第一章

1. 三层架构

界面层：和用户打交道的，接收用户的请求参数，西安市处理结果的。（jsp, html , servlet）

业务逻辑层：接收了界面层传递的数据，计算逻辑，调用数据库，获取数据

数据访问层：就是访问数据库，执行对数据的查询，修改，删除等等的。

三层对应的包：

界面层： controller包(servlet)

业务逻辑层：service包(XXXService类)

数据访问层：das包（XXXDao类）

三层中类的交互

用户使用界面层🡪业务逻辑层🡪数据访问层（持久层）🡪数据库(mysql)

三层对应的处理框架

界面层---servlet---springmvc(框架)

业务逻辑层----service类---spring（框架）

数据访问层---dao类---mybatis(框架)

2. 框架

框架是一个舞台，一个模板

模板：

1. 规定好了一些条款，内容。
2. 加入自己的东西

框架是一个模板

1. 框架中定义好了一些功能。这些功能是可用的。
2. 可以加入项目中自己的功能，这些功能可以利用框架中写好的功能。

框架是一个软件，半成品的软件，定义好了一些基础功能，需要加入你的功能就是完整的。基础功能是可重复使用的，可升级的。

框架的特点：

1. 框架一般不是全能的，不能做所有事情
2. 框架是针对某一个领域有效，特长在某一方面，比如mybatis做数据库操作强，但他不能做其他的。
3. 框架是一个软件

Mybatis框架

一个框架，早期叫做ibatis，代码在github。Mybatis是sql映射框架

Mybatis提供了哪些功能：

1. 提供了创建Connection, Statement, ResultSet的能力， 不用开发人员创建这些对象了
2. 提供了执行sql语句的能力，不用你执行sql
3. 提供了循环sql，把sql的结果转为java对象，List集合的能力
4. 提供了关闭资源的能力，不用你关闭Connection, Statement, ResultSet

开发人员做的是：提供sql语句

最后是：开发人员提供sql语句—mybatis处理sql—开发人员得到List集合或java对象（表中的数据）

总结：

Mybatic是一个sql映射框架，提供的数据库的操作能力，增强的JDBC，

使用mybatis让开发人员集中精力写sql就可以了，不必关心Connection,Statement,ResultSet的创建，销毁，Sql的执行。

第二章

主要类的介绍

1. Resources: mybatis中的一个类，负责读取主配置文件

InputStream in = Resources.getResourceAsStream(config);

1. SqlSessionFactoryBuilder: 创建SqlSessionFactory对象，

SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();

// 创建SqlSessionFactory对象

SqlSessionFactory factory = builder.build(in);

1. SqlSessionFactory：重量级对象，程序创建一个对象好事比较长，使用资源比较多。在整个项目中游一个就可以了。是个接口，接口实现类：DefaultSqlSessionFactory.

作用：获取SqlSession对象。SqlSession sqlSession = factory.openSession();

openSession()方法说明：

1. OpenSession()：无参数的，获取的是非自动提交事务的sqlSession对象
2. Opensession(Boolean): openSession(true) 获取自动提交事务的sqlSession,

openSession(false) 非自动提交事务的sqlSession对象

1. SqlSession接口：定义了操作数据的方法， 例如 selectOne(), selectList(), insert(), update(), delect(), commit(), rollback()

SqlSession接口的实现类DefaultSqlSession。

使用要求：SqlSession对象不是线程安全的，需要在方法内部使用，在执行sql语句之前，使用openSession()获取SqlSession，在执行完sql语句后，需要关闭它，执行SqlSession.close()，这样能保证他的使用时线程安全的。

第三章

1. 动态代理：使用SqlSession.getMapper(dao接口.class)，获取这个dao接口的对象

2. 出入参数：从java代码中把数据传入到mapper文件的sql语句中。

1）parameterType: 写在mapper文件中的一个属性。表示dao接口中方法的参数的数据类型。例如StudentDao接口

Public Student selectStudentById(Integer id);

2） 一个简单类型的参数：

简单类型： mybatis把java的基本数据类型和String都叫简单类型

在mapper文件获取简单类型的一个参数的值，使用 #{任意字符}

接口：Public Student selectStudentById(Integer id);

Mapper：select id, name, email, age from student where id = #{id}

3） 多个参数，使用@Param命名参数

接口 public List<Student> selectMulitParam(@Param(“myname”) String name, @Param(“myage”) Integer age)

使用 @Param(“参数名”) String name

Mapper文件：

<select>

Select \* from student where name = #{myname} or age = #{myage}

</select>

1. 多个参数，使用java对象

语法 #{属性名}

5）#和$

select id, name, email, age from student where id = #{id}

#的结果： select id, name, email, age from student where id = ?

select id, name, email, age from student where id = ${id}

$的结果： select id, name, email, age from student where id = 1001

使用的Statement对象执行sql，效率比PreparedStatement低，会有sql注入问题。

$：可以替换表名或者列名，你能确定数据是安全的。可以使用$

#和$区别

1. #使用?在sql语句中做占位的，使用PreparedStatement执行sql，效率高
2. #能够避免sql注入，更安全
3. $不使用占位符，是字符串连接方式，使用Statement对象执行sql，效率低
4. $有sql注入的风险，却反安全性
5. $可以替换表名或者列名

3. mybatis的输出结果

Mybatis执行了sql语句，得到java对象。

1. resultType结果类型， 指sql语句执行完毕后，数据转为的java对象。Java类型是任意的。

resultType结果类型：1. 类型的全限定名称 2. 类型的别名，例如 java.lang.Integer别名是int

处理方式：

1. mybatis执行sql语句，然后mybatis调用类的无参数构造方法，创建对象。
2. mybatis把ResultSet指定列值付给同名的属性。

<select id="selectMultiPosition" resultType="org.example.domain.Student">

select id, name, email, age from student

</select>

对等的jdbc

ResultSet rs = executeQuery(“select id, name, email, age from student”);

While (rs.next()) {

Student student = new Student();

Student.setId(rs.getInt(“id”));

Student.setName(rs.getString(“name”));

}

2）定义自定义类型的别名

1. 在mybatis主配置文件中定义，使用<typeAlias>定义别名
2. 可以在resultType中使用自定义别名

3）resultMap：结果映射，指定列名和java对象的属性对应关系。

1）你自定义列值赋值给哪个属性

2）当你的列名和属性名不一样时，一定使用resultMap

resultMap和resultType不要一起使用，二选一