# MyBatis缓存介绍

正如大多数持久层框架一样，MyBatis 同样提供了**一级缓存**和**二级缓存**的支持

1.**一级缓存**: 基于PerpetualCache 的 HashMap本地缓存，其**存储作用域为** **Session**，当 **Session flush 或 close** 之后，该**Session中的所有 Cache 就将清空**。

2. **二级缓存**与一级缓存其机制相同，默认也是采用 PerpetualCache，HashMap存储，不同在于其**存储作用域为 Mapper(Namespace)**，并且**可自定义存储源**，如 Ehcache。

3. 对于缓存数据更新机制，当某一个作用域(一级缓存Session/二级缓存Namespaces)的进行了 C/U/D 操作后，默认该作用域下所有 select 中的缓存将被clear。

#### 1.1、Mybatis一级缓存测试

|  |
| --- |
| /\*\*  \* **@author** gacl  \* 测试一级缓存  \*/  **public** **class** TestOneLevelCache {    /\*  \* 一级缓存: 也就Session级的缓存(默认开启)  \*/  @Test  **public** **void** testCache1() {  SqlSession session = MyBatisUtil.getSqlSession();  String statement = "me.gacl.mapping.userMapper.getUser";  User user = session.selectOne(statement, 1);  System.***out***.println(user);    /\*  \* 一级缓存默认就会被使用  \*/  user = session.selectOne(statement, 1);  System.***out***.println(user);  session.close();  /\*  1. 必须是同一个Session,如果session对象已经close()过了就不可能用了  \*/  session = MyBatisUtil.getSqlSession();  user = session.selectOne(statement, 1);  System.***out***.println(user);    /\*  2. 查询条件是一样的  \*/  user = session.selectOne(statement, 2);  System.***out***.println(user);    /\*  3. 没有执行过session.clearCache()清理缓存  \*/  //session.clearCache();  user = session.selectOne(statement, 2);  System.***out***.println(user);    /\*  4. 没有执行过增删改的操作(这些操作都会清理缓存)  \*/  session.update("me.gacl.mapping.userMapper.updateUser", **new** User(2, "user", 23));  user = session.selectOne(statement, 2);  System.***out***.println(user);    }  } |

#### 1.2、Mybatis二级缓存测试

　　1、开启二级缓存，在userMapper.xml文件中添加如下配置

|  |
| --- |
| <mapper namespace="me.gacl.mapping.userMapper">  <!-- 开启二级缓存 -->  <cache/> |

　　2、测试二级缓存

|  |
| --- |
| /\*\*  \* **@author** gacl  \* 测试二级缓存  \*/  **public** **class** TestTwoLevelCache {    /\*  \* 测试二级缓存  \* 使用两个不同的SqlSession对象去执行相同查询条件的查询，第二次查询时不会再发送SQL语句，而是直接从缓存中取出数据  \*/  @Test  **public** **void** testCache2() {  String statement = "me.gacl.mapping.userMapper.getUser";  SqlSessionFactory factory = MyBatisUtil.getSqlSessionFactory();  //开启两个不同的SqlSession  SqlSession session1 = factory.openSession();  SqlSession session2 = factory.openSession();  //使用二级缓存时，User类必须实现一个Serializable接口===> User implements Serializable  User user = session1.selectOne(statement, 1);  session1.commit();//不懂为啥，这个地方一定要提交事务之后二级缓存才会起作用  System.***out***.println("user="+user);    //由于使用的是两个不同的SqlSession对象，所以即使查询条件相同，一级缓存也不会开启使用  user = session2.selectOne(statement, 1);  //session2.commit();  System.***out***.println("user2="+user);  }  } |

#### 1.3、二级缓存补充说明

　　1. 映射语句文件中的所有select语句将会被缓存。

　　2. 映射语句文件中的所有insert，update和delete语句会刷新缓存。

　　3. 缓存会使用Least Recently Used（LRU，最近最少使用的）算法来收回。

　　4. 缓存会根据指定的时间间隔来刷新。

5. 缓存会存储1024个对象

#### cache标签常用属性：

|  |
| --- |
| <cache  eviction=*"FIFO"* <!--回收策略为先进先出-->  flushInterval="60000" <!--自动刷新时间60s-->  size="512" <!--最多缓存512个引用对象-->  readOnly="true" <!--只读-->  /> |