**1、导入需要的jar库**

**2、添加mybatis-config.xml、log4j.properties文件**

**3、创建需要的包**

Dao包

Domain包

Mapper包

Service包

Util包

Test包

**4、在工具包：util包创建工具**

（1）创建SqlSessionUtil类

public class SqlSessionUtil {

private SqlSessionUtil() {}

private static SqlSessionFactory *sqlSessionFactory*;

static {

String resource = "mybatis-config.xml";

InputStream inputStream = null;*//输入*

try {

*//通过加载Mybatis的主配置文件mybatis-config.xml，创建输入对象*

inputStream = Resources.*getResourceAsStream*(resource);

} catch (IOException e) {

e.printStackTrace();

}

*sqlSessionFactory* = new SqlSessionFactoryBuilder().build(inputStream);

}

private static ThreadLocal<SqlSession> *threadLocal* = new ThreadLocal<SqlSession>();

*//取得SqlSession对象*

public static SqlSession getSession() {

SqlSession session = *threadLocal*.get();

if (session == null) {

session = *sqlSessionFactory*.openSession();

*threadLocal*.set(session);

}

return session;

}

*//关闭SqlSession对象*

public static void myClose(SqlSession session) {

if (session != null) {

session.close();

*//必须加，非常容易忘记：线程并没有销毁，而是回到了线程池中，需要手动删除*

*threadLocal*.remove();

}

}

}

（2）创建TransactionInvocationHandler代理类

*//动态代理模式:背*

public class TransactionInvocationHandler implements InvocationHandler {

*//terget:张三*

private Object target;

public TransactionInvocationHandler(Object target) {

this.target = target;

}

*//代理类的业务方法*

*//invoke ：李四的表白方法*

@Override

public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {

SqlSession session = null;

Object object = null;

try {

*/\*SqlSessionUtil util = new SqlSessionUtil();*

*util.getSession();\*///垃圾代码*

session = SqlSessionUtil.*getSession*();*//使用类直接调方法*

*//处理业务与逻辑*

*//method.invoke ：张三的表白方法*

object = method.invoke(target, args);

*//处理业务逻辑后，提交事务*

session.commit();

} catch (Exception e) {

session.rollback();

} finally {

SqlSessionUtil.*myClose*(session);

}

return object;

}

*//取得ls对象*

public Object getProxy() {

return Proxy.*newProxyInstance*(target.getClass().getClassLoader(), target.getClass().getInterfaces(), this);

}

}

（3）创建ServiceFactory类

public class ServiceFactory {

*//传递zs对象，得到ls对象的过程*

public static Object getService(Object service){

return

new TransactionInvocationHandler(service).getProxy();

}

}

**5、写dao层**

（1）写dao层接口

public interface StudentDao {

public Student getById(String id);

public void save(Student student);

}

（2）写dao层接口实现类：impl --> StudentDaoImpl

public class StudentDaoImpl implements StudentDao {

@Override

public Student getById(String id) {

SqlSession session = SqlSessionUtil.*getSession*();

Student student = session.selectOne("test1.getById", id);

return student;

}

@Override

public void save(Student student) {

SqlSession session = SqlSessionUtil.*getSession*();

session.insert("test1.save", student);

*//session.commit();//在业务层提交事务：Service*

}

}

**6、写Mapper映射文件**

<mapper namespace="test1">

<select id="getById" parameterType="java.lang.String" resultType="com.cn.hm.domain.Student">

select \* from tbl\_student where id = #{id}

</select>

<insert id="save">

insert into tbl\_student(id,name,age) values(#{id},#{name},#{age})

</insert>

</mapper>

**7、业务层**

（1）写service接口

public interface StudentService {

public Student getById(String id);

public void save(Student student);

}

（2）写service接口实现类：impl —> StudentServiceImpl

public class StudentServiceImpl implements StudentService {

private StudentDao studentDao = new StudentDaoImpl();*//多态*

@Override

public Student getById(String id) {

Student student = studentDao.getById(id);

return student;

}

@Override

public void save(Student student) {

studentDao.save(student);

}

}

**8、写test测试层**

public class Test1 {

public static void main(String[] args) {

*// StudentService studentService = new StudentServiceImpl();*

StudentService studentService = (StudentService) ServiceFactory.*getService*(new StudentServiceImpl());

*//测试添加操作*

*/\* Student student = new Student();*

*student.setId("A0006");*

*student.setName("cxk");*

*student.setAge(26);*

*studentService.save(student);\*/*

*//测试根据id查询单条操作*

Student student = studentService.getById("A0001");

System.*out*.println(student);

}

}