Mybatis的第一种方式

SqlMapConfig.xml文件

<configuration>

<!-- 和spring整合后 environments配置将废除 -->

<environments default=*"development"*>

<environment id=*"development"*>

<!-- 使用jdbc事务管理 -->

<transactionManager type=*"JDBC"* />

<!-- 数据库连接池 -->

<dataSource type=*"POOLED"*>

<property name=*"driver"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"url"*

value=*"jdbc:mysql://localhost:3306/mybatis?characterEncoding=utf-8"* />

<property name=*"username"* value=*"root"* />

<property name=*"password"* value=*"root"* />

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource=*"com/itheima/mybatis/sqlmap/User.xml"*/>

</mappers>

</configuration>

Pojo中的user类:

**public** **class** User {

**private** Integer id;

**private** String username;// 用户姓名

**private** String sex;// 性别

**private** Date birthday;// 生日

**private** String address;// 地址

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getSex() {

**return** sex;

}

**public** **void** setSex(String sex) {

**this**.sex = sex;

}

**public** Date getBirthday() {

**return** birthday;

}

**public** **void** setBirthday(Date birthday) {

**this**.birthday = birthday;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

/\* (non-Javadoc)

\* @see java.lang.Object#toString()

\*/

@Override

**public** String toString() {

**return** "User [" + (id != **null** ? "id=" + id + ", " : "")

+ (username != **null** ? "username=" + username + ", " : "") + (sex != **null** ? "sex=" + sex + ", " : "")

+ (birthday != **null** ? "birthday=" + birthday + ", " : "")

+ (address != **null** ? "address=" + address : "") + "]";

}

}

User.xml配置:

<mapper namespace=*"test"*>

<select id=*"queryUserById"* parameterType=*"int"* resultType=*"com.itheima.mybatis.pojo.User"*>

select \* from user where id=#{id}

</select>

</mapper>

Juint测试类:

**public** **class** MybatisTest {

@Test

**public** **void** testFindUserById() **throws** Exception{

String resource = "SqlMapConfig.xml";

InputStream in =Resources.*getResourceAsStream*(resource);

SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(in);

SqlSession sqlSession = sqlSessionFactory.openSession();

User user=sqlSession.selectOne("test.queryUserById",25);

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

}

}

模糊查询

User.xml:

<select id=*"queryUserByUserName"* parameterType=*"String"* resultType=*"com.itheima.mybatis.pojo.User"*>

select \* from user where username like "%"#{vvv}"%"

</select>

测试类:

@Test

**public** **void** testQueryUserByUserName() **throws** Exception{

String resource = "SqlMapConfig.xml";

InputStream in = Resources.*getResourceAsStream*(resource);

SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(in);

SqlSession sqlSession = sqlSessionFactory.openSession();

List<Object> userLIst = sqlSession.selectList("test.queryUserByUserName","五");

**for** (Object object : userLIst) {

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

}

}

添加用户:

User.xml:

</select>

<insert id=*"saveUser"* parameterType=*"com.itheima.mybatis.pojo.User"*>

insert into user

(username,birthday,sex,address) values

(#{username},#{birthday},#{sex},#{address})

</insert>

测试类:

@Test

**public** **void** testSaveUser() **throws** IOException{

String resource = "SqlMapConfig.xml";

InputStream in = Resources.*getResourceAsStream*(resource);

SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(in);

SqlSession sqlSession = sqlSessionFactory.openSession();

User user = **new** User();

user.setUsername("123456");

user.setId(30);

sqlSession.insert("test.saveUser",user);

sqlSession.commit();//注意提交事务.

}

主键自增类型的用户添加:

<insert id=*"addUser"* parameterType=*"com.itheima.mybatis.pojo.User"* useGeneratedKeys=*"true"* keyProperty=*"id"*>

insert into user

(username,sex,birthday,address)

values

(#{username},#{sex},#{birthday},#{address})

</insert>

@Test

**public** **void** testSaveUser() **throws** IOException{

String resource = "SqlMapConfig.xml";

InputStream in = Resources.*getResourceAsStream*(resource);

SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(in);

SqlSession sqlSession = sqlSessionFactory.openSession();

User user = **new** User();

user.setUsername("123456");

sqlSession.insert("test.addUser",user);

sqlSession.commit();

}

修改用户

<update id=*"updateUserById"* parameterType=*"com.itheima.mybatis.pojo.User"*>

update user set

username = #{username} where id = #{id}

</update>

@Test

**public** **void** testUpdateUser() **throws** IOException{

String resource = "SqlMapConfig.xml";

InputStream in = Resources.*getResourceAsStream*(resource);

SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(in);

SqlSession sqlSession = sqlSessionFactory.openSession();

User user = **new** User();

user.setUsername("華仔");

user.setId(30);

sqlSession.update("test.updateUserById",user);

sqlSession.commit();//特别注意要提交事务,要不然修改不成功

sqlSession.close();//记得关闭资源.

}

根据id查询用户:

<delete id=*"deleteUser"* parameterType=*"integer"*>

delete from user where id=#{id}

</delete>

@Test

**public** **void** testDeleteUser() **throws** IOException{

String resource = "SqlMapConfig.xml";

InputStream in = Resources.*getResourceAsStream*(resource);

SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(in);

SqlSession sqlSession = sqlSessionFactory.openSession();

sqlSession.delete("test.deleteUser",31);

sqlSession.commit();

sqlSession.close();

}

SqlSessionFactory的最佳使用范围是整个应用运行期间

原始dao的开发:

创建接口

**public** **interface** userDao {

User queryById(**int** id);

List<User> queryUserByUsername(String username);

**void** saveUser(User user);

}

创建实现类:

**public** **class** userDaoImpl **implements** userDao {

**private** SqlSessionFactory sqlSessionFactory;

**public** userDaoImpl(SqlSessionFactory sqlSessionFactory) {

**super**();

**this**.sqlSessionFactory=sqlSessionFactory;

}

@Override

**public** User queryById(**int** id) {

SqlSession sqlSession = **this**.sqlSessionFactory.openSession();

User user = sqlSession.selectOne("queryUserById",30);

sqlSession.close();

**return** user;

}

@Override

**public** List<User> queryUserByUsername(String username) {

SqlSession sqlSession = **this**.sqlSessionFactory.openSession();

List<User> list = sqlSession.selectList("queryUserByUserName","五");

sqlSession.close();

**return** list;

}

@Override

**public** **void** saveUser(User user) {

// 创建SqlSession

SqlSession sqlSession = **this**.sqlSessionFactory.openSession();

// 执行保存逻辑

sqlSession.insert("saveUser", user);

// 提交事务

sqlSession.commit();

// 释放资源

sqlSession.close();

}

}

创建测试类:

**public** **class** MyBatisDao {

**private** SqlSessionFactory sqlSessionFactory;

@Before

**public** **void** init() **throws** Exception {

// 创建SqlSessionFactoryBuilder

SqlSessionFactoryBuilder sqlSessionFactoryBuilder = **new** SqlSessionFactoryBuilder();

// 加载SqlMapConfig.xml配置文件

InputStream inputStream = Resources.*getResourceAsStream*("SqlMapConfig.xml");

// 创建SqlsessionFactory

**this**.sqlSessionFactory = sqlSessionFactoryBuilder.build(inputStream);

}

@Test

**public** **void** testQueryUserById() {

// 创建DAO

userDao userDao = **new** userDaoImpl(**this**.sqlSessionFactory);

// 执行查询

User user = userDao.queryById(1);

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

}

@Test

**public** **void** testQueryUserByUsername() {

// 创建DAO

userDao userDao = **new** userDaoImpl(**this**.sqlSessionFactory);

// 执行查询

List<User> list = userDao.queryUserByUsername("张");

**for** (User user : list) {

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

}

}

@Test

**public** **void** testSaveUser() {

// 创建DAO

userDao userDao = **new** userDaoImpl(**this**.sqlSessionFactory);

// 创建保存对象

User user = **new** User();

user.setUsername("刘备");

user.setBirthday(**new** Date());

user.setSex("1");

user.setAddress("蜀国");

// 执行保存

userDao.saveUser(user);

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

}

}

Mapper的动态代理:

Mapper接口开发需要遵循以下规范：

1. Mapper.xml文件中的namespace与mapper接口的类路径相同。
2. Mapper接口方法名和Mapper.xml中定义的每个statement的id相同
3. Mapper接口方法的输入参数类型和mapper.xml中定义的每个sql 的parameterType的类型相同
4. Mapper接口方法的输出参数类型和mapper.xml中定义的每个sql的resultType的类型相同

UserMapper.xml配置文件内容：

<!-- namespace：命名空间，用于隔离sql -->

<!-- 还有一个很重要的作用，使用动态代理开发DAO，1. namespace必须和Mapper接口类路径一致 -->

<mapper namespace=*"cn.itcast.mybatis.mapper.UserMapper"*>

<!-- 根据用户id查询用户 -->

<!-- 2. id必须和Mapper接口方法名一致 -->

<!-- 3. parameterType必须和接口方法参数类型一致 -->

<!-- 4. resultType必须和接口方法返回值类型一致 -->

@Test

**public** **void** testQueryUserById() {

// 创建DAO

// 获取sqlSession，和spring整合后由spring管理

SqlSession sqlSession = **this**.sqlSessionFactory.openSession();

// 执行查询

userDao dao = sqlSession.getMapper(userDao.**class**);

User user = dao.queryUserById(1);

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

}

SqlMapConfig.xml中属性配置的内容和顺序如下：

<typeAliases>

<!-- 单个别名定义 -->

<typeAlias alias=*"user"* type=*"cn.itcast.mybatis.pojo.User"* />

<!-- 批量别名定义，扫描整个包下的类，别名为类名（大小写不敏感） -->

<package name=*"cn.itcast.mybatis.pojo"* />

<package name=*"其它包"* />

</typeAliases>

在mapper.xml配置文件中，就可以使用设置的别名了

别名大小写不敏感

### <mapper class=" " />

使用mapper接口类路径

如：<mapper class="cn.itcast.mybatis.mapper.UserMapper"/>

注意：此种方法要求mapper接口名称和mapper映射文件名称相同，且放在同一个目录中。

### <package name=""/>

注册指定包下的所有mapper接口

如：<package name="cn.itcast.mybatis.mapper"/>

注意：此种方法要求mapper接口名称和mapper映射文件名称相同，且放在同一个目录中。