SpringBoot整合Ehcache3的实现步骤

作者：有故事的人 来源：范文网 www.wtabcd.cn/fanwen/

本文原地址：https://www.wtabcd.cn/fanwen/zuowen/7982fe419162e2e86595ffc6dd819673.html

前言

公司部门老项目要迁移升级java版本，需要进行缓存相关操作，原框架未支持这部分，经过调研java相关缓存方案大致分为ehcache和redis两种，redis的value最大值为500mb且超过1mb会对存取有性能影响，业务系统需要支持列表查询缓存就不可避免的涉及到大量的数据存取过滤，ehcache支持内存+磁盘缓存不用担心缓存容量问题，所以框架初步版本决定集成ehcache3，设计流程结构如下图所示

缓存配置

maven引用

<dependency> word下划线怎么打 <groupid>org.springframework.boot</gr
oupid> <artifactid>spring-boot-starter-cache</artifactid> </dependency>
<dependency> <groupid>org.ehcache</groupid>
<artifactid>ehcache</artifactid> </dependency>

个性化配置

#缓存配置 cache: ehcache: heap: 1000 offheap: 100 disk: 5
00 diskdir: tempfiles/cache/

@component@configurationproperties("frmae.cache.ehcache")public class ehcacheconfiguration { /** * ehcache
heap大小 * jvm内存中缓存的key数量 */ private int heap; /** * ehcache offheap大小 *
堆外内存大小, 单位: mb */ private int offheap; /** * 磁盘持久化目录 */ private strin
g diskdir; /** * ehcache disk * 持久化到磁盘的大小, 单位: mb * diskdir有效时才生效 */
private int disk; public ehcacheconfiguration(){ heap = 1000; offheap = 100;
disk = 500; diskdir = "tempfiles/cache/"; }}

代码注入配置

因为springboot默认缓存优先注入redis配置，所以需要手动声明bean进行注入，同时ehcache的value值必须支持序列化接口，不能使用object代替，这里声明一个缓存基类，所有缓存value对象必须继承该类

public class basesystemobject implements serializable { }

```
@configuration@enablecachingpublic class ehcacheconfig { @autowired private ehcacheconfiguration ehcache
configuration; @autowired private applicationcontext context; @bean(name = "ehcachemanager") publ
ic cachemanager getcachemanager() { //资源池生成器配置持久化 resourcepoolsbuilder resourcepo
olsbuilder = resourcepoolsbuilder.newresourcepoolsbuilder() // 堆内缓存大小
.heap(ehcacheconfiguration.getheap(), entryunit.entries) // 堆外缓存大小
.offheap(ehcacheconfiguration.getoffheap(), memoryunit.mb) // 文件缓存大小 .d
isk(ehcacheconfiguration.getdisk(), memoryunit.mb); //生成配置 expirypolicy expirypolicy = expiry
policybuilder.noexpiration(); cacheconfiguration config = cacheconfigurationbuilder.newcacheconfigurationbuil
der(string.class, basesystemobject.class, resourcepoolsbuilder) //设置永不过期 .with
expiry(expirypolicy) .build(); cachemanagerbuilder cachemanagerbuilder = cachemanagerbui
lder.newcachemanagerbuilder() .with(cachemanagerbuilder.persistence(ehcacheconfiguration.getdisk
dir())); return cachemanagerbuilder.build(true); }}
```

缓存操作

缓存预热

针对缓存框架选择的双写策略，即数据库和缓存同时写入，所以在系统启动时需要预先将数据库数据加载到缓存中
针对单表声明自定义注解，个性化缓存定义自定义接口

```
@target({elementtype.type})@retention(retentionpolicy.runtime)@documented@inheritedpublic @interface hpcache {}
```

```
public interface ihpcacheinitservice { string getcachename(); void initcache();}
```

系统初始化时同步进行缓存初始化，扫描注解实体类与接口实现bean

```
@async public void initcache(class runtimeclass, list<string> extrapackagenamelist) { list<class<?>> c
acheentitylist = new arraylist<>(); if (!runtimeclass.getpackage().getname().equals(application.class.getpackage(
).getname())) { cacheentitylist.addall(scanutil.getallclassbypackagename_annotation(runtimeclass.getpackage(),
hpcache.class)); } for (string packagename : extrapackagenamelist) { cacheentitylist
.addall(scanutil.getallclassbypackagename_annotation(packagename, hpcache.class)); } for (class clazz
: cacheentitylist) { tablename tablename = (tablename) clazz.getannotation(tablename.class);
list<linkedhashmap<string, object>> resultlist = commondto.selectlist(tablename.value(), "*", "1=1", "", n
ew hashmap<>(), false); for (linkedhashmap<string, object> map : resultlist) {
cache cache = cachemanager.getcache(clazz.getname(), string.class, basesystemobject.class); string
unitguid = convertop.convert2string(map.get("unitguid")); try { object ob
j = clazz.newinstance(); obj = convertop.convertlinkhashmaptobean(map, obj);
cache.put(unitguid, obj); } catch (exception e) { e.printstacktr
ace(); } } } //自定义缓存 map<string, ihpcacheinitservice> r
es = context.getbeansoftype(ihpcacheinitservice.class); for (map.entry en : res.entryset()) { i
hpcacheinitservice service = (ihpcacheinitservice) en.getvalue(); service.initcache(); } sy
stem.out.println("缓存初始化完毕"); }
```

需要注意，在ehcacheconfig配置类中需要进行缓存名称的提前注册，否则会导致操作缓存时空指针异常

```
map<string, object> annotatedbeans = context.getbeanswithannotation(springbootapplication.class); class
runtimeclass = annotatedbeans.values().toarray()[0].getclass(); //do，dao扫描 list<string> extrapack
agenamelist = new arraylist<string>(); extrapackagenamelist.add(application.class.getpackage().getname());
list<class<?>> cacheentitylist = new arraylist<>(); if (!runtimeclass.getpackage().getname().equals(applic
ation.class.getpackage().getname())) { cacheentitylist.addall(scanutil.getallclassbypackagename_annotation(runti
meclass.getpackage(), hpcache.class)); } for (string packagename : extrapackagenamelist) {
cacheentitylist.addall(scanutil.getallclassbypackagename_annotation(packagename, hpcache.class)); }
```

for (class clazz : cacheentitylist) { cachemanagerbuilder = cachemanagerbuilder.withcache(clazz.ge tname(), config); } //自定义缓存 map<string, ihpcacheinitservice> res = context.getbeans oftype(ihpcacheinitservice.class); for (map.entry en :res.entryset()) { ihpcacheinitservice service = (ihpcacheinitservice)en.getvalue(); cachemanagerbuilder = cachemanagerbuilder.withcache(service.getc achename(), config); }

更新操作

手动获取ehcache的bean对象，调用put，repalce，delete方法进行操作

private cachemanager cachemanager = (cachemanager) springbootbeanutil.getbean("ehcachemanager"); publi c void executeupdateoperation(string cachename, string key, basesystemobject value) { cache cache = cachemanager.getcache(cache尽情的近义词是什么name, string.class, basesystemobject.class); if (cache.cont ainskey(key)) { cache.replace(key, value); } else { cache.put(key, value); } } public void executedeleteoperation(string cachename, string key) { cache cache = cach emanager.getcache(cachename, string.class, basesystemobject.class); cache.remove(key); }

查询操作

缓存存储单表以主键—obj马拉松训练方法ect形式存储，个性化缓存为key-object形式存储，单条记录可以通过getcache方法查询，列表查询需要取出整个缓存按条件进行过滤

public object getcache(string cachename, string key){ cache cache = cachemanager.getcache(cachename , string.class, basesystemobject.class); return cache.get(key); } public list<object> getallcache(string cachename){ list result = new arraylist<>(); cache cache = cachemanager.getcache(cachename, string.class, basesystemobject.class); iterator iter = cache.iterator(); while (iter.hasnext()) { cache.entry entry = (cache.entry) iter.next(); result.add(entry.getvalue()); } r eturn result; }

缓存与数据库数据一致性

数据库数据操作与缓存操作顺序为先操作数据后操作缓存，在开启数据库事务的情况下针对单条数据单次操作是没有问题的，如果是组合操作一旦数据库操作发生异常回滚，缓存并没有回滚就会导致数据的不一致，比如执行顺序为dbop1=》cacheop1=》dbop2=》cacheop2，dbop2异常，cacheop1的操作已经更改了缓存
这里选择的方案是在数据库全部执行完毕后统一操作缓存，这个方案有一个缺点是如果缓存操作发生异常还是会出现上述问题，实际过程中缓存只是对内存的操作异常概率较小，对缓存操作持乐观状态，同时我们提供手动重置缓存的功能，算是一个折中方案，下面概述该方案的一个实现

声明自定义缓存事务注解

@target({elementtype.method})@retention(retentionpolicy.runtime)@documented@inheritedpublic @interface cachetransact ional {}

声明切面监听，在标记了cachetransactional注解的方法执行前进行redis标识，统一执行完方法体后执行缓存操作
将缓存操作以线程id区分放入待执行队列中序列化到redis，提供方法统一操作

public class cacheexecutemodel implements serializable { private string obejctclazzname; private string cachename; private string key; private basesystemobject value; private string executetype;}private cach emanager cachemanager = (cachemanager) springbootbeanutil.getbean("ehcachemanager"); @autowired private redisutil redisutil; public void putcacheintotransition(){ string threadid = thread.currentthread().getn ame(); system.out.println("init threadid:"+threadid); cacheexecutemodel cacheexecutemodel = new

cacheexecutemodel(); cacheexecutemodel.setexecutetype("option"); redisutil.redistemplatesetforcollection(thr
eadid,cacheexecutemodel, globalenum.redisdbnum.cache.get_value()); redisutil.setexpire(threadid,5, timeunit.min
utes, globalenum.redisdbnum.cache.get_value()); } public void putcache(string cachename, string key, bas
esystemobject value) { if(checkcacheoptinionintransition()){ string threadid = thread.currentth
read().getname(); cacheexecutemodel cacheexecutemodel = new cacheexecutemodel("update", cachena
me, key, value.getclass().getname(),value); redisutil.redistemplatesetforcollection(threadid,cacheexecutemodel,
globalenum.redisdbnum.cache.华语辩论get_value()); redisutil.setexpire(threadid,5, timeunit.minutes, glob
alenum.redisdbnum.cache.get_value()); }else{ executeupdateoperation(cachename,key,value);
} } class的复数public void deletecache(string cachename, string key) { if(checkcacheoptinionintra
nsition()){ string threadid = thread.currentthread().getname(); cacheexecutemodel cacheex
ecutemodel = new cacheexecutemodel("delete", cachename, key); redisutil.redistemplatesetforcollection(
threadid,cacheexecutemodel, globalenum.redisdbnum.cache.get_value()); redisutil.setexpire(threadid,5, timeu
nit.minutes, globalenum.redisdbnum.cache.get_value()); }else{ executedeleteoperation(cachename,key
); } } public void executeoperation(){ string threadid = thread.currentthread().getname();
if(checkcacheoptinionintransition()){ list<linkedhashmap> executelist = redisutil.redistemplateget
forcollectionall(threadid, globalenum.redisdbnum.cache.get_value()); for (linkedhashmap obj:executelist)
{ string executetype = convertop.convert2string(obj.get("executetype")); if(executet
ype.contains("option")){ continue; } string obejctclazzname
= convertop.convert2string(obj.get("obejctclazzname")); string cachename = convertop.convert2strin
g(obj.get("cachename")); string key = convertop.convert2string(obj.get("key")); lin
kedhashmap valuemap = (linkedhashmap)obj.get("value"); string valuemapjson = json.tojsonstri
ng(valuemap); try{ object valueinstance = json.parseobject(valuemapjson,class
.forname(obejctclazzname)); if(executetype.equals("update")){ executeup
dateoperation(cachename,key,(basesystemobject)valueinstance); }else if(executetype.equals("delete"))
{ executedeleteoperation(cachename,key); } }catch
(exception e){ e.printstacktrace(); } } redisutil.rediste
mplateremove(threadid,globalenum.redisdbnum.cache.get_value()); } } public boolean checkcacheoptinio
nintransition(){ string threadid = thread.currentthread().getname(); system.out.println("check threadi
d:"+threadid); return redisutil.isvalid(threadid, globalenum.redisdbnum.cache.get_value()); }

到此这篇关于springboot整合ehcache3的实现步骤的文章就介绍到这了,更多相关springboot整合ehcache3内容请搜索ww
w.887551.com以前的文章或继续浏览下面的相关文章希望大家以后多多支持www.887551.com！