**1、本章面试题**

            说说你对mybatis一级和二级缓存的理解

**2、知识点**

**2.1、课程回顾**

          动态sql :

where +if , where+set,choose when,trim, foreach

高级映射：resultMap

association 一对一 多对一

collection       一对多 多对多

**2.2、本章重点**

         一二级缓存

注解的用法

反向工程

**3、具体内容**

**3.1 一级二级缓存**

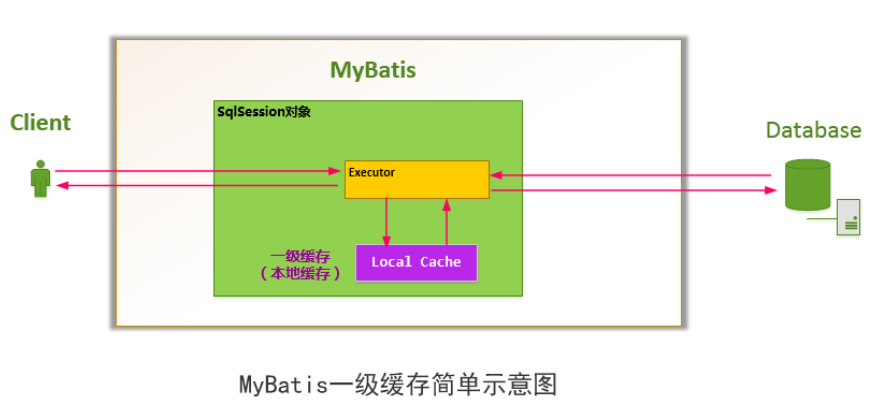
MyBatis 内置了一个强大的事务性查询缓存机制，它可以非常方便地配置和定制。 为了使它更加强大而且易于配置，我们对 MyBatis 3 中的缓存实现进行了许多改进。

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。

**一级缓存**：

一级缓存默认开启的，基于SqlSession级别的缓存,每一个sqlsession之间是相互独立的。

当SqlSession执行：关闭、更新操作（增删改）、清空缓存操作时：会清空缓存；



**测试：**

打印接口的查询结果，相同说明是同一对象，如果数据库数据发生变化，就会重新缓存，更新数据后，再次查询

1，全局配置中加入控制台日志打印配置

<!--开启控制台日志-->

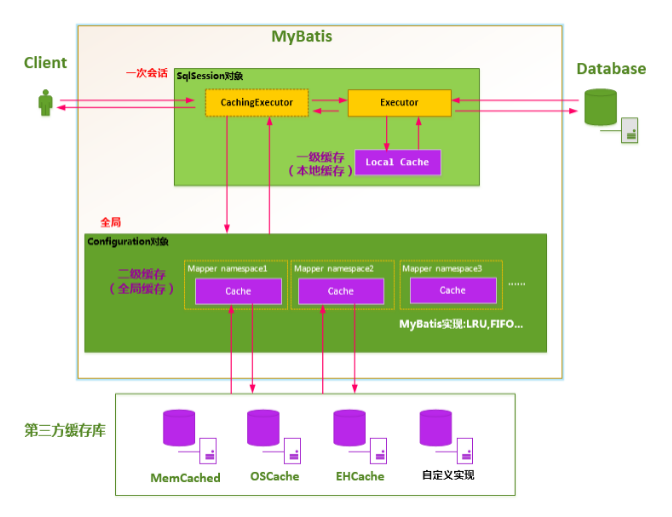
<setting name="logImpl" value="STDOUT\_LOGGING"/>

2，编写测试类



**二级缓存：**

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。对应到每一个mapper。



缓存效果：

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

**二级缓存实现过程：**

**1，全局配置（加入日志，可以观察语句执行情况）**

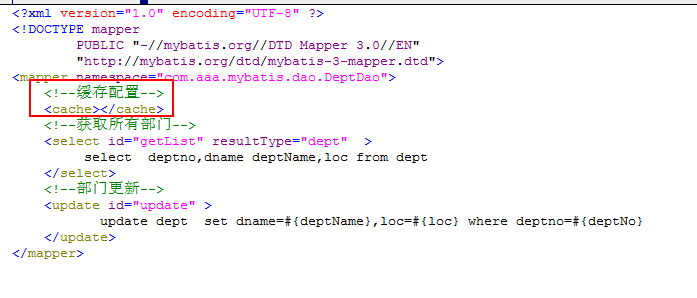
<!--配置日志-->

<setting name="logImpl" value="STDOUT\_LOGGING"/>

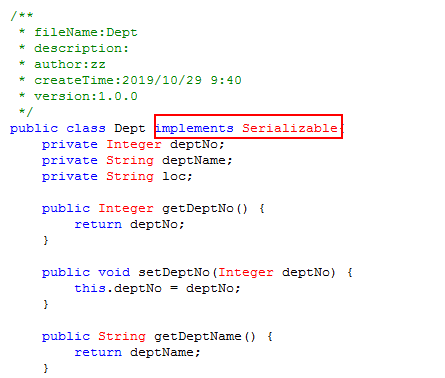
<!--全局开启缓存，不配置默认为开启，配置false，二级缓存不可用-->

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

**2，mapper配置<cache></cache>**



**3，如果使用实体，实体需要实现序列化接口**Serializable



**4，测试不同SqlSession查询数据时，语句执行情况**

**测试增删改后，缓存变化**

**测试flushInterval**

**测试flushCache="false"**



**3.2 注解的使用**

**DeptDao:**

package com.aaa.mybatis.dao;

import com.aaa.mybatis.entity.Dept;

import com.sun.scenario.effect.impl.sw.sse.SSEBlend\_SRC\_OUTPeer;

import org.apache.ibatis.annotations.\*;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

/\*\*

\* fileName:UserDao

\* description:

\* author:zz

\* createTime:2019/10/29 9:17

\* version:1.0.0

\*/

public interface DeptDao {

/\*\*

\*获取部门分页数据

\* @return

\*/

@Select({"<script>select deptno,dname deptName,loc from (" ,

"select rownum rn,a.\* from " ,

"(select \* from dept " ,

"<where>" ,

"<if test='deptName!=null'> and dname like '%'||#{deptName}||'%'</if>",

"<if test='loc!=null'> and loc like '%'||#{loc}||'%'</if>"+

" </where>",

" order by deptno desc) a where rownum &lt; #{end} ",

") b where b.rn &gt; #{start} </script>"})

List<Dept> getPage(Map map);

/\*\*

\*获取部门分页数据

\* @return

\*/

@Select({"<script>select deptno,dname deptName,loc from (" ,

"select rownum rn,a.\* from " ,

"(select \* from dept " ,

"<where>" ,

"<if test='deptName!=null'> and dname like '%'||#{deptName}||'%'</if>",

"<if test='loc!=null'> and loc like '%'||#{loc}||'%'</if>"+

" </where>",

" order by deptno desc) a where rownum &lt; #{end} ",

") b where b.rn &gt; #{start} </script>"})

List<Dept> getPageA(@Param("deptName") String deptName,@Param("loc") String loc,@Param("start") int start,@Param("end") int end);

/\*\*

\* 根据ID获取对象

\* @param deptNo

\* @return

\*/

@Select("select deptno,dname deptName,loc from dept where deptno = #{deptNo}")

Dept getById(int deptNo);

/\*\*

\* 部门更新

\* @param dept

\* @return

\*/

@Update("update dept set dname = #{deptName},loc = #{loc} where deptno=#{deptNo}")

int update(Dept dept);

/\*\*

\* 添加

\* @param dept

\* @return

\*/

@Insert("insert into dept values(seq\_dept\_id.nextval,#{deptName},#{loc})")

int add(Dept dept);

/\*\*

\* 删除

\* @param deptNo

\* @return

\*/

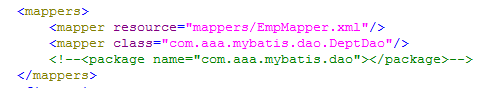
@Delete("delete from dept where deptno=#{deptId}")

int delete(int deptNo);

}

//</editor-fold>

**mybatis-config.xml:**



**测试：**

package com.aaa.mybatis.test;

import com.aaa.mybatis.dao.DeptDao;

import com.aaa.mybatis.entity.Dept;

import com.aaa.mybatis.util.SqlSessionFacotryUtil;

import org.apache.ibatis.session.SqlSession;

import org.junit.Test;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

/\*\*

\* fileName:DeptDaoTest

\* description:

\* author:zz

\* createTime:2019/10/30 9:12

\* version:1.0.0

\*/

public class DeptDaoTest {

/\*\*

\* 分页测试

\*/

@Test

public void testGetPage(){

SqlSession sqlSession = null;

try {

sqlSession = SqlSessionFacotryUtil.getSqlSession();

DeptDao deptDao = sqlSession.getMapper(DeptDao.class);

int pageNo = 1;

int pageSize = 3;

// 计算开始值

int start = (pageNo-1)\*pageSize;

//计算结束值

int end = pageNo\*pageSize+1;

Map map = new HashMap();

//map.put("deptName","");

// map.put("loc","1");

map.put("start",start);

map.put("end",end);

List<Dept> deptList = deptDao.getPage(map);

if(deptList!=null&&deptList.size()>0){

for (Dept dept : deptList) {

System.out.println(dept.getDeptName());

}

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(sqlSession!=null)

sqlSession.close();

}

}

/\*\*

\* 分页测试

\*/

@Test

public void testGetPageA(){

SqlSession sqlSession = null;

try {

sqlSession = SqlSessionFacotryUtil.getSqlSession();

DeptDao deptDao = sqlSession.getMapper(DeptDao.class);

int pageNo = 1;

int pageSize = 3;

// 计算开始值

int start = (pageNo-1)\*pageSize;

//计算结束值

int end = pageNo\*pageSize+1;

List<Dept> deptList = deptDao.getPageA("",null,start,end);

if(deptList!=null&&deptList.size()>0){

for (Dept dept : deptList) {

System.out.println(dept.getDeptName());

}

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(sqlSession!=null)

sqlSession.close();

}

}

/\*\*

\* 测试根据ID获取对象

\*/

@Test

public void testGetById(){

SqlSession sqlSession = null;

try {

sqlSession = SqlSessionFacotryUtil.getSqlSession();

DeptDao deptDao = sqlSession.getMapper(DeptDao.class);

Map map = new HashMap();

Dept dept = deptDao.getById(10);

if(dept!=null ){

System.out.println(dept.getDeptName());

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(sqlSession!=null)

sqlSession.close();

}

}

/\*\*

\* 测试添加

\*/

@Test

public void testAdd(){

SqlSession sqlSession = null;

try {

sqlSession = SqlSessionFacotryUtil.getSqlSession();

DeptDao deptDao = sqlSession.getMapper(DeptDao.class);

Dept deptA = new Dept();

deptA.setLoc("42楼");

deptA.setDeptName("开发42部");

int i = deptDao.add(deptA);

sqlSession.commit();

if(i >0 ){

System.out.println("操作成功");

}else {

System.out.println("操作失败");

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(sqlSession!=null)

sqlSession.close();

}

}

/\*\*

\* 测试更新

\*/

@Test

public void testUpdate(){

SqlSession sqlSession = null;

try {

sqlSession = SqlSessionFacotryUtil.getSqlSession();

DeptDao deptDao = sqlSession.getMapper(DeptDao.class);

Dept deptA = new Dept();

deptA.setDeptNo(42);

deptA.setLoc("422楼");

deptA.setDeptName("开发422部");

int i = deptDao.update(deptA);

sqlSession.commit();

if(i >0 ){

System.out.println("操作成功");

}else {

System.out.println("操作失败");

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(sqlSession!=null)

sqlSession.close();

}

}

/\*\*

\* 测试更新

\*/

@Test

public void testDelete(){

SqlSession sqlSession = null;

try {

sqlSession = SqlSessionFacotryUtil.getSqlSession();

DeptDao deptDao = sqlSession.getMapper(DeptDao.class);

int i = deptDao.delete(42);

sqlSession.commit();

if(i >0 ){

System.out.println("操作成功");

}else {

System.out.println("操作失败");

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(sqlSession!=null)

sqlSession.close();

}

}

}

**3.3 逆向工程**

MyBatis生成器（MBG）是MyBatis MyBatis 和iBATIS的代码生成器。它将为MyBatis的所有版本以及版本2.2.0之后的iBATIS生成代码。它将内省一个数据库表（或多个表），并将生成可用于访问表的工件。这减轻了设置对象和配置文件以与数据库表进行交互的麻烦。

**1， 引入jar包，配置插件：**

<!-- https://mvnrepository.com/artifact/org.mybatis.generator/mybatis-generator-core -->

<dependency>

<groupId>org.mybatis.generator</groupId>

<artifactId>mybatis-generator-core</artifactId>

<version>1.3.7</version>

</dependency>

<!-- mybatis generator 自动生成代码插件 -->

<plugin>

<groupId>org.mybatis.generator</groupId>

<artifactId>mybatis-generator-maven-plugin</artifactId>

<version>1.3.7</version>

<configuration>

<!--指定配置文件的名称。默认值:${basedir}/src/main/resources/generatorConfig.xml-

->

<configurationFile>src/main/resources/generatorConfig.xml</configurationFile>

<!--新生成的文件会覆盖原有的文件。-->

<overwrite>true</overwrite>

<!--如果指定该参数，执行过程会输出到控制台-->

<verbose>true</verbose>

</configuration>

</plugin>

**2，创建generatorConfig配置文件**

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

<!DOCTYPE generatorConfiguration

PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"

"http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd">

<generatorConfiguration>

<!-- 数据库驱动包位置 -->

<classPathEntry location="E:\ojdbc6.jar" />

<context id="oracle" targetRuntime="MyBatis3">

<commentGenerator>

<property name="suppressDate" value="true"/>

<!-- 是否去除自动生成的注释 true：是 ： false:否 -->

<property name="suppressAllComments" value="true"/>

</commentGenerator>

<jdbcConnection driverClass="oracle.jdbc.driver.OracleDriver"

connectionURL="jdbc:oracle:thin:@localhost:1521:orcl"

userId="scott"

password="tiger">

</jdbcConnection>

<!-- 默认false，把JDBC DECIMAL 和 NUMERIC 类型解析为 Integer，为 true时把JDBC DECIMAL 和 NUMERIC 类型解析为java.math.BigDecimal -->

<javaTypeResolver >

<property name="forceBigDecimals" value="false" />

</javaTypeResolver>

<!--生成实体类-->

<javaModelGenerator targetPackage="com.aaa.mybatis.entity1" targetProject="src/main/java">

<property name="enableSubPackages" value="true" />

<property name="trimStrings" value="true" />

</javaModelGenerator>

<!--生成映射文件-->

<sqlMapGenerator targetPackage="mappers1" targetProject="src/main/resources">

<property name="enableSubPackages" value="true" />

</sqlMapGenerator>

<!--生成mapper文件对应的dao-->

<javaClientGenerator type="XMLMAPPER"

targetPackage="com.aaa.mybatis.dao1"

targetProject="src/main/java">

<property name="enableSubPackages" value="true" />

</javaClientGenerator>

<!--配置表信息-->

<table tableName="emp"

domainObjectName="Emp"

enableCountByExample="false"

enableUpdateByExample="false"

enableDeleteByExample="false"

enableSelectByExample="false"

selectByExampleQueryId="false" />

<table tableName="dept"

domainObjectName="Dept"

enableCountByExample="false"

enableUpdateByExample="false"

enableDeleteByExample="false"

enableSelectByExample="false"

selectByExampleQueryId="false" />

</context>

</generatorConfiguration>

**3.执行命令：**

mvn mybatis-generator:generate

**4、本章总结**

**4.1 总结本章知识点**

**4.2 面试题答案**

**4.3 预习下一章重点**