链滴

MyBatis 常用注解及基本增删改查的注解实现

作者：terwergreen

原文链接：https://ld246.com/article/1661917665843

来源网站：链滴

MyBatis 的常用注解

注解可以减少 Mapper 文件的编写，常用注解如下;

@Insert：实现新增

@Update：实现更新

@Delete：实现删除

@Select：实现查询

@Result：实现结果集封装

@Results：可以和@Result 一起使用，封装多个结果集

@One：实现一对一结果集封装

@Many：实现多对多结果集封装

MyBatis 的增删改查

数据库配置依旧保存不变

jdbc.properties

```
# 新版驱动名称发生了改变
# jdbc.driver=com.mysql.jdbc.Driver
jdbc.driver=com.mysql.cj.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/test?characterEncoding=utf8&useSSL=false
jdbc.username=terwer
jdbc.password=123456
```

sqlMapConfig.xml

User

```
class User {
    var id: Int? = null
    var username: String? = null

    override fun toString(): String {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                '}'
    }
}
```

```
public class User {
    private Integer id;
    private String username;

    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;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                '}';
    }
}
```

UserMapper

```
interface UserMapper {
    @Select("select * from user")
    fun findAll(): List?

    @Insert("insert into user(username) values(#{username})")
    fun add(user: User?)

    @Update("update user set username=#{username} where id=#{id}")
    fun update(user: User?)

    @Delete("delete from user where id=#{id}")
    fun delete(id: Int?)
}

/**
 * 用户映射
 *
 * @name: UserMapper
 * @author: terwer
 * @date: 2022-05-25 13:27
 **/
public interface UserMapper {
    @Select("select * from user")
    List findAll();

    @Insert("insert into user(username) values(#{username})")
    void add(User user);

    @Update("update user set username=#{username} where id=#{id}")
    void update(User user);
```

```
    @Delete("delete from user where id=#{id}")
    void delete(Integer id);
}
```

结果测试

UserMapper

```
class UserMapperTest {
    private var userMapper: UserMapper? = null
    private var sqlSession: SqlSession? = null
    @Before
    @Throws(Exception::class)
    fun before() {
        println("before...")
        val resourceAsStream = Resources.getResourceAsStream("sqlMapConfig.xml")
        val sqlSessionFactory = SqlSessionFactoryBuilder().build(resourceAsStream)
        sqlSession = sqlSessionFactory.openSession()
        userMapper = sqlSession?.getMapper(UserMapper::class.java)
    }

    @Test
    fun testFindAll() {
        val all = userMapper!!.findAll()
        for (user in all) {
            println(user)
        }
    }

    @Test
    @Throws(IOException::class)
    fun add() {
        val user = User()
        user.username = "测试3"
        userMapper!!.add(user)

        // 这里一定要加，否则不会提交事务
        sqlSession!!.commit(true)
    }

    @Test
    fun update() {
        val user = User()
        user.id = 3
        user.username = "测试11"
        userMapper!!.update(user)

        // 这里一定要加，否则不会提交事务
        sqlSession!!.commit(true)
    }

    @Test
    fun delete() {
        userMapper!!.delete(3)
```

```
        // 这里一定要加，否则不会提交事务
        sqlSession!!.commit(true)
    }
}

public class UserMapperTest {

    private UserMapper userMapper;
    private SqlSession sqlSession;

    @Before
    public void before() throws Exception {
        System.out.println("before...");
        InputStream resourceAsStream = Resources.getResourceAsStream("sqlMapConfig.xml");
        SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(resourceAsS
ream);
        sqlSession = sqlSessionFactory.openSession();
        userMapper = sqlSession.getMapper(UserMapper.class);
    }

    @Test
    public void testFindAll() {
        List all = userMapper.findAll();
        for (User user : all) {
            System.out.println(user);
        }
    }

    @Test
    public void add() throws IOException {
        User user = new User();
        user.setUsername("测试3");
        userMapper.add(user);

        // 这里一定要加，否则不会提交事务
        sqlSession.commit(true);
    }

    @Test
    public void update() {
        User user = new User();
        user.setId(3);
        user.setUsername("测试11");
        userMapper.update(user);

        // 这里一定要加，否则不会提交事务
        sqlSession.commit(true);
    }

    @Test
    public void delete() {
        userMapper.delete(3);
```

```
        // 这里一定要加，否则不会提交事务
        sqlSession.commit(true);
    }
}
```

注意：默认不会自动提交事务，可以手动设置SqlSession，也可以在创建SqlSession的时候指定自动交事务。

```
// sqlSession = sqlSessionFactory.openSession();
// 这样也是可以的，这样的话后面就不用每次都设置了
sqlSession = sqlSessionFactory.openSession(true);
```

修改MyBatis的核心配置文件，我们使用了注解替代的映射文件，所以我们只需要加载使用了注解的Mpper接口

即可

或者指定扫描包含映射关系的接口所在的包也可以

本文代码地址

mybatis-annotation

文章更新历史

2022-08-30 feat:初稿