回顾：

SSM　[Spring SpringMVC　MyBtis]

SSH　[Struts Spring Hibernate]

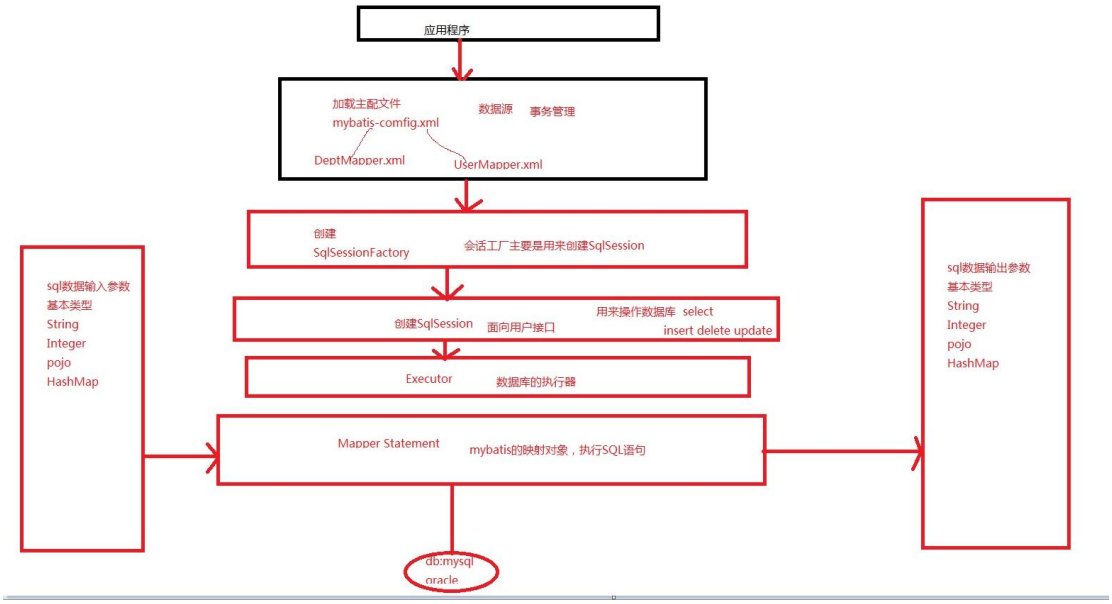
ORM【Object　Relation　Mapping】

JDBC 编程步骤：

1. 加载数据驱动　　Class.forname()
2. 创建并获得连接　Connection conn = DriverManager.getConnection()
3. 创建 SQL 的执行平台 Statement stmt = conn.createStatement()
4. 设置 SQL 语句　　sql = “select　* from emp where last_name = ? and pass = ? ”
5. 设置 SQL 语句的参数
6. 执行 SQL 语句　stmt.executeQuery(sql)
7. 处理结果
8. 释放资源

JDBC 的缺点：

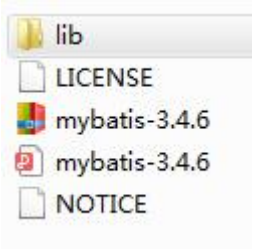
1. 频繁的创建和释放资源，可以通过连接池解决
2. Sql 语句所在的代码中存在硬编码，造成维护困难,sql 语变动需要改变 java 代码
3. PreparedStatement 中的占位符存在硬编码 因为 where 条件不定的，可以多，可以少，代码维护不方例
4. 对结果集的处理存在码编码 sql 语句的变化会导致解析代码的变化，不易维护

1. 

Mybatis 下载

https://github.com/mybatis

Jar 包



JDBC 编程步骤:

1. 加载数据库驱动
2. 创建并获取连接
3. 创建 SQL　　执行平台
4. 设置 SQL 语句
5. 设置 SQL 语句中的参数
6. 执行 SQL 语句
7. 处理结果
8. 释放资源

JDBC 的缺点：

1. 数据的连接创建，释放资源频繁，从而影响系统的性能，可以通过连接池解决
2. Sql 语句在代码中存在硬编码，造成维护代码困难，SQL 变动需要改 JAVA 代码
3. prepareStatement 中的占位符参数存在硬编码，因为 where 条件是不定的，可以多，也可以少，所以修改代码不方便，系统维护不容易
4. 对结果集处理存在硬编码，sql 变化就会导致解析代码的变化，维护不易

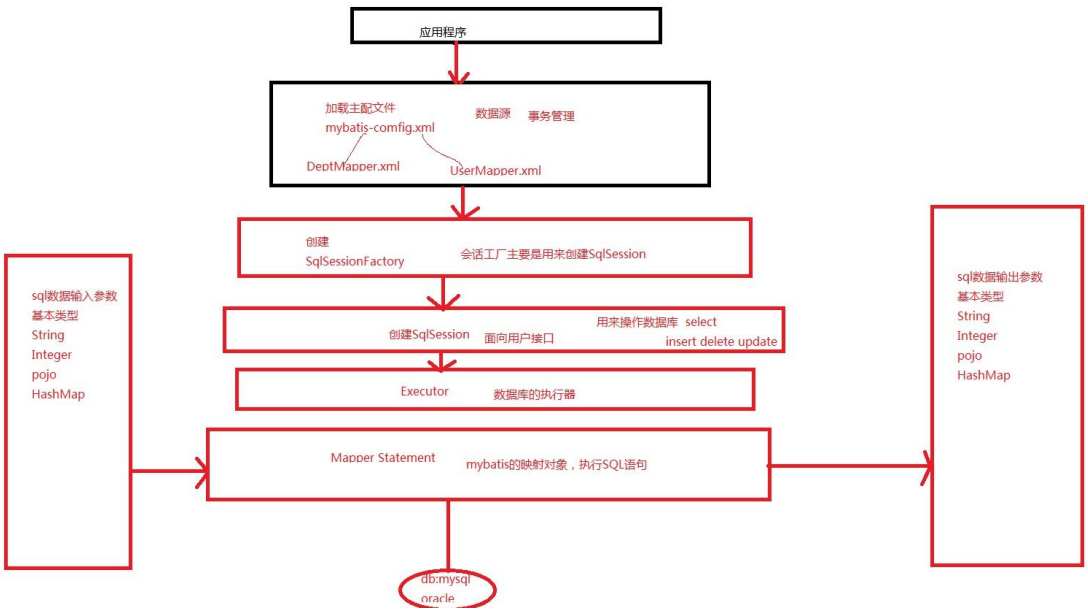
一、MyBatis

MyBatis 本是 apache 的一个开源项目 iBatis, 2010 年这个项目由 apache software foundation 迁移到了 google code，并且改名为 MyBatis 。2013 年 11 月迁移到 Github。
iBATIS 一词来源于“internet”和“abatis”的组合，是一个基于 Java 的持久层框架。iBATIS 提供的持久层框架包括 SQL Maps 和 Data Access Objects（DAO）

MyBatis 消除了几乎所有的 JDBC 代码和参数的手工设置以及对结果集的检索。 MyBatis 可以使用简单的 XML 或注解用于配置和原始映射，将接口和 Java 的 POJO（ Plain Old Java Objects，普通的 Java 对象）映射成数据库中的记录。
mybatis 提供一种“半自动化”的 ORM 实现。这里的“半自动化”，是相对 Hibernate 等提供了全面的数据库封装机制的“全自动化”ORM 实现而言，“全自动”ORM 实现了 POJO 和数据库表之间的映射，以及 SQL 的自动生成和执行。而 mybatis 的着力点，则在于 POJO 与 SQL 之间的映射关系。

1. mybatis 是一个基于 Java 的持久层框架，它对 JDBC 操作数据库的过程进行封装，使我们开发者只需关注 SQL 本身，而不需要花费大量的时间去处理连接，和 sql 结果
2. Mybatis 通过 XML 可以使用简单的 XML 或注解用于配置和原始映射，将接口和 Java 的 POJO（ Plain Old Java Objects，普通的 Java 对象）映射成数据库中的记录。
3. 由 mybatis 框架执行 SQL 语句并将结果映射成 java 对象并返回

二、Mybatis 框架结构



三、MyBatis 下载

1. Mybatis 的代码由 github 管理

下载地址：https://github.com/mybatis/mybatis-3/releases

2.Jar 包

4. 案例

根据部门的编号，查询部门的信息
添加部门
更新部门
删除部门

5. 使用 mybatis 的步骤

1. 创建一个 java 工程
2. 导入 jar 包，包括核心包，依赖包，数据库驱动
3. 以 classpath 下创建 log4j.properties　mybatis 默认使用 log4j 输出日志信息
4. 创建 mybatis 主配文件

四、JUnit

JUnit 是一个 Java 语言的单元测试框架。它由 Kent Beck 和 Erich Gamma 建立，逐渐成为源于 Kent Beck 的 sUnit 的 xUnit 家族中最为成功的一个。 JUnit 有它自己的 JUnit 扩展生态圈。多数 Java 的开发环境都已经集成了 JUnit 作为单元测试的工具。

JUnit 是由 Erich Gamma 和 Kent Beck 编写的一个回归测试框架（regression testing framework）。Junit 测试是程序员测试，即所谓白盒测

试，因为程序员知道被测试的软件如何（How）完成功能和完成什么样（What）的功能。Junit 是一套框架，继承 TestCase 类，就可以用 Junit 进行自动测试了。

```
public class Test {
        //会话工厂
    private SqlSessionFactory    sqlSessionFactory;

    @Before
    public void createSqlSessionFactory()
    {
        //加载配置文件
          String resource ="mybatis-config.xml";
          try {
            InputStream in = Resources.getResourceAsStream(resource);
            //使用 SqlSessionFactoryBuilder()从 xml 配置文件中创建 sqlSessionFactory
            sqlSessionFactory = new SqlSessionFactoryBuilder().build(in);
      } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      }

    }
    //根据部门的编号查询部门信息
    @org.junit.Test
   public void    testQueryDeptByDeptno() {
          SqlSession      session = null;
          try{
          session = sqlSessionFactory.openSession();
          Dept dept = session.selectOne("abc.queryDeptByDeptno",40);
          System.out.println(dept);
          } catch(Exception e)
          {
              e.printStackTrace();
          }finally{
              if(session !=null)
              {
                  session.close();
              }
          }

   }
    @org.junit.Test
      public void    testQueryDeptByDname() {
            SqlSession      session = null;
```

```
                try{
                session = sqlSessionFactory.openSession();
                List<Dept> list = session.selectList("abc.queryDeptByDname","I");
                System.out.println(list);
                } catch(Exception e)
                {
                    e.printStackTrace();
                }finally{
                    if(session !=null)
                    {
                        session.close();
                    }
                }

        }
}
```

五、#{}和 ${}

#{}表示一个占位符，通过#{}可以实现 prepareStatement 向占位符设置值，自动进行 java 类型和 jdbc 类型的转换,#{}可以有效防止 SQL 注入,#{}可以接收基本类型或 pojo 类型。

如果参数是基本类型的，#{}括号中的内容可以是 value,或任意内容

${} 表示的是拼接 SQL 字符串，通过 ${}可以将参数传入的内容拼接在 SQL 中且不进行 JDBC 的类型转换 ${}可以接收基本类型或 pojo 类型，

如果参数是基本类型的 ${}括号内只能是 value

六、parameterType 和 ResultType

1. **parameterType: 指定输入参数的类型，通过 ognl 从输入对象的参数中获取参数的值，拼接在 SQL 中**
2. **resultType:指定输出结果的类型， 将查询结果的一行记录数据映射为 resultType 所指定的类型对象**
3. **selectOne:查询一条记录**
4. **selectList：可以查询一条或多条记录**

七、MyBatis 配置文件分析

1）主配置文件可以随意命名，其主要完成以下几个功能：

2）Mybatis-config.xml　SqlMapConfig.xml

1. 注册 DB 连接四要素的属性文件
2. 注册实体类的全限定类名的别名
3. 配置 MyBatis 运行环境，即数据源与事务管理器

4. 注册 SQL 映射文件

1)SQL 映射文件中的 元素:

A)MyBatis 配置:

SqlMapConfig.xml,这个文件为 MyBatis 的主配置文件,配置了 mybatis 的运行环境.

Mapper.xml 文件是 sql 映射文件 ,文件中配置了操作数据库的 SQL 语句，这个文件要在 SqlMapConfig.xml 中加载.

B) 通过 mybatis 环境配置，构造 SqlSessionFactory 即会话工厂

C) 由会话工厂，创建 SqlSession,即会话，用会话来操作数据库

D) Mybatis 底层定义了一个执行 executor 接口来操作数据库，executor 接口有两个实现，一个基本执行器，一个缓存执行器

E) Mapped statement 也是 mybatis 的一个底层封装对象,它包括 mybatis 配置信息及 sql 语句的映射,mapper.xml 这个文件中,一个 sql 对应一个 mapped Statement 对象,sql 的 id 即 mapped Statement 的 ID,也可以认为它一个方法名

F) mapped Statement 对 sql 执行输入参数进行定义，包括基本类型,String ,Integer pojo，hashmap 等.通过 mapped Statement 在执行 sql 前将输入的 java 对象映射到 sql 中,输入参数就相当于 jdbc 中对 preparedstatement 参数设置

G) mapped Statement 对输出结果进行定义，包括基本类型,String ,Integer pojo，hashmap 等，通过 mapped Statement 在执行 sql 后将输出的对象映射到 java 对象中,输出结果映射过程相当于 JDBC 中对结果集的解析处理过程

MAPPER 映射文件分析

```
<mapper namespace="com.chinasoft.entity">
    <select  id="queryDeptByDeptno"  parameterType="int"
resultType="com.chinasoft.entity.Dept">
      select * from dept where deptno =#{deptno}
    </select>
```

namespace：命名空间，类似于包名

select：存放查询的 SQL 语句

id：该标签的唯一标示符

parameterType：传入参数类型

resultType：返回值类型

#{}占位符 #{} ognl 表达式

2)主配置文件：SqlMapConfig.xml

environments：开放环境，用 default 指定当前的环境
mapper resource：加载 SQL 映射文件

1.resultType ：不论返回值是一个 Student 对象，或多个 Student 对象，resultType 都是"com.chinasoft.entity.Student"　session.selectone() session.selectlist()

2.若 XML 中没有提示，可按下列操作增加提示：

①下载 XML 头文件中的 xxx.dtd 文件
②windows-->preferences-->搜索 Xml-->XML Catalog-->Add:Loacation 选 File System 找到下载的 dtd，并在 Key 输入 xml 头文件中的 public

1、MyBatis 主配置文件 mybatis.xml

```
<?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>
        <!-- 这里写配置内容 -->
</configuration>
```

2、MyBatis 映射文件 mapper.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="命名空间">
        <!-- SQL语句 -->
</mapper>
```

3、MyBatis 约束文件 是约束xml规范的。常见的有2种*.dtd　*.xsd
MyBatis的约束文件位置，都放在jar包里面
org.apache.ibatis.builder.xml.mybatis-3-config.dtd
org.apache.ibatis.builder.xml.mybatis-3-mapper.dtd

3.增删改，必须加上 session.commit()

如果传入的是对象类型，如 parameterType="org.chinasoft.entity.Student"
则可以通过 OGNL,#{stuNo},#{stuName},#{stuAge},#{graName}

4.如果 parameterType 传入的是简单类型

（8 个基本类型+String），则#{任意名} byte char int float double boolean short long
8 个基本：byte,short,int,long,char,float,double,boolean 如果 parameterType 传入的是对象类型，则#{对象的属性名}

八、mybatis 开发过程小结

1、编写 SqlMapConfig.xml
2、编写 mapper.xml
定义了 statement
3、编程通过配置文件创建 SqlSessionFactory
4、通过 SqlSessionFactory 获取 SqlSession
5、通过 SqlSession 操作数据库
如果执行添加、更新、删除需要调用 SqlSession.commit()
6、SqlSesion 使用完成要关闭

parameterType:

指定输入参考数的类型，通过 ognl 从输入对象的参数中获取参数的值，拼接在 sql 中

resultType:

指定输出结果的类型，将查询结果的一行数据映射为 resultType 所指定的类型对象

selectOne:查询一条记录

Selectlist 可以查一条或多少记录

Mybatis

九、Mybatis 的全局配置文件

1、mybatis-config.xml 是 mybatis 的全局配置文件，配置内容如下：

properties（属性）
settings（全局配置参数）
typeAliases（类型别名）
typeHandlers（类型处理器）
objectFactory（对象工厂）
plugins（插件）
environments（环境集合属性对象）
--environment（环境子属性对象）
--transactionManager（事务管理）
--dataSource（数据源）
mappers（映射器）

2、properties

将数据库连接参数单独配置在 db.properties 中，放在类路径下。这样只需要在 SqlMapConfig.xml 中加载 db.properties 的属性值。这样在 SqlMapConfig.xml 中就不需要对数据库连接参数硬编码。
将数据库连接参数只配置在 db.properties 中，原因：方便对参数进行统一管理，其它 xml 可以引用该 db.properties。
使用示例：
db.properties

2、properties

将数据库连接参数单独配置在db.properties中，放在类路径下。这样只需要在SqlMapConfig.xml中加载db.properties的属性值。这样在SqlMapConfig.xml中就不需要对数据库连接参数硬编码。

将数据库连接参数只配置在db.properties中，原因：方便对参数进行统一管理，其它xml可以引用该db.properties。

使用示例：

db.properties

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

相应的SqlMapConfig.xml

```
<properties resource="db.properties"/>
        <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>
```

注意： MyBatis 将按照下面的顺序来加载属性：
首先在 properties 标签中指定的属性文件首先被读取。
然后会读取 properties 元素中 resource 或 url 加载的属性，它会覆盖已读取的同名属性。
最后读取 parameterType 传递的属性，它会覆盖已读取的同名属性。
常用做法：
不要在 properties 元素体内添加任何属性值，只将属性值定义在外部 properties 文件中。
在 properties 文件中定义属性名要有一定的特殊性，如：XXXXX.XXXXX.XXXX 的形式，就像 jdbc.driver。这样可以防止和 parameterType 传递的属性名冲突，从而被覆盖掉。

Mybatis 开发 DAO 模式

IUserDAO ---UserDaoImpl
通过 mapper 代理的方式来实现，程序员只需要写出 dao 接口，dao 实现由 mybatis 完成

Mapper 的开发规范

1. mapper.xml 中的 namespace 指定为 mapper 的接口全类名
2. mapper.xml 中的 ID 就是 mapper 接口中的方法名
3. mapper.xml 中的 parameterType 和 mapper 接口方法输入参数类型一样
4. mapper.xml 中的 resultType 和 mapper 接口中的方法的返回值类一样