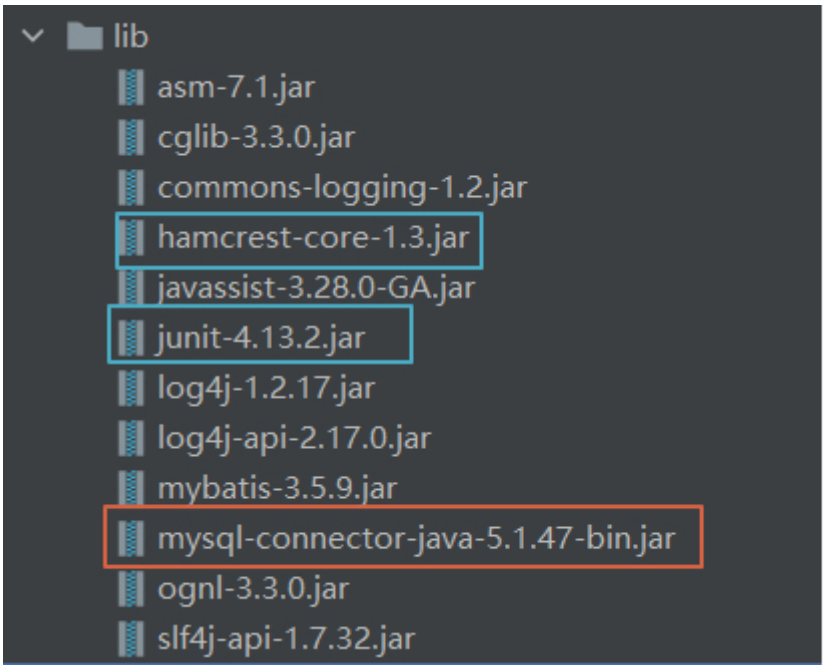
MyBatis入门

一、安装和基础

MyBatis是一个持久化层框架，可以帮助开发人员快速完成数据库的访问，只需要下载mybatis的jar，加入到项目的lib中即可



蓝色标注的是用来做测试的，橙色是数据库连接包，其他都是mybatis的依赖包。在项目中加入了junit，junit可以帮助开发人员进行单元测试，在学习阶段中，引入junit之后，可以在一个类中运行多个方法。加入junit的jar包之后，在任意一个类中方法上添加`@Test`这个方法就是一个单元测试方法，可以直接运行

```
package org.wlw.test;

import org.junit.Test;

public class TestUnit {
    @Test
    public void hello() {
        System.out.println("hello");
    }

    @Test
    public void world() {
        System.out.println("world");
    }
}
```

二、MyBatis的应用

要使用mybatis需要有如下几个核心步骤：

1、 创建mybatis的配置文件

可以在配置文件中编写如何连接数据库，首先在src的根目录创建mybatis-config.xml,首先是进行环境配置，用来设置如何访问数据库

```
<configuration>
    <environments default="development">
        <!--enviroment用来进行数据库的连接配置-->
        <environment id="development">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <!--
                    dirver:设置驱动
                    url:设置连接字符串,jdbc:mysql://localhost:3306/20wlw_manager?
characterEncoding=utf8，表示使用utf8的字符编码连接20wlw_manager
                    username:用户名
                    password:密码
                 -->
                <property name="driver" value="com.mysql.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://localhost:3306/20wlw_manager?
characterEncoding=utf8"/>
                <property name="username" value="20wlw"/>
                <property name="password" value="20wlw123"/>
            </dataSource>
        </environment>
    </environments>
    <mappers>
    </mappers>
</configuration>
```

2、创建实体类

根据数据库中的表创建Student实体类

```java
package org.wlw.model;

public class Student {
    private int id;
    private String name;
    private String no;
    private int gender;
    private String qq;
    private String icon;
    private String mobile;
    private String address;
    private int cid;

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getNo() {
        return no;
    }

    public void setNo(String no) {
        this.no = no;
    }

    public int getGender() {
        return gender;
    }

    public void setGender(int gender) {
        this.gender = gender;
    }

    public String getQq() {
        return qq;
    }

    public void setQq(String qq) {
```

```
        this.qq = qq;
    }

    public String getIcon() {
        return icon;
    }

    public void setIcon(String icon) {
        this.icon = icon;
    }

    public String getMobile() {
        return mobile;
    }

    public void setMobile(String mobile) {
        this.mobile = mobile;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public int getCid() {
        return cid;
    }

    public void setCid(int cid) {
        this.cid = cid;
    }

    @Override
    public String toString() {
        return "Student{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", no='" + no + '\'' +
                ", gender=" + gender +
                ", qq='" + qq + '\'' +
                ", icon='" + icon + '\'' +
                ", mobile='" + mobile + '\'' +
                ", address='" + address + '\'' +
                ", cid=" + cid +
                '}';
    }
}
```

3、为实体类创建映射文件

通过映射文件完成对这个实体类的sql操作。该文件一般来说放到实体类的包中，建议取名为类名Mapper，如: `StudentMapper.xml`

```
<?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用来编写配置，namespace表示的是命名空间，防止重复，建议使用类的全路径-->
<mapper namespace="org.wlw.model.Student">
    <!--编写数据库的操作，id用来标识一个唯一的操作，resultType表示返回的类型-->
    <select id="find" resultType="org.wlw.model.Student">
        select * from t_stu
    </select>

    <select id="load" parameterType="int" resultType="org.wlw.model.Student">
        select * from t_stu where id=#{id}
    </select>
</mapper>
```

4、将映射文件添加到配置文件中

```
<mappers>
      <mapper resource="org/wlw/model/StudentMapper.xml"/>
</mappers>
```

完整的配置文件如下

```
<?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>
    <environments default="development">
        <!--enviroment用来进行数据库的连接配置-->
        <environment id="development">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <!--
                    dirver:设置驱动
                    url:设置连接字符串,jdbc:mysql://localhost:3306/20wlw_manager?
characterEncoding=utf8，表示使用utf8的字符编码连接20wlw_manager
                    username:用户名
                    password:密码
                 -->
                <property name="driver" value="com.mysql.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://localhost:3306/20wlw_manager?
characterEncoding=utf8"/>
                <property name="username" value="20wlw"/>
                <property name="password" value="20wlw123"/>
            </dataSource>
        </environment>
    </environments>
    <mappers>
        <mapper resource="org/wlw/model/StudentMapper.xml"/>
    </mappers>
</configuration>
```

5、具体的使用流程

首先创建SQLSessionFactory

```
String xml = "mybatis-config.xml";
//1、根据配置文件创建输入流
InputStream is = Resources.getResourceAsStream(xml);
//2、创建SqlSessionFactory来完成数据库配置
SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(is);
```

之后根据Factory创建session，并执行相应的sql

```
//3、根据Factory创建SqlSession
session = sqlSessionFactory.openSession();
//4、执行sql
List<Student> stus = session.selectList("org.wlw.model.Student.find");
for(Student stu:stus) {
     System.out.println(stu);
}
```

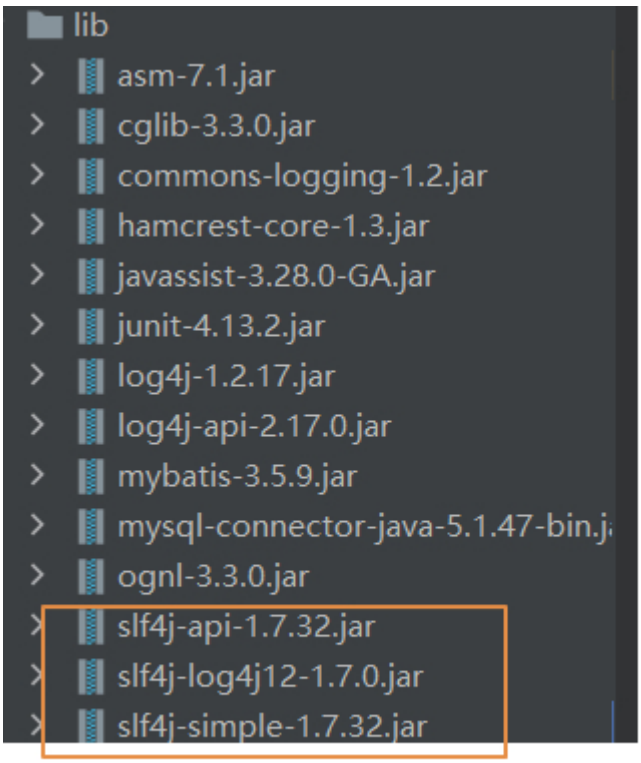
查找sql的方法是找到mapper，根据namespace和id发现要执行的sql。完整的代码如下

```java
package org.wlw.test;

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 org.junit.Before;
import org.junit.Test;
import org.wlw.model.Student;

import java.io.IOException;
import java.io.InputStream;
import java.util.List;

public class TestMybatis {
    SqlSessionFactory sqlSessionFactory;
    //@Before表示执行任意的单元测试都要提前执行的代码，可以做一些数据的初始化操作
    @Before
    public void init() {
//        System.out.println("before");
        try {
            String xml = "mybatis-config.xml";
            //1、根据配置文件创建输入流
            InputStream is = Resources.getResourceAsStream(xml);
//            System.out.println(is);
            //2、创建SqlSessionFactory来完成数据库配置
            sqlSessionFactory = new SqlSessionFactoryBuilder().build(is);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    @Test
    public void testMybatis() {
        SqlSession session = null;
        try {
            String xml = "mybatis-config.xml";
            //1、根据配置文件创建输入流
            InputStream is = Resources.getResourceAsStream(xml);
//            System.out.println(is);
            //2、创建SqlSessionFactory来完成数据库配置
            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(is);
            //3、根据Factory创建SqlSession
            session = sqlSessionFactory.openSession();
            //4、执行sql
            List<Student> stus = session.selectList("org.wlw.model.Student.find");
            for(Student stu:stus) {
                System.out.println(stu);
            }
            //5、关闭session
```

```
//            session.close();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            //关闭session
            if(session!=null) session.close();
        }
    }

    @Test
    public void selectOne() {
        SqlSession session = null;
        try {
            session = sqlSessionFactory.openSession();
            Student stu = session.selectOne("org.wlw.model.Student.load",1);
            System.out.println(stu);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close()
        }
    }
}
```

六、log4j简介

通过mybatis执行相应的命令之后，看不到sql语句，此时需要使用log4j来进行日志控制，log4j会使用到slf4j中的依赖，将包导入，导入完成之后的包如下图所示



之后在src目录中添加log4j.properties

```
#log4j日志级别如下：
#A：off      最高等级，用于关闭所有日志记录。
#B：fatal    指出每个严重的错误事件将会导致应用程序的退出。
#C：error    指出然发生错误事件，但仍然不影响系统的继续运行。
#D：warn     表明会出现潜在的错误情形。
#E：info     一般和在粗粒度级别上，强调应用程序的运行全程。
#F：debug    一般用于细粒度级别上，对调试应用程序非常有帮助。
#G：all      最低等级，用于打开所有日志记录。

#但log4j只建议使用4个级别，优先级从高到低分别是：
#error>warn>info>debug

#开关：debug级别,向控制台和文件输出
log4j.rootLogger =debug,systemOut,logFile

#输出到控制台
log4j.appender.systemOut = org.apache.log4j.ConsoleAppender
log4j.appender.systemOut.layout = org.apache.log4j.PatternLayout
log4j.appender.systemOut.layout.ConversionPattern = [%-5p][%-22d{yyyy/MM/dd
HH:mm:ssS}][%l]%n%m%n
log4j.appender.systemOut.Target = System.out

#输出到文件
log4j.appender.logFile = org.apache.log4j.FileAppender
log4j.appender.logFile.layout = org.apache.log4j.PatternLayout
log4j.appender.logFile.layout.ConversionPattern = [%-5p][%-22d{yyyy/MM/dd HH:mm:ssS}]
[%l]%n%m%n
log4j.appender.logFile.File = E:/log/log4j.log
log4j.appender.logFile.Encoding = UTF-8
```

添加了log4j之后可以在项目中通过`Logger.getRootLogger().debug("")`来输出信息，debug，info，warn，error这些都是日志级别。

七、实现增删改查

为StudentMapper添加其他的操作

```xml
<?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用来编写配置，namespace表示的是命名空间，防止重复，建议使用类的全路径-->
<mapper namespace="org.wlw.model.Student">
    <!--编写数据库的操作，id用来标识一个唯一的操作，resultType表示返回的类型-->
    <select id="find" resultType="org.wlw.model.Student">
        select * from t_stu
    </select>

    <select id="load" parameterType="int" resultType="org.wlw.model.Student">
        select * from t_stu where id=#{id}
    </select>

    <update id="update" parameterType="org.wlw.model.Student">
        update t_stu set no=#{no},name=#{name},icon=#{icon},mobile=#{mobile},qq=#{qq},
                         address=#{address},cid=#{cid},gender=#{gender} where id=#{id}
    </update>

    <delete id="delete" parameterType="int">
        delete from t_stu where id=#{id}
    </delete>

    <insert id="add" parameterType="org.wlw.model.Student">
        insert into t_stu(no,name,gender,qq,mobile,address,cid,icon) value
            (#{no},#{name},#{gender},#{qq},#{mobile},#{address},#{cid},#{icon})
    </insert>
</mapper>
```

具体的操作代码如下所示

```java
package org.wlw.test;

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 org.apache.log4j.Logger;
import org.junit.Before;
import org.junit.Test;
import org.wlw.model.Student;

import java.io.IOException;
import java.io.InputStream;
import java.util.List;

public class TestMybatis {
    SqlSessionFactory sqlSessionFactory;
    //@Before表示执行任意的单元测试都要提前执行的代码，可以做一些数据的初始化操作
    @Before
    public void init() {
//        System.out.println("before");
        try {
            String xml = "mybatis-config.xml";
            //1、根据配置文件创建输入流
            InputStream is = Resources.getResourceAsStream(xml);
//            System.out.println(is);
            //2、创建SqlSessionFactory来完成数据库配置
            sqlSessionFactory = new SqlSessionFactoryBuilder().build(is);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    private String getId(String id) {
        return "org.wlw.model.Student."+id;
    }
    @Test
    public void selectList() {
        SqlSession session = null;
        try {
            String xml = "mybatis-config.xml";
            //1、根据配置文件创建输入流
            InputStream is = Resources.getResourceAsStream(xml);
//            System.out.println(is);
            //2、创建SqlSessionFactory来完成数据库配置
            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(is);
            //3、根据Factory创建SqlSession
            session = sqlSessionFactory.openSession();
            //4、执行sql
            List<Student> stus = session.selectList(getId("find"));
```

```
            for(Student stu:stus) {
                System.out.println(stu);
            }
            //5、关闭session
//            session.close();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            //关闭session
            if(session!=null) session.close();
        }
    }

    @Test
    public void selectOne() {
        SqlSession session = null;
        try {
            session = sqlSessionFactory.openSession();
            Student stu = session.selectOne("org.wlw.model.Student.load",1);
            System.out.println(stu);
            Logger.getRootLogger().debug(stu);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
        }
    }

    @Test
    public void testUpdate() {
        try(SqlSession session = sqlSessionFactory.openSession()) {
            Student stu = session.selectOne("org.wlw.model.Student.load",3);
            stu.setNo("0233");
            stu.setMobile("343434");
            session.update("org.wlw.model.Student.update",stu);
            session.commit();//提交事务
        }
    }

    @Test
    public void testInsert() {
        try(SqlSession session = sqlSessionFactory.openSession()) {
            Student stu = new Student();
            stu.setName("王小二");
            stu.setNo("999");
            stu.setAddress("崂山区");
            stu.setCid(1);
            stu.setGender(1);
            stu.setIcon("002.png");
            stu.setQq("33333");
            stu.setMobile("2323333");
            session.insert("org.wlw.model.Student.add",stu);
```

```
            session.commit();
        }
    }

    @Test
    public void testDelete() {
        try(SqlSession session = sqlSessionFactory.openSession()) {
            session.delete("org.wlw.model.Student.delete",3);
            session.commit();
        }
    }
}
```