Mybatis介绍及环境配置（重点）

一、MyBatis 框架概述

mybatis 是一个优秀的基于 java 的持久层框架，它内部封装了 jdbc，使开发者只需要关注 sql 语句本身， 而不需要花费精力去处理加载驱动、创建连接、创建 statement 等繁杂的过程。

mybatis 通过XML或注解的方式将要执行的各种 statement 配置起来，并通过 java 对象和statement 中 sql 的动态参数进行映射生成最终执行的 sql 语句，最后由 mybatis 框架执行 sql 并将结果映射为 java 对象并返回。

采用 ORM 思想解决了实体和数据库映射的问题，对 jdbc 进行了封装，屏蔽了jdbc api 底层访问细节，使我们不用与 jdbc api 打交道，就可以完成对数据库的持久化操作。

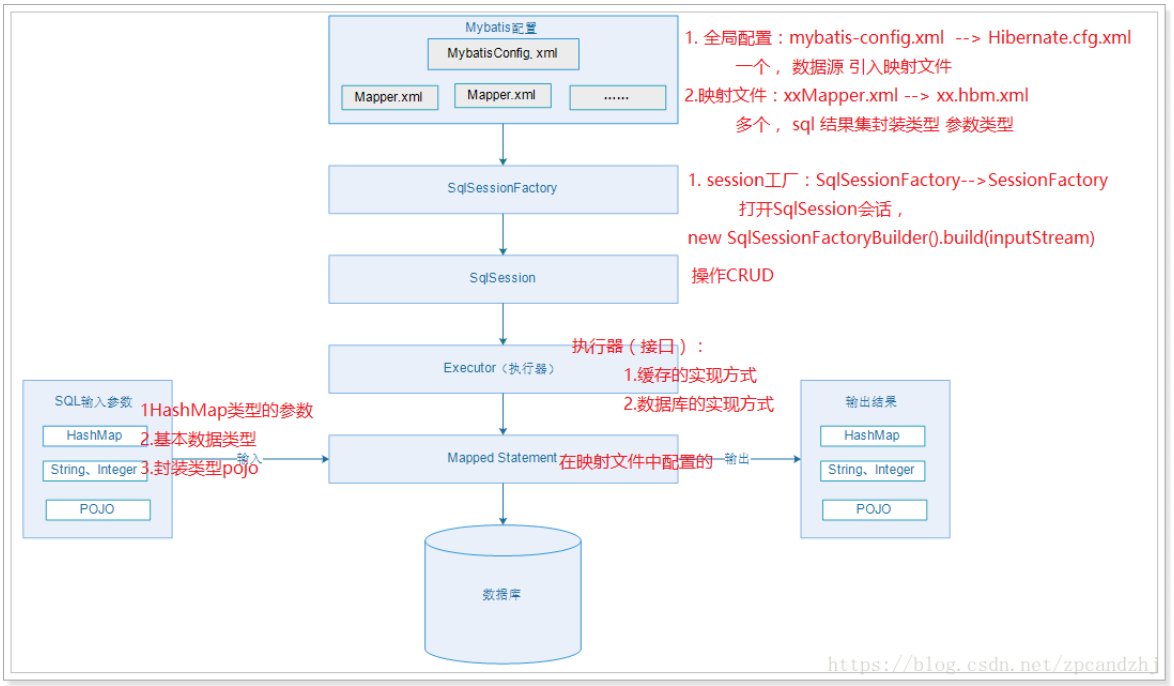
1、ORM 思想

Object Relational Mapping，对象关系映射

（1）实体类和数据库表一一对应，实体类属性和表中字段一一对应
（2）不需要直接操作数据库表，只需操作表对应的实体类对象即可

二、jdbc缺点分析

三、整体结构



四、环境搭配（重点）

环境搭建的注意事项：（重点）

第一个：创建IUserDao.xml 和 IUserDao.java时，在Mybatis中它把持久层的操作接口和映射文件也叫做Mapper，

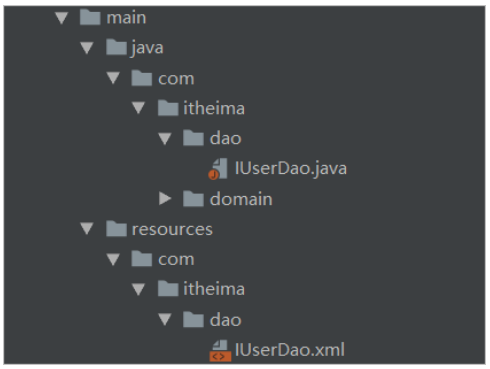
所以：IUserDao 和 IUserMapper是一样的

第二个：在idea中创建目录的时候，如果要创建多级目录，需要一个一个目录创建

包在创建时：com.itheima.dao它是三级结构

目录在创建时：com.itheima.dao是一级目录

第三个：mybatis的映射配置文件位置必须和dao接口的包结构相同



第四个：映射配置文件的mapper标签namespace属性的取值必须是dao接口的全限定类名

第五个：映射配置文件的操作配置（select），id属性的取值必须是dao接口的方法名

当我们遵从了第三，四，五点之后，我们在开发中就无须再写dao的实现类。

1、创建表和Maven工程

2、引入依赖（pom.xml）

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.1</version>
</dependency>
```

3、全局配置文件（SqlMapconfig.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">

<!-- mybatis的主配置文件 -->
<configuration>
<!-- 配置环境 可以配置多个，default：指定采用哪个环境-->
    <environments default="mysql">
        <!-- 配置mysql的环境-->
        <environment id="mysql">
            <!-- 配置事务的类型-->
            <transactionManager type="JDBC"></transactionManager>
            <!-- 配置数据源（连接池） -->
            <dataSource type="POOLED">
                <!-- 配置连接数据库的4个基本信息 -->
                <property name="driver" value="com.mysql.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://localhost:3306/eesy_mybatis"/>
                <property name="username" value="root"/>
                <property name="password" value="1234"/>
            </dataSource>
        </environment>
    </environments>

      <environment id="development">
        <!-- 事务管理器，JDBC类型的事务管理器 -->
        <transactionManager type="JDBC" />
        <!-- 数据源，池类型的数据源 -->
        <dataSource type="POOLED">
          <property name="driver" value="${driver}" /> <!-- 配置了properties，所以可以直接引用 -->
          <property name="url" value="${url}" />
          <property name="username" value="${username}" />
          <property name="password" value="${password}" />
        </dataSource>
      </environment>
    </environments>


    <!-- 指定映射配置文件的位置，映射配置文件指的是每个dao独立的配置文件 -->
    <mappers>
        <mapper resource="com/itheima/dao/IUserDao.xml"/>
```

```
    </mappers>

  </configuration>
```

4、.配置Map.xml（IUser.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.itheima.dao.IUserDao">
    <!--配置查询所有-->
    <select id="findAll" resultType="com.itheima.domain.User">
       select * from user
    </select>
</mapper>
```

5、测试

第一种:

第一步：读取配置文件

第二步：创建SqlSessionFactory工厂

第三步：创建SqlSession

第四步：创建Dao接口的代理对象

第五步：执行dao中的方法

第六步：释放资源

```
public static void main(String[] args)throws Exception {
    //1.读取配置文件
    InputStream in = Resources.getResourceAsStream("SqlMapConfig.xml");
    //2.创建SqlSessionFactory工厂
    SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
    SqlSessionFactory factory = builder.build(in);
    //3.使用工厂生产SqlSession对象
    SqlSession session = factory.openSession();
    //4.使用SqlSession创建Dao接口的代理对象
    IUserDao userDao = session.getMapper(IUserDao.class);
    //5.使用代理对象执行方法
    List<User> users = userDao.findAll();
    for(User user : users){
       System.out.println(user);
    }
    //6.释放资源
    session.close();
    in.close();
}
```

第二种:

1)构建sqlSessionFactory（MybatisTest.java）

```
       // 指定全局配置文件
      String resource = "mybatis-config.xml";
      // 读取配置文件
      InputStream inputStream = Resources.getResourceAsStream(resource);
```

```
// 构建sqlSessionFactory
SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
```

2)打开sqlSession会话，并执行sql

```
    // 获取sqlSession
SqlSession sqlSession = sqlSessionFactory.openSession();
// 操作CRUD，第一个参数：指定statement，规则：命名空间+“.”+statementId
// 第二个参数：指定传入sql的参数：这里是用户id
User user = sqlSession.selectOne("MyMapper.selectUser", 1);
System.out.println(user);
```

6、缺点分析