使用mybatis实现单张表CRUD

1、创建项目c-mybatis-crud，新建lib目录

2、编写配置文件mybatis-config.xml和SqlMapper.xml，只要放到类路径下即可

3、一般情况下，连接数据库的信息单独写到一个属性配置文件中，使得用户修改比较方便，就算用户再不会改这个属性配置文件，我们还可以写一个脚本文件（如.bat、.exe）帮助用户修改属性配置文件，从而更改xml文件。

在类的根路径下，新建一个jdbc.properties,配置连接数据库的信息。如下

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/mysqlstudy
jdbc.username=root
jdbc.password=rong195302
```

然后在核心配置文件中引入该文件，同时配置好sql映射文件SqlMapper.xml的路径，则mybatis-config.xml如下

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>

    <!--引入独立的配置文件-->
    <!--mybatis中的resource默认从项目的类路径查找-->
    <properties resource="jdbc.properties"/>

    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <!--注意${jdbc.driver}不是EL表达式，是mybatis自定制的语法
                <property name="driver" value="${jdbc.driver}"/>
                <property name="url" value="${jdbc.url}"/>
                <property name="username" value="${jdbc.username}"/>
                <property name="password" value="${jdbc.password}"/>
            </dataSource>
        </environment>
    </environments>
    <mappers>
        <mapper resource="SqlMapper.xml"/>
    </mappers>
</configuration>
```

我们在新建编写xml、html等文件的时候，发现生成的文件上面有如下面的开头信息

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
```

其中有两条信息"-//mybatis.org//DTD Config 3.0//EN"和"http://mybatis.org/dtd/mybatis-3-config.dtd"，前者是后者的唯一标识名字，后者是一个dtd约束文件，"mybatis-3-config.dtd"是约束mybatis3核心配置文件的约束文件，这种文件是约束要编辑的mybatis核心配置文件应该填什么标签，怎么填，填的顺序怎么样等等，填的时候会有提示让你选择什么标签，以防你填错，其他类型的文件填写也有自己的dtd约束文件，如sql映射文件有"mybatis-3-mapper.dtd"。当我们idea连网时，使用的约束文件就是通过网址"http://mybatis.org/dtd/mybatis-3-config.dtd"来约束的，当我们不连网时，是不会有提示的，这就需要我们下载后各种文件的dtd约束文件配置到idea中。方法如下（idea）：

"-//mybatis.org//DTD Config 3.0//EN"，这个就是一个唯一标识id名，idea通过这个名字找到对应的约束文件，对应的可以是URL（网上的dtd资源）也可以是本地dtd资源路径。

我们将"http://mybatis.org/dtd/mybatis-3-config.dtd"双引号中的网址复制

点击File->Settings->Languages & Frameworks->点击Schemas and DTDs->点击右边的+->将网址填到URL项，下面的file找到自己下载好的dtd文件->Apply-Ok

这样idea到这个"http://mybatis.org/dtd/mybatis-3-config.dtd"网址找约束文件的时候就会去本地找了，不需要联网了。

4、创建javabean，如下（没有特殊情况，尽量将实体类中的属性字段名与表中的一致）

```
package com.mybatis.domain;

public class User {

    private int id;
    private String loginName;
    private String loginPwd;
    private String realName;

    public User() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getLoginName() {
        return loginName;
    }

    public void setLoginName(String loginName) {
        this.loginName = loginName;
    }
```

```
    public String getLoginPwd() {
        return loginPwd;
    }

    public void setLoginPwd(String loginPwd) {
        this.loginPwd = loginPwd;
    }

    public String getRealName() {
        return realName;
    }

    public void setRealName(String realName) {
        this.realName = realName;
    }

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

5、编写sql映射文件SqlMapper.xml，如下

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="test">
    <!--
        注意：parameterType属性是给sql语句中占位符传值的，定义占位符传什么类型对象的属性的值到该位置上
        翻译为：参数类型，下面insert语句中传递的是com.mybatis.domain.User类的属性字段值
        同时javabean在给占位符传值时，程序员要告诉mybatis应该将javabean的哪个属性字段值传到哪个占位符上
        所以占位符中需要有javabean的属性字段名
        mybatis中占位符不用问号表示，因为要放javabean的字段信息，所以用：#{这里写javabean的属性字段名} 表示
        #{loginName}底层原理是mybatis会将#{}中的属性字段名获取到，将其首字母大写，然后在前面添加get，调用javabean中的各字段
        的get方法来获取到值，从而组成sql语句执行
    -->
    <insert id="save" parameterType="com.mybatis.domain.User">
        insert into t_user
            (loginName,loginPwd,realName)
        values
            (#{loginName},#{loginPwd},#{realName})
```

```
    </insert>

    <!--能复制的就复制，不要自己将属性名敲上去，以防敲错-->
    <update id="update" parameterType="com.mybatis.domain.User">
        update t_user set
            loginPwd = #{loginPwd},realName = #{realName}
        where
            id = #{id}
    </update>

    <!--resultType属性是查询结果集类型，我们发现insert和update标签是没有resultType属性的，只有在select标签中才有-->
    <!--parameterType这个属性是专门负责给sql语句传值的-->
    <!--当传值的参数类型标签中的数据类型是"简单类型"时，该属性可以省略
        即   parameterType="简单类型"  可省略
        但是
             resultType="简单类型"  是不能省略的
        "简单类型"：17种
        byte short int long float double boolean char
        Byte Short Integer Long Float Double Boolean Character
        String
        由于这里根据id查询一条记录，传入一个int类型的id，自动封装成Integer类型，可省略
        而且只传入一个参数，#{}中的名字可任意编写
    -->
    <!--<select id="selectOne" parameterType="java.lang.Integer" resultType="com.mybatis.domain.User">-->
    <select id="getById"  resultType="com.mybatis.domain.User">
        select
            id,loginName,loginPwd,realName
        from
            t_user
        where
            id = #{id}
    </select>

    <select id="getAll" resultType="com.mybatis.domain.User">
        select
            *
        from
            t_user
    </select>

    <delete id="deleteById">
        delete from t_user
        where
        id=#{id}
    </delete>
</mapper>
```

6、创建MyBatisCRUD.java测试类，代码如下

```
package com.mybatis.test;

import com.mybatis.domain.User;
import org.apache.ibatis.io.Resources;
```

```
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import javax.annotation.Resource;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class MyBatisTest01 {

    public static void main(String[] args) {

        SqlSession sqlSession = null;

        try {

//            String resource = "mybatis-config.xml";
//            InputStream inputStream = Resources.getResourceAsStream(resource);
//            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(inputStream);
            //合并上面代码
            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(Resources.getResourceAsReader("mybatis-
config.xml"));

            //开启事务
            sqlSession = sqlSessionFactory.openSession();

            /*
                do work() 编写业务代码
             */
            //insert
//            User user = new User();
//            user.setLoginName("18888688@qq.com");
//            user.setLoginPwd("1238888ffff");
//            user.setRealName("马云&马化腾");
            //sqlSession.insert()方法中的参数可以只传一个sql映射文件中sql语句的id字符
串，表示sql映射文件中的insert语句是这样的：
            // insert into t_user(loginName,lodinPwd,realName)
values("xxx","yyy","zzz") 所以不需要再传一个javabean对象
            //而这里我们是需要传javabean对象的，传两个参数
            //返回一个int类型数据，表示改动的记录条数
//            int count1 = sqlSession.insert("save",user);

            //update
//            User user = new User();
//            user.setId(5);
//            user.setLoginPwd("88888888");
//            user.setRealName("大马猴");
//            int count = sqlSession.update("update",user);
//            System.out.println(count);

            //selectOne
            User user1 = sqlSession.selectOne("getById",1);
            System.out.println(user1);
            System.out.println("=======================================");
```

```
            //selectList
            List<User> userList1 = new ArrayList<>();
            userList1 = sqlSession.selectList("getAll");
            for (User user2:
                 userList1) {
                System.out.println(user2);
            }
            System.out.println("========================================");

            //delete
            int count = sqlSession.delete("deleteById",5);
            System.out.println(count);
            System.out.println("========================================");

            //selectList
            List<User> userList2 = new ArrayList<>();
            userList2 = sqlSession.selectList("getAll");
            for (User user2:
                    userList2) {
                System.out.println(user2);
            }

            //事务正常结束，提交
            sqlSession.commit();
        } catch (IOException e) {
            //出现异常，事务回滚
            if (sqlSession != null) {
                sqlSession.rollback();
            }
            e.printStackTrace();
        }finally {
            //关闭资源
            if (sqlSession != null) {
                sqlSession.close();
            }
        }

    }

}
```

在上面的sql映射文件中，有一个缺点：

参数类型属性parameterType中我们要把javabean的完整类名填上去，每一条sql语句的parameterType属性都写上一个全限定类名，发现非常的重复，开发效率降低。

所以这里介绍mybatis中在核心配置文件中对javabean起别名的方式解决上面的缺点，核心配置文件如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
```

```
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>

    <!--引入独立的配置文件-->
    <!--mybatis中的resource默认从项目的类路径查找-->
    <properties resource="jdbc.properties"/>

    <!--mybatis别名机制-->
    <!--第一种：该方式可以自定义别名，但是实体类较多的话，对很多javabean起别名，不方便-->
    <!--
    <typeAliases>
        <typeAlias type="com.mybatis.domain.User" alias="User"/>
    </typeAliases>
    -->
    <!--
    第二种：通过对某个包如实体类包domain中的所有javabean起别名，
    所有javabean的别名是其简化类名，缺点是不能自定义别名,大部分情况使用
    -->
    <typeAliases>
        <package name="com.mybatis.domain"/>
    </typeAliases>

    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="${jdbc.driver}"/>
                <property name="url" value="${jdbc.url}"/>
                <property name="username" value="${jdbc.username}"/>
                <property name="password" value="${jdbc.password}"/>
            </dataSource>
        </environment>
    </environments>
    <mappers>
        <mapper resource="SqlMapper2.xml"/>
    </mappers>
</configuration>
```

然后将项目中的新建一个用别名的SqlMapper2.xml文件，然后核心配置文件关联新建的sql映射文件，如下

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="test">
    <insert id="save" parameterType="User">
        insert into t_user
            (loginName,loginPwd,realName)
        values
            (#{loginName},#{loginPwd},#{realName})
    </insert>

    <update id="update" parameterType="User">
```

```
        update t_user set
            loginPwd = #{loginPwd},realName = #{realName}
        where
            id = #{id}
    </update>


    <select id="getById"  resultType="User">
        select
            id,loginName,loginPwd,realName
        from
            t_user
        where
            id = #{id}
    </select>

    <select id="getAll" resultType="User">
        select
            *
        from
            t_user
    </select>

    <delete id="deleteById">
        delete from t_user
        where
        id=#{id}
    </delete>
</mapper>
```