分布式(redis mongoDB memecached ehcache) ehcache

第1步：开启二级缓存

第2步：导入ehcache的jar包 2个

第3步：在config编写ehcache.xml文件

<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="../config/ehcache.xsd">

<!--diskStore：缓存数据持久化的目录 地址 -->

<diskStore path=".\develop\ehcache" />

<defaultCache

maxElementsInMemory="1000"

maxElementsOnDisk="10000000"

eternal="false"

overflowToDisk="true"

diskPersistent="true"

timeToIdleSeconds="120"

timeToLiveSeconds="120"

diskExpiryThreadIntervalSeconds="120"

memoryStoreEvictionPolicy="LRU">

</defaultCache>

</ehcache>

第4步：mapper.xml中（选配 ---局部配置）

<cache type="org.mybatis.caches.ehcache.EhcacheCache">

<property name="memoryStoreEvictionPolicy" value="FIFO"/>

<property name="maxElementsInMemory" value="1"/>

</cache>

ehcache.xml:配置详解

    name：指定cache的名称。

       maxEntriesLocalDisk：指定允许在硬盘上存放元素的最大数量，0表示不限制。这个属性我们也可以在运行期通过CacheConfiguration来

更改。

       maxEntriesLocalHeap：指定允许在内存中存放元素的最大数量，0表示不限制。这个属性也可以在运行期动态修改。

       maxEntriesInCache：指定缓存中允许存放元素的最大数量。这个属性也可以在运行期动态修改。但是这个属性只对Terracotta分布式缓

存有用。

       maxBytesLocalDisk：指定当前缓存能够使用的硬盘的最大字节数，其值可以是数字加单位，单位可以是K、M或者G，不区分大小写，

如：30G。当在CacheManager级别指定了该属性后，Cache级别也可以用百分比来表示，如：60%，表示最多使用

CacheManager级别指定硬盘容量的60%。该属性也可以在运行期指定。当指定了该属性后会隐式的使当前Cache的

overflowToDisk为true。

       maxBytesLocalHeap：指定当前缓存能够使用的堆内存的最大字节数，其值的设置规则跟maxBytesLocalDisk是一样的。

       maxBytesLocalOffHeap：指定当前Cache允许使用的非堆内存的最大字节数。当指定了该属性后，会使当前Cache的overflowToOffHeap

的值变为true，如果我们需要关闭overflowToOffHeap，那么我们需要显示的指定overflowToOffHeap的值为false。

       overflowToDisk：boolean类型，默认为false。当内存里面的缓存已经达到预设的上限时是否允许将按驱除策略驱除的元素保存在硬盘上，

默认是LRU（最近最少使用）。当指定为false的时候表示缓存信息不会保存到磁盘上，只会保存在内存中。该属性现在

已经废弃，推荐使用cache元素的子元素persistence来代替，如：<persistence strategy=”localTempSwap”/>。

       diskSpoolBufferSizeMB：当往磁盘上写入缓存信息时缓冲区的大小，单位是MB，默认是30。

       overflowToOffHeap：boolean类型，默认为false。表示是否允许Cache使用非堆内存进行存储，非堆内存是不受Java GC影响的。该属性

只对企业版Ehcache有用。

       copyOnRead：当指定该属性为true时，我们在从Cache中读数据时取到的是Cache中对应元素的一个copy副本，而不是对应的一个引 用。

默认为false。

       copyOnWrite：当指定该属性为true时，我们在往Cache中写入数据时用的是原对象的一个copy副本，而不是对应的一个引用。默认为 false。

       timeToIdleSeconds：单位是秒，表示一个元素所允许闲置的最大时间，也就是说一个元素在不被请求的情况下允许在缓存中待的最大时 间。

默认是0，表示不限制。

       timeToLiveSeconds：单位是秒，表示无论一个元素闲置与否，其允许在Cache中存在的最大时间。默认是0，表示不限制。

       eternal：boolean类型，表示是否永恒，默认为false。如果设为true，将忽略timeToIdleSeconds和timeToLiveSeconds，Cache内的元素永

远都不会过期，也就不会因为元素的过期而被清除了。

分布缓存

我们系统为了提高系统并发，性能、一般对系统进行分布式部署（集群部署方式）

不使用分布缓存，缓存的数据在各各服务单独存储，不方便系统开发。所以要使用分布式缓存对缓存数据进行集中管理。

mybatis无法实现分布式缓存，需要和其它分布式缓存框架进行整合。

整合方法(掌握)

mybatis提供了一个cache接口，如果要实现自己的缓存逻辑，实现cache接口开发即可。

mybatis和ehcache整合，mybatis和ehcache整合包中提供了一个cache接口的实现类。

package org.apache.ibatis.cache;
import java.util.concurrent.locks.ReadWriteLock;
/\*\*
\* SPI for cache providers.
\*
\* One instance of cache will be created for each namespace.
\*
\* The cache implementation must have a constructor that receives the cache id as an String parameter.
\*
\* MyBatis will pass the namespace as id to the constructor.
\*
\* <pre>
\* public MyCache(final String id) {
\* if (id == null) {
\* throw new IllegalArgumentException("Cache instances require an ID");
\* }
\* this.id = id;
\* initialize();
\* }
\* </pre>
\*
\* @author Clinton Begin
\*/
public interface Cache {
/\*\*
\* @return The identifier of this cache
\*/
String getId();
/\*\*
\* @param key Can be any object but usually it is a {@link CacheKey}
\* @param value The result of a select.
\*/
void putObject(Object key, Object value);
/\*\*
\* @param key The key
\* @return The object stored in the cache.
\*/
Object getObject(Object key);
/\*\*
\* Optional. It is not called by the core.
\*
\* @param key The key
\* @return The object that was removed
\*/
Object removeObject(Object key);
/\*\*
\* Clears this cache instance
\*/
void clear();
/\*\*
\* Optional. This method is not called by the core.
\*
\* @return The number of elements stored in the cache (not its capacity).
\*/
int getSize();
/\*\*
\* Optional. As of 3.2.6 this method is no longer called by the core.
\*
\* Any locking needed by the cache must be provided internally by the cache provider.
\*
\* @return A ReadWriteLock
\*/
ReadWriteLock getReadWriteLock();
}

mybatis默认实现cache类是：

package org.apache.ibatis.cache.impl;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.locks.ReadWriteLock;
import org.apache.ibatis.cache.Cache;
import org.apache.ibatis.cache.CacheException;
/\*\*
\* @author Clinton Begin
\*/
public class PerpetualCache implements Cache {
private String id;
private Map<Object, Object> cache = new HashMap<Object, Object>();
public PerpetualCache(String id) {
this.id = id;
}
public String getId() {
return id;
}
public int getSize() {
return cache.size();
}
public void putObject(Object key, Object value) {
cache.put(key, value);
}
public Object getObject(Object key) {
return cache.get(key);
}
public Object removeObject(Object key) {
return cache.remove(key);
}
public void clear() {
cache.clear();
}
public ReadWriteLock getReadWriteLock() {
return null;
}
public boolean equals(Object o) {
if (getId() == null) throw new CacheException("Cache instances require an ID.");
if (this == o) return true;
if (!(o instanceof Cache)) return false;
Cache otherCache = (Cache) o;
return getId().equals(otherCache.getId());
}
public int hashCode() {
if (getId() == null) throw new CacheException("Cache instances require an ID.");
return getId().hashCode();
}
}

整合ehcache

加入ehcache包

ehcache-core-2.6.5.jar

mybatis-ehcache-1.0.2.jar

配置mapper中cache中的type为ehcache对cache接口的实现类型

<!-- 开启本mapper的namespace下的二级缓存
type：指定cache接口的实现类的类型，mybatis默认使用PerpetualCache
要和ehcache整合，需要配置type为ehcache实现cache接口的类型
<cache />
-->
<cache type="org.mybatis.caches.ehcache.EhcacheCache"/>

加入ehcache的配置文件

在classpath下配置ehcache.xml

<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../config/ehcache.xsd">
<diskStore path="F:\develop\ehcache" />
<defaultCache
maxElementsInMemory="1000"
maxElementsOnDisk="10000000"
eternal="false"
overflowToDisk="false"
timeToIdleSeconds="120"
timeToLiveSeconds="120"
diskExpiryThreadIntervalSeconds="120"
memoryStoreEvictionPolicy="LRU">
</defaultCache>
</ehcache>