Ehcache

1. 引入依赖包

|  |
| --- |
| <!-- 缓存 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-cache</artifactId>  </dependency>  <!--Ehcache包 -->  <dependency>  <groupId>net.sf.ehcache</groupId>  <artifactId>ehcache</artifactId>  <version>2.10.2</version>  </dependency> |

1. 在src/main/resources添加配置文件ehcache.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:noNamespaceSchemaLocation="http://ehcache.org/ehcache.xsd"  updateCheck="false">  <!-- 磁盘缓存位置 -->  <diskStore path="java.io.tmpdir/ehcache" />  <!-- 默认缓存 -->  <defaultCache  eternal="false"  timeToIdleSeconds="200"  timeToLiveSeconds="200"  maxEntriesLocalDisk="10000000"  diskExpiryThreadIntervalSeconds="120"  memoryStoreEvictionPolicy="LRU"/>  <!-- 项目相关开始 -->  <cache name="project"  maxEntriesLocalHeap="0"/>  <!-- 项目相关结束 -->  <!-- eternal：true表示对象永不过期，此时会忽略timeToIdleSeconds和timeToLiveSeconds属性，默认为false -->  <!-- maxEntriesLocalHeap：堆内存中最大缓存对象数，0没有限制 -->  <!-- timeToIdleSeconds： 设定允许对象处于空闲状态的最长时间，以秒为单位。当对象自从最近一次被访问后，  如果处于空闲状态的时间超过了timeToIdleSeconds属性值，这个对象就会过期，EHCache将把它从缓存中清空。  只有当eternal属性为false，该属性才有效。如果该属性值为0，则表示对象可以无限期地处于空闲状态 -->  </ehcache>  配置说明：  diskStore ： ehcache支持内存和磁盘两种存储  path ：指定磁盘存储的位置  defaultCache ： 默认的缓存  maxEntriesLocalHeap=”10000”  eternal=”false”  timeToIdleSeconds=”120”  timeToLiveSeconds=”120”  maxEntriesLocalDisk=”10000000”  diskExpiryThreadIntervalSeconds=”120”  memoryStoreEvictionPolicy=”LRU”  cache ：自定的缓存，当自定的配置不满足实际情况时可以通过自定义（可以包含多个cache节点）  name : 缓存的名称，可以通过指定名称获取指定的某个Cache对象  maxElementsInMemory ：内存中允许存储的最大的元素个数，0代表无限个  clearOnFlush：内存数量最大时是否清除。  eternal ：设置缓存中对象是否为永久的，如果是，超时设置将被忽略，对象从不过期。根据存储数据的不同，例如一些静态不变的数据如省市区等可以设置为永不过时  timeToIdleSeconds ： 设置对象在失效前的允许闲置时间（单位：秒）。仅当eternal=false对象不是永久有效时使用，可选属性，默认值是0，也就是可闲置时间无穷大。  timeToLiveSeconds ：缓存数据的生存时间（TTL），也就是一个元素从构建到消亡的最大时间间隔值，这只能在元素不是永久驻留时有效，如果该值是0就意味着元素可以停顿无穷长的时间。  overflowToDisk ：内存不足时，是否启用磁盘缓存。  maxEntriesLocalDisk：当内存中对象数量达到maxElementsInMemory时，Ehcache将会对象写到磁盘中。  maxElementsOnDisk：硬盘最大缓存个数。  diskSpoolBufferSizeMB：这个参数设置DiskStore（磁盘缓存）的缓存区大小。默认是30MB。每个Cache都应该有自己的一个缓冲区。  diskPersistent：是否在VM重启时存储硬盘的缓存数据。默认值是false。  diskExpiryThreadIntervalSeconds：磁盘失效线程运行时间间隔，默认是120秒。 |

1. 在application.yml添加配置

|  |
| --- |
| spring:  profiles:  active: test  # 缓存配置  cache:  ehcache:  config: ehcache.xml |

1. 添加启动注解

|  |
| --- |
| @EnableCaching  public class Application extends SpringBootServletInitializer {  public static void main(String[] args){  SpringApplication.run(Application.class, args);  }  } |

1. 注解基本使用方法

|  |
| --- |
| Spring对缓存的支持类似于对事务的支持。  首先使用注解标记方法，相当于定义了切点，然后使用Aop技术在这个方法的调用前、调用后获取方法的入参和返回值，进而实现了缓存的逻辑。   * @Cacheable   表明所修饰的方法是可以缓存的：当第一次调用这个方法时，它的结果会被缓存下来，在缓存的有效时间内，以后访问这个方法都直接返回缓存结果，不再执行方法中的代码段。  这个注解可以用condition属性来设置条件，如果不满足条件，就不使用缓存能力，直接执行方法。  可以使用key属性来指定key的生成规则。  @Cacheable 支持如下几个参数：   * value：缓存位置名称，不能为空，如果使用EHCache，就是ehcache.xml中声明的cache的name, 指明将值缓存到哪个Cache中 * key：缓存的key，默认为空，既表示使用方法的参数类型及参数值作为key，支持SpEL，如果要引用参数值使用井号加参数名，如：#userId，   一般来说，我们的更新操作只需要刷新缓存中某一个值，所以定义缓存的key值的方式就很重要，最好是能够唯一，因为这样可以准确的清除掉特定的缓存，而不会影响到其它缓存值 ，  本例子中使用实体加冒号再加ID组合成键的名称，如”user:1”、”order:223123”等   * condition：触发条件，只有满足条件的情况才会加入缓存，默认为空，既表示全部都加入缓存，支持SpEL   // 将缓存保存到名称为UserCache中，键为"user:"字符串加上userId值，如 'user:1'  @Cacheable(value="UserCache", key="'user:' + #userId")  public User findById(String userId) {  return (User) new User("1", "mengdee");  }  // 将缓存保存进UserCache中，并当参数userId的长度小于12时才保存进缓存，默认使用参数值及类型作为缓存的key  // 保存缓存需要指定key，value， value的数据类型，不指定key默认和参数名一样如："1"  @Cacheable(value="UserCache", condition="#userId.length() < 12")  public boolean isReserved(String userId) {  System.out.println("UserCache:"+userId);  return false;  }   * @CachePut   与@Cacheable不同，@CachePut不仅会缓存方法的结果，还会执行方法的代码段。它支持的属性和用法都与@Cacheable一致。   * @CacheEvict   与@Cacheable功能相反，@CacheEvict表明所修饰的方法是用来删除失效或无用的缓存数据。  @CacheEvict 支持如下几个参数：   * + value：缓存位置名称，不能为空，同上   + key：缓存的key，默认为空，同上   + condition：触发条件，只有满足条件的情况才会清除缓存，默认为空，支持SpEL   + allEntries：true表示清除value中的全部缓存，默认为false   //清除掉UserCache中某个指定key的缓存  @CacheEvict(value="UserCache",key="'user:' + #userId")  public void removeUser(User user) {  System.out.println("UserCache"+user.getUserId());  }  //清除掉UserCache中全部的缓存  @CacheEvict(value="UserCache", allEntries=true)  public final void setReservedUsers(String[] reservedUsers) {  System.out.println("UserCache deleteall");  } |

1. Springboot+ehcache复杂查询

|  |
| --- |
| public Result getPage(@RequestParam(required = false,defaultValue = "1") Integer page,  @RequestParam(required = false,defaultValue = "5") Integer rows,  @RequestParam(defaultValue = "") String key,  @RequestParam(defaultValue = "") Integer settlementYear,  @RequestParam(defaultValue = "") Boolean carryOver,  @RequestParam(defaultValue = "") Boolean waitMoney) {  try {  Page<Project> pageBean = projectService.findSettlementYear(page, rows, key, settlementYear,carryOver,waitMoney);  @Cacheable(value = "projects",key = "#page+#rows+#key+#settlementYear+#carryOver+#waitMoney")  public Page<Project> findSettlementYear(Integer page, Integer rows,  String key, Integer settlementYear,  Boolean carryOver,Boolean waitMoney) {  复杂查询时缓存不能有null，所以要使用@RequestParam(defaultValue = "")，不然会报错 |