## EhCache

## **EhCache简介**

EhCache是一个纯Java的进程内缓存框架，具有快速、精干等优点，是Hibernate中默认的CacheProvider。Ehcache是一种广泛使用的开源Java分布式缓存，其主要面向通用缓存，主要应用于Java轻量级容器。其优点主要为：

1. 快速、简单；
2. 多种缓存策略；
3. 缓存数据有两级：内存和磁盘，因此无需担心容量问题；
4. 缓存数据会在虚拟机重启的过程中写入磁盘；
5. 可以通过RMI、可插入API等方式进行分布式缓存；
6. 具有缓存和缓存管理器的侦听接口；
7. 支持多缓存管理器实例，以及一个实例的多个缓存区域；
8. 提供Hibernate的缓存实现。

其缺点主要为：

1. 使用磁盘Cache的时候非常占用磁盘空间。DiskCache算法简单、效率非常高。它只是对元素直接追加存储，在很频繁的应用中使用DiskCache，磁盘很快会满；
2. 不能保证数据的安全：服务器突然挂掉时可能会产生冲突，EhCache解决方法是如果文件冲突则重建cache，这对于数据保存不利。

## **Maven中Spring整合EhCache步骤为：**

1. pom文件中引入依赖：

***<!--ehcache -->***

***<dependency>***

***<groupId>net.sf.ehcache</groupId>***

***<artifactId>ehcache</artifactId>***

***<version>2.10.0</version>***

***</dependency>***

***<!--ehcache依赖slf4j-->***

***<dependency>***

***<groupId>org.slf4j</groupId>***

***<artifactId>slf4j-api</artifactId>***

***<version>1.7.18</version>***

***</dependency>***

***<dependency>***

***<groupId>org.slf4j</groupId>***

***<artifactId>slf4j-log4j12</artifactId>***

***<version>1.7.18</version>***

***</dependency>***

***<!--slf4j依赖log4j-->***

***<dependency>***

***<groupId>org.apache.logging.log4j</groupId>***

***<artifactId>log4j-core</artifactId>***

***<version>2.5</version>***

***</dependency>***

1. 添加ehcache配置文件ehcache.xml：

***<?xml version="1.0" encoding="UTF-8"?>***

***<ehcache>***

***<!-- 指定一个文件目录，当EhCache把缓存数据写到硬盘上时，将把数据写到这个文件目录下 -->***

***<diskStore path="java.io.tmpdir"/>***

***<!-- 设定默认的缓存数据过期策略 -->***

***<defaultCache***

***maxElementsInMemory="10000"***

***eternal="false"***

***overflowToDisk="true"***

***timeToIdleSeconds="10"***

***timeToLiveSeconds="20"***

***diskPersistent="false"***

***diskExpiryThreadIntervalSeconds="120"/>***

***<cache name="MyCache"***

***maxElementsInMemory="1000"***

***eternal="false"***

***overflowToDisk="true"***

***timeToIdleSeconds="10"***

***timeToLiveSeconds="20"/>***

***<cache name="AnotherCache"***

***maxElementsInMemory="1000"***

***.../>***

***</ehcache>***

其中Cache标签的属性含义为：

* name：缓存名称；
* maxElementsInMemory：内存中最大缓存对象数；
* maxElementsOnDisk：硬盘中最大缓存对象数，0表示无穷大；
* eternal：true表示对象永不过期，此时会忽略timeToIdleSeconds和timeToLiveSeconds属性，默认为false；
* overflowToDisk：true表示当内存缓存的对象数目达到了maxElementsInMemory界限后，会把溢出的对象写到硬盘缓存中。注意：如果缓存的对象要写入到硬盘中的话，则该对象必须实现了Serializable接口才行；
* diskSpoolBufferSizeMB：磁盘缓存区大小，默认为30MB，每个Cache都应该有自己的一个缓存区；
* diskPersistent：是否持久化磁盘缓存。当这个属性的值为true时，系统在初始化时会在磁盘中查找文件名为cache名称、后缀名为index的文件，找到后会把cache加载到内存。要想把cache真正持久化到磁盘，写程序时注意执行net.sf.ehcache.Cache.put(Element element)后要调用flush()方法；
* diskExpiryThreadIntervalSeconds：磁盘失效线程运行时间间隔，默认为120秒；
* 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；
* timeToLiveSeconds：设定对象允许存在于缓存中的最长时间，以秒为单位。当对象自从被存放到缓存中后，如果处于缓存中的时间超过了timeToLiveSeconds属性值，这个对象就会过期，EHCache将把它从缓存中清除。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存在于缓存中。timeToLiveSeconds必须大于timeToIdleSeconds属性，才有意义
* memoryStoreEvictionPolicy：当达到maxElementsInMemory限制时，Ehcache将会根据指定的策略去清理内存。可选策略有：LRU（最近最少使用，默认策略）、FIFO（先进先出）、LFU（最少访问次数）。

1. spring配置文件中加入：

*<beans ...*

***xmlns:cache="http://www.springframework.org/schema/cache"***

*xsi:schemaLocation="...*

***http://www.springframework.org/schema/cache***

***http://www.springframework.org/schema/cache/spring-cache-3.1.xsd***

*">*

*...*

***<!-- ehcache缓存配置 -->***

***<bean id="ehcache" class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean">***

***<property name="configLocation" value="classpath:ehcache.xml"></property>***

***</bean>***

***<bean id="cacheManager" class="org.springframework.cache.ehcache.EhCacheCacheManager">***

***<property name="cacheManager" ref="ehcache"></property>***

***</bean>***

***<cache:annotation-driven cache-manager="cacheManager" />***

</beans>

1. 代码中加入：

*...*

***import org.springframework.cache.Cache;***

***import org.springframework.cache.ehcache.EhCacheCacheManager;***

*@Service*

*public class LampServiceImpl implements LampService{*

***@Resource***

***EhCacheCacheManager cacheManager;***

*@Override*

***//value值要和ehcache配置文件ehcache.xml中的Cache名称对应***

***//@Cacheable：方法调用后会将其返回值缓存起来，以保证下次利用同样的参数来执行该方法时可以直接从缓存中获取结果，而不需要再次执行该方法。需要注意的是当一个支持缓存的方法在对象内部被调用时是不会触发缓存功能的***

***@Cacheable(value="MyCache", key="#lamp.getLampId()")***

*public boolean addLamp(Lamp lamp) {*

*...*

*}*

*@Override*

***//@CachePut：方法在每次执行前都会检查Cache中是否存在相同key的缓存元素，如果存在就不再执行该方法，而是直接从缓存中获取结果进行返回，否则才会执行并将返回结果存入指定的缓存中。与@Cacheable不同的是，@CachePut在执行前不会去检查缓存中是否存在之前执行过的结果，而是每次都会执行该方法，并将执行结果以键值对的形式存入指定的缓存中***

***@CachePut(value="MyCache", key="#lamp.getLampId()")***

*public boolean updateLamp(Lamp lamp) {*

*...*

*}*

*@Override*

***//@CacheEvict：方法执行后清除缓存元素***

***@CacheEvict(value="MyCache", allEntries=true)***

*public boolean deleteLamp(List<Long> lampIds) {*

*...*

*}*

*@Override*

*public LampState getLampState(long lampId) {*

***Cache cache = cacheManager.getCache("MyCache");***

*LampState lampState =* ***cache.get(lampId, LampState.class);***

*...*

*}*

注解相关概念：

@Cacheable、@CachePut和@CacheEvict可以标记在一个方法上，也可以标记在一个类上。当标记在一个方法上时表示该方法是支持相关缓存操作的，当标记在一个类上时则表示该类所有的方法都是支持相关缓存操作的。

@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相关注解，其拥有三个属性：cacheable、put和evict，分别用于指定@Cacheable、@CachePut和@CacheEvict。例如：

***@Caching(***

***cacheable = @Cacheable("users"),***

***evict = {***

***@CacheEvict("cache2"),***

***@CacheEvict(value = "cache3", allEntries = true)***

***}***

***)***

@Cacheable和@Cacheable可以指定三个属性，value、key和condition，而@CacheEvict可以指定的属性有value、key、condition、allEntries和beforeInvocation。属性作用如下：

* value：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，可以是一个Cache，也可以是多个Cache。
* key：缓存的Key，key的生成策略有两种：一种是默认策略，一种是自定义策略。当我们没有指定该属性时，Spring将使用默认策略生成key。自定义策略通过Spring的EL表达式来指定key，例如使用方法参数时我们可以直接使用“#参数名”或者“#p参数index”。默认的key生成策略是通过KeyGenerator生成的，其默认策略如下：

1. 如果方法没有参数，则使用0作为key；
2. 如果只有一个参数的话则使用该参数作为key；
3. 如果参数多余一个的话则使用所有参数的hashCode作为key

* condition：有的时候我们可能并不希望缓存一个方法所有的返回结果，通过condition属性可以实现这一功能。condition属性可以通过SpringEL表达式来指定，默认为空表示将缓存所有结果。例如condition="#id%2==0"表示只有当id为偶数时才进行缓存。
* allEntries：boolean型，表示是否需要清除缓存中的所有元素。默认为false，表示不需要。当指定了allEntries为true时，将清除Cache中所有元素。

beforeInvocation：清除操作默认是在方法成功执行之后触发的，方法如果因为抛出异常而未能成功返回时不会触发清除操作。使用beforeInvocation可以改变触发清除操作的时间，其值为true时Spring会在调用该方法之前清除缓存。