首先呢，我们用mybatis肯定涉及到mybatis的引用，其次，mybatis是用来连接数据库的，所以又涉及到对对应数据库的引用，所以最基本的我们要导入两个jar包，这边数据库以mysql为例。引入如下两个依赖（maven用法），对应两个jar包，分别提供了mabatis服务，连接mysql数据库服务。

```
<!-- Mybatis依赖 -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.2.5</version>
        </dependency>

        <!-- MySQL依赖 -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.38</version>
            <scope>runtime</scope>
        </dependency>
```

```
mybatis-3.2.5.jar
mysql-connector-java-5.1.38.jar
```

然后，我们就涉及到mybatis的配置文件，现在的流行框架大多数都会将配置工作交给xml在使用时解析，mybatis也不例外（后期整合spring后，就不会有单独的mybatis配置文件，不代表没有，只是整个的交给了spring的配置文件去配置）。废话不多数，先上配置文件。顺便提一下，配置文件随你放项目哪，只要你用的时候找到对应路径，一般学习时放src下即可（maven项目的话，放java下），这样读取时只需要配置文件名就可以找到。当然，数据库的配置信息对应自己的咯。

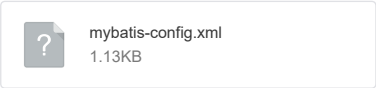
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!-- 连接数据库 -->
    <environments default="development">
        <environment id="development">
            <!-- 指定事务管理类型，JDBC直接简单使用JDBC的提交和回滚设置 -->
            <transactionManager type="JDBC" />
            <!-- 数据源配置，POOLED是JDBC连接对象的数据源连接池的实现 -->
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver" />
                <property name="url"
                    value="jdbc:mysql://localhost:3306/test?
useUnicode=true&amp;characterEncoding=UTF-8" />
                <property name="username" value="root" />
                <property name="password" value="123456" />
            </dataSource>
        </environment>
    </environments>

    <!-- 配置持久化映射文件的路径，通俗讲就是完成mapper接口或mapper xml文件在mybatis的注册
-->
    <mappers>
        <mapper resource="com/jlb/mapper/PersonMapper.xml" />
    </mappers>
</configuration>
```

这边也有张截图，看的更清楚点（LOG4J那个暂时无视，那是绑定日志的）

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybati
<configuration>
    <!-- 指定mybatis所用日志的具体实现 -->
    <settings>
        <setting name="logImpl" value="LOG4J" />
    </settings>
    <!-- 连接数据库 -->
    <environments default="mysql">
        <environment id="mysql">
            <!-- 指定事务管理类型，type="JDBC"直接简单使用JDBC的提交和回滚设置 -->
            <transactionManager type="JDBC" />
            <!-- database指数据源配置，POOLED是JDBC连接对象的数据源连接池的实现 -->
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver" />
                <property name="url" value="jdbc:mysql://localhost:3306/mybatis" />
                <property name="username" value="root" />
                <property name="password" value="123456" />
            </dataSource>
        </environment>
    </environments>
    <!-- mappers告诉mybatis去哪找持久化类的映射文件 -->
    <mappers>
        <mapper resource="com/jlb/mapper/CardMapper.xml" />
        <mapper resource="com/jlb/mapper/PersonMapper.xml" />
    </mappers>
</configuration>
```

以及直接的配置文件



然后就是我们的持久化类映射文件（mapper映射文件）了，三种方式，一种基于xml，一种基于注解写接口，还有一种就是xml结合接口了，但是接口就是纯粹的接口，和普通的接口一样，但不需要实现类，因为类名和mapper映射文件namespace对应的话，就已经自动绑定了，方法名对应mapper映射文件内的各种id，也会自动去匹配，或者说，这种方式才算是mybatis真正的魅力所在吧。这边因为是入门嘛，只基于第一种方式，以我这边所用的例子为例，写了一个PersonMapper.xml，也就是上面配置文件注册的那个映射文件。

该映射文件完成了简单crud的操作（为了笔记的简洁性以及代码的可读性，关于代码的解说，都有对应的注释，直接看代码就是），整体来说，入门算是够了。具体一些名词的解释不放在这里做。

```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 namespace="com.jlb.mapper.PersonMapper">
    <!-- 查询一个人 -->
    <select id="selectOnePerson" resultType="com.jlb.domain.Person">
        select * from person
        where
        id = #{id}
    </select>

    <!-- 查询所有人 -->
    <select id="selectAllPerson" resultType="com.jlb.domain.Person">
        select * from person
    </select>

    <!-- 删除一个人 -->
    <delete id="deleteOnePerson">
        delete from person
        where
        id = #{id}
    </delete>

    <!-- 新增一个人 -->
    <insert id="insertOnePerson">
        insert into person(name,age)
        values(#{name},#{age})
    </insert>
```

```
    <!-- 删除一个人 -->
    <update id="updateOnePerson">
        update person
        set name = #{name}
        where id = #{id}
    </update>
</mapper>
```

我们看看具体操作，先把配置文件放到流里，然后通过SqlSessionFactory创建相关东西并通过它创建我们主要执行体SqlSession对象，mybatis对对应操作都有对应的处理，查一个，查多个，新增，修改，删除都有。不过做QL时不需要commit，做DML时必须执行commit。（这边的代码用了Junit单元测试的方式，有部分看不懂，去查阅专门的Junit笔记）

```
public class TestPerson {

    private SqlSession session;
    private static SqlSessionFactory sqlSessionFactory;

    @BeforeClass
    public static void testBeforeClass() throws Exception{
        String resource = "mybatis-config.xml";
        InputStream inputStream = Resources.getResourceAsStream(resource);
        sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
    }

    @Before
    public void testBefore() {
        try {
            session = sqlSessionFactory.openSession();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @After
    public void testAfter() {
        session.close();
    }

    /*
     * 查询一个人
     */
    @Test
    public void testselectOnePerson() {
        try {
            Person person =
session.selectOne("com.jlb.mapper.PersonMapper.selectOnePerson", 1);
            System.out.println(person);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /*
     * 查询所有人
     */
    @Test
    public void testselectAllPerson() {
        try {
            List<Person> list =
session.selectList("com.jlb.mapper.PersonMapper.selectAllPerson");
            for (Object object : list) {
                System.out.println(object);
            }
            session.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /*
     * 新增一个人
     */
    @Test
    public void testinsertOnePerson() {
        try {
            Person person = new Person();
```

```
                person.setName("xiaoming");
                person.setAge(22);
                int i = session.insert("com.jlb.mapper.PersonMapper.insertOnePerson",
person);
                session.commit();
                System.out.println(i);
                session.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        /*
         * 修改一个人
         */
        @Test
        public void testupdateOnePerson() {
            try {
                Person person = new Person();
                person.setName("老莫");
                person.setId(4);
                int i = session.insert("com.jlb.mapper.PersonMapper.updateOnePerson",
person);
                session.commit();
                System.out.println(i);
                session.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        /*
         * 修改一个人
         */
        @Test
        public void testdeleteOnePerson() {
            try {
                int i = session.delete("com.jlb.mapper.PersonMapper.deleteOnePerson", 3);
                System.out.println(i);
                session.commit();
                session.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

}
```

最后上Person类

```
public class Person implements Serializable {

    private Integer id;
    private String name;
    private Integer age;

    public Integer getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
```

```
        this.age = age;
    }

    @Override
    public String toString() {
        return id + " " + name + " " + age;
    }

}
```

总结一下整体的操作过程，首先明确我们是在用mybatis开发，是连接数据库的框架，那么mybatis的jar包和mysql的jar包一定要导入。其次mybatis的配置文件基于xml，该配置文件的创建必不可少。在来我们要处理哪张表，其对应实体类和对应的映射文件有没有创建。最后在操作时有没有创建SqlSession对象，做对应处理，DML操作不要忘了commit。