Atitit mybatis 二级缓存开启测试

[1. MyBatis的flushCache和useCache的使用 1](#_Toc22609)

[2. 通过打开debug sql日志看到是否启用 3](#_Toc31784)

[2.1. 、二级缓存  3](#_Toc11469)

[2.2. 1.5  二级应用场景 4](#_Toc10835)

[2.3. 1.6  二级缓存局限性 4](#_Toc6029)

# **MyBatis的flushCache和useCache的使用**

原创 2016年08月19日 11:17:07

* 标签：
* [mybatis](http://so.csdn.net/so/search/s.do?q=mybatis&t=blog" \t "https://blog.csdn.net/ssssny/article/details/_blank)
* 8884

（1）当为select语句时：

flushCache默认为false，表示任何时候语句被调用，都不会去清空本地缓存和二级缓存。

useCache默认为true，表示会将本条语句的结果进行二级缓存。

（2）当为insert、update、delete语句时：

flushCache默认为true，表示任何时候语句被调用，都会导致本地缓存和二级缓存被清空。

useCache属性在该情况下没有。

当为select语句的时候，如果没有去配置flushCache、useCache，那么默认是启用缓存的，所以，如果有必要，那么就需要人工修改配置，修改结果类似下面：

**[xml]** [view plain](https://blog.csdn.net/ssssny/article/details/52248960" \o "view plain" \t "https://blog.csdn.net/ssssny/article/details/_self) [copy](https://blog.csdn.net/ssssny/article/details/52248960" \o "copy" \t "https://blog.csdn.net/ssssny/article/details/_self)

1. **<select** id="save" parameterType="XX" flushCache="true" useCache="false"**>**
2. ……
3. **</select>**

update 的时候如果 flushCache="false"，则当你更新后，查询的数据数据还是老的数据。

单纯的 <cache /> 表示如下意思:  
  
1.所有在映射文件里的 select 语句都将被缓存。  
2.所有在映射文件里 insert,update 和 delete 语句会清空缓存。  
3.缓存使用“最近很少使用”算法来回收  
4.缓存不会被设定的时间所清空。  
5.每个缓存可以存储 1024 个列表或对象的引用（不管查询出来的结果是什么） 。  
6.缓存将作为“读/写”缓存，意味着获取的对象不是共享的且对调用者是安全的。不会有其它的调用者或线程潜在修改。  
缓存元素的所有特性都可以通过属性来修改。比如：

IMG_256 程序代码

<cache eviction="FIFO" flushInterval="60000" size="512" readOnly="true" />

eviction是缓存的淘汰算法，可选值有"LRU"、"FIFO"、"SOFT"、"WEAK"，缺省值是LRU

flashInterval指缓存过期时间，单位为毫秒，60000即为60秒，缺省值为空，即只要容量足够，永不过期

size指缓存多少个对象，默认值为1024

readOnly是否只读，如果为true，则所有相同的sql语句返回的是同一个对象（有助于提高性能，但并发操作同一条数据时，可能不安全），如果设置为false，则相同的sql，后面访问的是cache的clone副本。

# 通过打开debug sql日志看到是否启用

正常情况下，只有第一个发出sql。。。

如果没有看到 ，可能是sesseon没有sumibt ,,submit ha zo ok//

**private** **static** **void** testMybatisCache(String cfg) {

HashMap **map** = **new** HashMap();

Object **productId**=9999;

map.put("productId", productId);

Object **programId**=9999;

map.put("programId", programId);

Object **paramNo**=12;

map.put("paramNo", paramNo);

SqlSession **ss** = MybatisUtil.*getSqlSessionNew*(cfg);

*lgr*.info("$$$$$$$$$$$$ SqlSession id " + ss.toString());

MybatisUtil **MybatisUtil1** = **new** MybatisUtil();

Object **rzt**= ss.selectOne("CliSendAdvice.getParameter", map);

ss.close();

System.***out***.println(rzt);

}

## **、二级缓存**

1. 二级缓存(second level cache)，全局作用域缓存；二级缓存默认不开启，需要手动配置
2. MyBatis提供二级缓存的接口以及实现，缓存实现要求 POJO实现Serializable接口
3. 二级缓存在 SqlSession 关闭或提交之后才会生效

# 1.5  二级应用场景

对于访问多的查询请求且用户对查询结果实时性要求不高，此时可采用mybatis二级缓存技术降低数据库访问量，提高访问速度，业务场景比如：耗时较高的统计分析sql、电话账单查询sql等。

         实现方法如下：通过设置刷新间隔时间，由mybatis每隔一段时间自动清空缓存，根据数据变化频率设置缓存刷新间隔****flushInterval****，比如设置为30分钟、60分钟、24小时等，根据需求而定。

## **1.6  二级缓存局限性**

         mybatis二级缓存对细粒度的数据级别的缓存实现不好，对同时缓存较多条数据的缓存，比如如下需求：对商品信息进行缓存，由于商品信息查询访问量大，但是要求用户每次都能查询最新的商品信息，此时如果使用mybatis的二级缓存就无法实现当一个商品变化时只刷新该商品的缓存信息而不刷新其它商品的信息，因为mybaits的二级缓存区域以mapper为单位划分，当一个商品信息变化会将所有商品信息的缓存数据全部清空。解决此类问题需要在业务层根据需求对数据有针对性缓存。需要使用三级缓存

mybatis的缓存机制（一级缓存二级缓存和刷新缓存）和mybatis整合ehcache - CSDN博客.html