

06-MyBatis基于XML的详细使用——缓存

06-MyBatis基于XML的详细使用——缓存

2、一级缓存的使用

2、二级缓存

1、缓存的使用

2、缓存的属性

3、二级缓存的作用范围

4、整合第三方缓存

1、介绍

MyBatis 内置了一个强大的事务性查询缓存机制，它可以非常方便地配置和定制。为了使它更加强大而且易于配置，我们对 MyBatis 3 中的缓存实现进行了许多改进。

默认情况下，只启用了本地的会话缓存，它仅仅对一个会话中的数据进行缓存。要启用全局的二级缓存，只需要在你的 SQL 映射文件中添加一行：

```
1 <cache/>
```

当添加上该标签之后，会有如下效果：

- 映射语句文件中的所有 select 语句的结果将会被缓存。
- 映射语句文件中的所有 insert、update 和 delete 语句会刷新缓存。
- 缓存会使用最近最少使用算法（LRU, Least Recently Used）算法来清除不需要的缓存。
- 缓存不会定时进行刷新（也就是说，没有刷新间隔）。
- 缓存会保存列表或对象（无论查询方法返回哪种）的 1024 个引用。
- 缓存会被视为读/写缓存，这意味着获取到的对象并不是共享的，可以安全地被调用者修改，而不干扰其他调用者或线程所做的潜在修改。

在进行配置的时候还会分为一级缓存和二级缓存：

一级缓存：线程级别的缓存，是本地缓存，sqlSession级别的缓存

二级缓存：全局范围的缓存，不止局限于当前会话

2、一级缓存的使用

一级缓存是sqlsession级别的缓存，默认是存在的。在下面的案例中，大家发现我发送了两个相同的请求，但是sql语句仅仅执行了一次，那么就意味着第一次查询的时候已经将结果进行了缓存。

```
1 @Test
```

```

2  public void test01() {
3
4      SqlSession sqlSession = sqlSessionFactory.openSession();
5      try {
6          EmpDao mapper = sqlSession.getMapper(EmpDao.class);
7          List<Emp> list = mapper.selectAllEmp();
8          for (Emp emp : list) {
9              System.out.println(emp);
10         }
11         System.out.println("-----");
12         List<Emp> list2 = mapper.selectAllEmp();
13         for (Emp emp : list2) {
14             System.out.println(emp);
15         }
16     } catch (Exception e) {
17         e.printStackTrace();
18     } finally {
19         sqlSession.close();
20     }
21 }

```

在大部分的情况下一级缓存是可以的，但是有几种特殊的情况会造成一级缓存失效：

1、一级缓存是sqlSession级别的缓存，如果在应用程序中只有开启了多个sqlSession，那么会造成缓存失效

```

1  @Test
2  public void test02(){
3      SqlSession sqlSession = sqlSessionFactory.openSession();
4      EmpDao mapper = sqlSession.getMapper(EmpDao.class);
5      List<Emp> list = mapper.selectAllEmp();
6      for (Emp emp : list) {
7          System.out.println(emp);
8      }
9      System.out.println("=====");
10     SqlSession sqlSession2 = sqlSessionFactory.openSession();
11     EmpDao mapper2 = sqlSession2.getMapper(EmpDao.class);
12     List<Emp> list2 = mapper2.selectAllEmp();
13     for (Emp emp : list2) {
14         System.out.println(emp);

```

```

15     }
16     sqlSession.close();
17     sqlSession2.close();
18 }

```

2、在编写查询的sql语句的时候，一定要注意传递的参数，如果参数不一致，那么也不会缓存结果

3、如果在发送过程中发生了数据的修改，那么结果就不会缓存

```

1  @Test
2  public void test03(){
3      SqlSession sqlSession = sqlSessionFactory.openSession();
4      EmpDao mapper = sqlSession.getMapper(EmpDao.class);
5      Emp empByEmpno = mapper.findEmpByEmpno(1111);
6      System.out.println(empByEmpno);
7      System.out.println("=====");
8      empByEmpno.setName("zhangsan");
9      int i = mapper.updateEmp(empByEmpno);
10     System.out.println(i);
11     System.out.println("=====");
12     Emp empByEmpno1 = mapper.findEmpByEmpno(1111);
13     System.out.println(empByEmpno1);
14     sqlSession.close();
15 }

```

4、在两次查询期间，手动去清空缓存，也会让缓存失效

```

1  @Test
2  public void test03(){
3      SqlSession sqlSession = sqlSessionFactory.openSession();
4      EmpDao mapper = sqlSession.getMapper(EmpDao.class);
5      Emp empByEmpno = mapper.findEmpByEmpno(1111);
6      System.out.println(empByEmpno);
7      System.out.println("=====");
8      System.out.println("手动清空缓存");
9      sqlSession.clearCache();
10     System.out.println("=====");
11     Emp empByEmpno1 = mapper.findEmpByEmpno(1111);
12     System.out.println(empByEmpno1);
13     sqlSession.close();
14 }

```

特性

```
1  /**
2   * 一级缓存
3   * 特性:
4   * 1. 默认就开启了, 也可以关闭一级缓存 localCacheScope=STATEMENT
5   * 2. 作用域: 是基于sqlSession (默认), 一次数据库操作会话。
6   * 3. 缓存默认实现类PerpetualCache ,使用map进行存储的
7   * 4. 查询完就会进行存储
8   * 5. 先从二级缓存中获取, 再从一级缓存中获取
9   * key==>    sqlid+sql
10  * 失效情况:
11  * 1. 不同的sqlSession会使一级缓存失效
12  * 2. 同一个SqlSession, 但是查询语句不一样
13  * 3. 同一个SqlSession, 查询语句一样, 期间执行增删改操作
14  * 4. 同一个SqlSession, 查询语句一样, 执行手动清除缓存
15  */
```

2、二级缓存

二级缓存是全局作用域缓存, 默认是不开启的, 需要手动进行配置。

Mybatis提供二级缓存的接口以及实现, 缓存实现的时候要求实体类实现Serializable接口, 二级缓存在sqlSession关闭或提交之后才会生效。

1、缓存的使用

步骤:

1、全局配置文件中添加如下配置:

```
1 <setting name="cacheEnabled" value="true"/>
```

2、需要在使用二级缓存的映射文件出使用<cache/>标签标注

3、实体类必须要实现Serializable接口

```
1 @Test
2     public void test04(){
3         SqlSession sqlSession = sqlSessionFactory.openSession();
4         SqlSession sqlSession2 = sqlSessionFactory.openSession();
5         EmpDao mapper = sqlSession.getMapper(EmpDao.class);
6         EmpDao mapper2 = sqlSession2.getMapper(EmpDao.class);
7         Emp empByEmpno = mapper.findEmpByEmpno(1111);
8         System.out.println(empByEmpno);
```

```

9      sqlSession.close();
10
11      Emp empByEmpno1 = mapper2.findEmpByEmpno(1111);
12      System.out.println(empByEmpno1);
13      sqlSession2.close();
14  }

```

2、缓存的属性

eviction:表示缓存回收策略，默认是LRU

LRU：最近最少使用的，移除最长时间不被使用的对象

FIFO：先进先出，按照对象进入缓存的顺序来移除

SOFT：软引用，移除基于垃圾回收器状态和软引用规则的对象

WEAK：弱引用，更积极地移除基于垃圾收集器状态和弱引用规则的对象

flushInterval:刷新闻隔，单位毫秒

默认情况是不设置，也就是没有刷新闻隔，缓存仅仅调用语句时刷新

size：引用数目，正整数

代表缓存最多可以存储多少个对象，太大容易导致内存溢出

readonly：只读，true/false

true：只读缓存，会给所有调用这返回缓存对象的相同实例，因此这些对象不能被修改。

false：读写缓存，会返回缓存对象的拷贝（序列化实现），这种方式比较安全，默认值

```

1  //可以看到会去二级缓存中查找数据，而且二级缓存跟一级缓存中不会同时存在数据，因为二级缓存中的数据是
2  @Test
3      public void test05(){
4          SqlSession sqlSession = sqlSessionFactory.openSession();
5          EmpDao mapper = sqlSession.getMapper(EmpDao.class);
6          Emp empByEmpno = mapper.findEmpByEmpno(1111);
7          System.out.println(empByEmpno);
8          sqlSession.close();
9
10         SqlSession sqlSession2 = sqlSessionFactory.openSession();
11         EmpDao mapper2 = sqlSession2.getMapper(EmpDao.class);
12         Emp empByEmpno2 = mapper2.findEmpByEmpno(1111);
13         System.out.println(empByEmpno2);
14         Emp empByEmpno3 = mapper2.findEmpByEmpno(1111);
15         System.out.println(empByEmpno3);
16         sqlSession2.close();
17     }

```

// 缓存查询的顺序是先查询二级缓存再查询一级缓存

```
1  @Test
2      public void test05(){
3          SqlSession sqlSession = sqlSessionFactory.openSession();
4          EmpDao mapper = sqlSession.getMapper(EmpDao.class);
5          Emp empByEmpno = mapper.findEmpByEmpno(1111);
6          System.out.println(empByEmpno);
7          sqlSession.close();
8
9          SqlSession sqlSession2 = sqlSessionFactory.openSession();
10         EmpDao mapper2 = sqlSession2.getMapper(EmpDao.class);
11         Emp empByEmpno2 = mapper2.findEmpByEmpno(1111);
12         System.out.println(empByEmpno2);
13         Emp empByEmpno3 = mapper2.findEmpByEmpno(1111);
14         System.out.println(empByEmpno3);
15
16         Emp empByEmpno4 = mapper2.findEmpByEmpno(7369);
17         System.out.println(empByEmpno4);
18         Emp empByEmpno5 = mapper2.findEmpByEmpno(7369);
19         System.out.println(empByEmpno5);
20         sqlSession2.close();
21     }
```

3、二级缓存的作用范围：

如果设置了全局的二级缓存配置，那么在使用的时候需要注意，在每一个单独的select语句中，可以设置将查询缓存关闭，以完成特殊的设置

1、在setting中设置，是配置二级缓存开启，一级缓存默认一直开启

```
1 <setting name="cacheEnabled" value="true"/>
```

2、select标签的useCache属性：

在每一个select的查询中可以设置当前查询是否要使用二级缓存，只对二级缓存有效

3、sql标签的flushCache属性

增删改操作默认值为true，sql执行之后会清空一级缓存和二级缓存，而查询操作默认是false

4、sqlSession.clearCache()

只是用来清楚一级缓存

特性：

```

1  /**
2   * 二级缓存:
3   *    特性:
4   *      1.默认开启了, 没有实现
5   *      2.作用域: 基于全局范围, 应用级别。
6   *      3.缓存默认实现类PerpetualCache ,使用map进行存储的但是二级缓存根据不同的mapper命名空间
7   *          : org.apache.ibatis.session.Configuration#caches      key:mapper命名空间
8   *          * key==> sqlid+sql
9   *      4.事务提交的时候 (sqlSession关闭)
10  *      5.先从二级缓存中获取, 再从一级缓存中获取
11  *    实现:
12  *      1.开启二级缓存<setting name="cacheEnabled" value="true"/>
13  *      2.在需要使用到二级缓存的映射文件中加入<cache></cache>,基于Mapper映射文件来实现缓存的,
14  *      3.在需要使用到二级缓存的javaBean中实现序列化接口implements Serializable
15  *          配置成功就会出现缓存命中率 同一个sqlId: 从缓存中拿出的次数/查询总次数
16  *
17  *    失效:
18  *      1.同一个命名空间进行了增删改的操作, 会导致二级缓存失效
19  *          但是如果不想要失效: 可以将SQL的flushCache 这是为false,但是要慎重设置, 因为会造成数据
20  *      2.让查询不缓存数据到二级缓存中useCache="false"
21  *      3.如果希望其他mapper映射文件的命名空间执行了增删改清空另外的命名空间就可以设置:
22  *          <cache-ref namespace="cn.tulingxueyuan.mapper.DeptMapper"/>
23  *
24  */

```

4、整合第三方缓存

1.整合redis

1.1.需要安装redis服务: <https://github.com/MicrosoftArchive/redis/tags>

1.2启动服务: 双击redis-server.exe 或 安装到windows服务: windows下redis配

置密码 (可选)

1.3测试redis是否能够正常运行:

1. 双击redis-cli.exe

2. 有密码的情况下输入密码

```
1 auth 密码
```

3.存储缓存set 命令

```
1 set key value
```

4.获取缓存 get命令

```
1 get key
```

1.4 添加redis-mybatis 缓存适配器 依赖

```
1 <dependencies>
2     <!--添加依赖-->
3     <dependency>
4         <groupId>org.mybatis.caches</groupId>
5         <artifactId>mybatis-redis</artifactId>
6         <version>1.0.0-beta2</version>
7     </dependency>
8 </dependencies>
```

1.5 添加redis.properties在resources根目录

```
1 host=localhost
2 port=6379
3 connectionTimeout=5000
4 soTimeout=5000
5 password=无密码可不填
6 database=0
7 clientName=
```

1.6 设置mybatis二级缓存实现类

```
1 <cache
2     ...
3     type="org.mybatis.caches.redis.RedisCache"
4     .../>
```

2.整合ehcache

1、导入对应的maven依赖

```
1 <!-- https://mvnrepository.com/artifact/org.ehcache/ehcache -->
2     <dependency>
```



```

3         <groupId>org.ehcache</groupId>
4         <artifactId>ehcache</artifactId>
5         <version>3.8.1</version>
6     </dependency>
7     <!-- https://mvnrepository.com/artifact/org.mybatis.caches/mybatis-ehcache -->
8     <dependency>
9         <groupId>org.mybatis.caches</groupId>
10        <artifactId>mybatis-ehcache</artifactId>
11        <version>1.2.0</version>
12    </dependency>
13

```

2、导入ehcache配置文件

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xsi:noNamespaceSchemaLocation="http://ehcache.org/ehcache.xsd">
4  <!-- 磁盘保存路径 -->
5  <diskStore path="D:\ehcache" />
6
7  <defaultCache
8      maxElementsInMemory="1"
9      maxElementsOnDisk="10000000"
10     eternal="false"
11     overflowToDisk="true"
12     timeToIdleSeconds="120"
13     timeToLiveSeconds="120"
14     diskExpiryThreadIntervalSeconds="120"
15     memoryStoreEvictionPolicy="LRU">
16  </defaultCache>
17  </ehcache>
18
19  <!--
20  属性说明：
21  1 diskStore: 指定数据在磁盘中的存储位置。
22  1 defaultCache: 当借助CacheManager.add("demoCache")创建Cache时，EhCache便会采用<defaultCache>
23
24  以下属性是必须的：
25  1 maxElementsInMemory - 在内存中缓存的element的最大数目
26  1 maxElementsOnDisk - 在磁盘上缓存的element的最大数目，若是0表示无穷大

```

```
27 l eternal - 设定缓存的elements是否永远不过期。如果为true，则缓存的数据始终有效，如果为false那么
28 l overflowToDisk - 设定当内存缓存溢出的时候是否将过期的element缓存到磁盘上
29
30 以下属性是可选的：
31 l timeToIdleSeconds - 当缓存在EhCache中的数据前后两次访问的时间超过timeToIdleSeconds的属性取值时，就将元素从EhCache中删除。该属性的默认值是0，表示将元素永久保留在缓存中。
32 l timeToLiveSeconds - 缓存element的有效生命期，默认是0.，也就是element存活时间无穷大
33 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都应该有一个自己的DiskStore。
34 l diskPersistent - 在VM重启的时候是否启用磁盘保存EhCache中的数据，默认是false。
35 l diskExpiryThreadIntervalSeconds - 磁盘缓存的清理线程运行间隔，默认是120秒。每个120s，相应的线程就会清理磁盘缓存区中的过期数据。
36 l memoryStoreEvictionPolicy - 当内存缓存达到最大，有新的element加入的时候，移除缓存中element
37 -->
```

3、在mapper文件中添加自定义缓存

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