# CACHE使用说明书

## 项目中的引用

### 1.1.maven构建

项目是使用maven构建的，则只需要在pom文件中引入如下依赖

<dependency>  
 <groupId>com.odianyun.cache</groupId>  
 <artifactId>ocache</artifactId>  
 <version>1.0-SNAPSHOT</version>  
</dependency>

<dependency>  
 <groupId>javax.cache</groupId>  
 <artifactId>cache-api</artifactId>  
 <version>1.0.0</version>  
</dependency>

<dependency>  
 <groupId>net.sf.ehcache</groupId>  
 <artifactId>ehcache</artifactId>  
 <version>2.8.1</version>  
</dependency>

版本按当前有效版本为准

## 配置信息

使用ocache中的缓存需要引入缓存的配置文件和spring的配置文件。需要配置全局文件路径 -Dglobal.config.path = disk:/path1/path2 ,相应的缓存配置文件放在全局路径下，如memcache.xml

### Memcached配置

配置ocache中的memcached缓存

#### 缓存文件配置

以下是memcached.xml的配置，以项目odemo为例

**Pool id：**应用模块使用缓存资源的id（格式是字符串）（该id应有应用开发人员给出，且该代号体现缓存的用途，一般包含poolname等信息）。

其他参数可以根据需要修改

compressEnable是否压缩value, compressThreshold是当value值大于多少byte时压缩

maintSleep为管理这个pool的间隔，单位是毫秒，这里就是说30秒会检查一次是否有空闲的连接需要释放,其他时间单位也是毫秒

1.memcache.xml文件中，过期时间的单位被重置为分钟而非秒；

2.下面的过期时间为720分钟，如果需要可以根据自己的项目修改，当过期时间为设置为0或小于0时，被设置为30分钟；

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <config>  <pool id="odemo-common"  initConn="1"  minConn="1"  maxConn="8"  maintSleep="30000"  socketTO="3000"  socketConnectTO="3000"  defaultExpiryMinutes="3"  compressEnable="true"  compressThreshold="4096">  <servers>  <server>127.0.0.1:11211</server>  </servers>  </pool>  </config> |

配置文件详解

|  |
| --- |
| initConn 初始连接数  minConn 最小连接数  maxConn 最大连接数  maintSleep 单位:ms,后台线程管理SocketIO池的检查间隔时间,如果设置为0，则表明不需要后台线程维护SocketIO线程池，默认需要管理  failover 启用故障转移  socketTo 传输超时设置,单位ms  socketConnTo 连接超时设置,单位ms  defaultExpiryMinutes 默认超时,单位分钟  maxBusyTime multi超时,单位ms  compressEnable 是对数据否进行压缩  versionEnable 是对缓存的key加上版本号  poolVersion pool版本号  JDK 是否用原生序列化  compressThreshold= 4096 当数据超过该字节数值时，进行压缩  sendEmailInterval= 10 每隔10分钟发送消息  servers 该pool所包含memcached server,server格式为host:port, 如 127.0.0.1:11211 |

#### Spring文件配置

Memcache在spring文件中的配置

|  |  |
| --- | --- |
| <bean id="memcacheExtend" class="com.odianyun.cache.MemcacheExtend" factory-method="getInstance">  <constructor-arg>  <value>file:${global.config.path}/odemo/odemo-common/memcache.xml</value>  </constructor-arg> </bean>  <bean id="cache" class="com.odianyun.cache.CacheProxy" factory-method="getCacheProxy" factory-bean="memcacheExtend" >  <constructor-arg>  <value>oxx-common</value>  </constructor-arg> </bean> |  |

## 2.2 ehCache配置

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

<ehcache>

<!--timeToIdleSeconds 当缓存闲置n秒后销毁 -->

<!--timeToLiveSeconds 当缓存存活n秒后销毁 -->

<!--

缓存配置

name:缓存名称。

maxElementsInMemory：缓存最大个数。

eternal:对象是否永久有效，一但设置了，timeout将不起作用。

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：设置对象在失效前允许存活时间（单位：秒）。最大时间介于创建时间和失效时间之间。仅当eternal=false对象不是永久有效时使用，默认是0.，也就是对象存活时间无穷大。

overflowToDisk：当内存中对象数量达到maxElementsInMemory时，Ehcache将会对象写到磁盘中。

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。

maxElementsOnDisk：硬盘最大缓存个数。

diskPersistent：是否缓存虚拟机重启期数据 Whether the disk store persists between restarts of the Virtual Machine. The default value is false.

diskExpiryThreadIntervalSeconds：磁盘失效线程运行时间间隔，默认是120秒。

memoryStoreEvictionPolicy：当达到maxElementsInMemory限制时，Ehcache将会根据指定的策略去清理内存。默认策略是LRU（最近最少使用）。你可以设置为FIFO（先进先出）或是LFU（较少使用）。

clearOnFlush：内存数量最大时是否清除。

-->

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

<defaultCache

maxElementsInMemory="500"

eternal="false"

timeToIdleSeconds="300"

timeToLiveSeconds="1200"

overflowToDisk="true" />

<cache name="owms" maxElementsInMemory="150" eternal="false" timeToLiveSeconds="36000" timeToIdleSeconds="3600" overflowToDisk="true"/>

</ehcache>

## 3.代码示例

### 3.1．简略使用信息

Ocache使用示例，通用示例，简略使用信息。

|  |
| --- |
| 把dao里面的  protected CacheProxy cache; 删除掉  然后继承**extends** CacheFactory  //将value加入到以key为键的缓存中  cache.put(key,value); |

### 3.2．详细使用信息

|  |
| --- |
| @Repository  **public** **class** DocPoHeaderDAOImpl **extends** CacheFactory **implements** DocPoHeaderDAO{  public List selectByExampleWithoutBLOBs(McChanelClientExample example, boolean fromCache) throws SQLException {  if(fromCache){  String key\_cache = "DaoKey\_" + this.getClass().getName()+ "\_SelectByExampleWithoutBLOBs\_" +MD5Support.*MD5*(example.allValue2String());  List list\_cache = (List)cache.get(key\_cache) ;  if(list\_cache == null){  List list = sqlMapClient.queryForList("mc\_chanel\_client.abatorgenerated\_selectByExample", example);  list\_cache = list;  if (list != null && list.size() > 0) {  cache.put( key\_cache , list\_cache );  }   }   return list\_cache;   } else {  List list = sqlMapClient.queryForList("mc\_chanel\_client.abatorgenerated\_selectByExample", example);  return list;  } } |

# 缓存策略配置文件

cache.ehcaches=2,3,4

cache.memcaches=1,5,6

里面需配置当前企业所使用的缓存以逗号隔开;如果没有配置默认会使用ehcache缓存

@Service("cacheFactory")

**public** **class** CacheFactory {

@Resource(name="cacheContext")

**private** CacheContext cacheContext;

**public** **static** CacheStrategr *cache*;

**private** **static** Map<String, CacheStrategr> *cacheMap* = **new** ConcurrentHashMap<String, CacheStrategr>();// 缓存容器

**public** CacheStrategr initCache(String entGuid) **throws** Exception{

*cache*=*cacheMap*.get(entGuid);

**if**(*cache*==**null**){

*cache*=createFactory(entGuid);

*cacheMap*.put(entGuid, *cache*);

}

**return** *cache*;

}

**public** CacheStrategr createFactory(String entGuid) **throws** Exception{

Properties properties=PropertyUtils.*getProperty*(CacheConstant.*CACHE\_URL*);

String ehCaches=properties.getProperty(CacheConstant.*EHCACHES*);

String[] ehs=ehCaches.split(",");

**for**(**int** i=0;i<ehs.length;i++){ //根据公司ID循环匹配

**if**(entGuid.equals(ehs[i])){

*cache*=EhCacheComponent.*getInstance*(); //创建EhCache

}

}

String memCaches=properties.getProperty(CacheConstant.*MEMCACHES*);

String[] mms=memCaches.split(",");

**for**(**int** i=0;i<mms.length;i++){

**if**(entGuid.equals(mms[i])){

*cache*=(CacheStrategr) cacheContext.getServiceInstance(CacheConstant.*MEMCACHE*);

}

}

//如果没有配置，默认使用chCache

**return** *cache*==**null**?EhCacheComponent.*getInstance*():*cache*;

}

}