- | --- |
| 注解 | 描述 |
| @Cacheable | 表明Spring在调用方法之前，首先应该在缓存中查找方法的返回值。如果这个值能够找到，就会返回返回值。否则的话，这个方法就会被调用，返回值会放到缓存中。**方法不能为void**。 |
| @CachePut | 表明Spring应该将方法的返回值放到缓存中。方法调用前不会检查缓存，方法始终都会被调用。**方法不能为void**。 |
| @CacheEvict | 表明Spring应该在缓存中清除一个或者多个条目 |
| @Caching | 这是一个分组的注解，能够同时应用多个其他的缓存注解 |

表中的所有注解都可以应用到方法或类上。当将其放在单个方法上时，注解所描述的缓存行为只会应用到这个方法上。如果注解放在类级别上的话，name缓存行为就会应用到所有类的所有方法上。

## 填充缓存

@Cacheable和@CachePut注解都可以填充缓存。它们有一个共有的属性。

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| value | String[] | 要使用的缓存名字 |
| condition | String | SpEL表达式，如果值为false,不会在方法上使用缓存 |
| key | String | SpEL表达式，用来计算自定义缓存 |
| unless | String | SpEL表达式，如果值为true,不会在方法上使用缓存 |

注意：@Cacheable注解添加到接口上，会被所有的实现继承。

### 将值放在缓存中

|  |
| --- |
| **value属性是必须指定的**，其表示当前方法的返回值是会被缓存在哪个Cache上的，对应Cache的名称。其可以是一个Cache也可以是多个Cache，当需要指定多个Cache时其是一个数组。  @Cacheable("cache1")**//Cache放在cache1缓存中**  public User find(Integer id) {  ...  return user;  }  @Cacheable({"cache1", "cache2"})**//Cache放生在cache1和cache2缓存中**  public User find(Integer id) {  ...  returnnull;  }  **注意**：一般在save方法中使用@CachePut(保存的时候，总是进行缓存)。**默认缓存的key是缓存名+方法的参数（defaultCache\_8）**。 |

### 自定义缓存key

|  |  |
| --- | --- |
| 表达式 | 描述 |
| #root.args | 传递给缓存方法的参数，形式为数组 |
| #root.caches | 该方法执行时所对应的缓存，形式为数组 |
| #root.target | 目标对象 |
| #root.targetClass | 目标对象类，是#root.target.class的简写形式 |
| #root.method | 缓存方法 |
| #root.methodName | 缓存方法的名字，是#root.method.name的简写形式 |
| #result | 方法调用的返回值，不能用在@Cacheable注解上 |
| #Argument | 任意方法参数名(如#argName)或参数索引(如#a0或#p0) |

当我们要使用root对象的属性作为key时我们也可以将"#root"省略，因为Spring默认使用的就是root对象的属性。自定缓存例子：

|  |
| --- |
|  |

### 条件化缓存key

condition和unless提供了条件化缓存。

**condition**(SpEL表达式)：如果值为false,不会在方法上使用缓存。

**unless**(SpEL表达式)：如果值为true,不会在方法上使用缓存。

|  |
| --- |
| #表示只有当user的id为偶数时才会进行缓存。  @Cacheable(value={"users"}, key="#user.id", condition="#user.id%2==0")  public User find(User user) {  System.out.println("find user by user " + user);  return user;  } |

## 移除缓存

@CacheEvict：表明Spring应该在缓存中清除一个或者多个条目。当实体被更新或者删除的时候，我们可以采用该注解来清除缓存。和@Cacheable和@  
CachePut不同，@CacheEvict能够应用在返回值为void的方法上。@CacheEvict注解属性，指定了哪些缓存条目应该被移除。

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| value | String[] | 使用缓存的名称 |
| key | String | SpEl表达式，计算自定义缓存key |
| condition | String | SpEL表达式，如果值为false,不会在方法上使用缓存 |
| allEntries | boolean | 如果为true, 特定的缓存的所有条目都移除 |
| beforeInvocation | boolean | 如果为true，在调用方法之前移除条目。如果为false，在方法成功调用之后移除。 |

## @Caching注解

Caching注解可以让我们在一个方法或者类上同时指定多个Spring Cache相关的注解。其拥有三个属性：cacheable、put和evict，分别用于指定@Cacheable、@CachePut和@CacheEvict。

|  |
| --- |
| @Caching(cacheable = @Cacheable("users"), evict = { @CacheEvict("cache2"),  @CacheEvict(value = "cache3", allEntries = true) })  public User find(Integer id) {  ...  return user;  } |