MyBatis

一、JDBC开发的弊端

JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用java编写的类和接口组成。传统JDBC操作数据需要经过哪几个步骤：

1. 使用JDBC编程需要连接数据库，注册驱动和数据库信息
2. 操作Connection，打开 Statement 对象 。
3. 通过Statement执行SQL， 返回结果到ResultSet对象。
4. 使用ResultSet读取数据，然后通过代码转化为具体的POJO对象。
5. 关闭数据库的相关资源。

那么，如果我们还是继续使用传统的JDBC方式来操作数据库会存在哪些**弊端**呢？

* 工作量相对较大。我们需要先连接，然后处理JDBC底层事务，处理数据类型。我们还需要操作Connection对象、Statement对象和ResultSet对象去拿到数据，并准确的关闭它们
* 我们要对JDBC编程可能产生的异常进行捕捉处理并正确关闭资源。

对于一个JDBC操作简单的SQL尚且如此的复杂，何况是更为复杂的应用呢？所以这种模式很快的就被一些新的方法所取代了。于是ORM就出现了。不过，我们要知道所有的ORM模型都是基于对JDBC的进行封装，不同的ORM模型对JDBC封闭的强度是不一样的。

二、什么是ORM

由于JDBC存在的缺陷，因此在实际工作中很少使用JDBC进行操作数据库的编程。于是提出了对象关系映射（Object Relational Mapping）简称 ORM，或者O/RM，或者 O/R mapping

1.1对象关系映射（Object Relational Mapping，简称ORM）是一种为了解决面向对象与关系数据库存在的互不匹配的现象的技术。 简单的说，ORM是通过使用描述对象和数据库之间映射的元数据，将java程序中的对象自动持久化到关系数据库中。本质上就是将数据从一种形式转换到另外一种形式。 这也同时暗示者额外的执行开销；然而，如果ORM作为一种中间件实现，则会有很多机会做优化，而这些在手写的持久层并不存在。 更重要的是用于控制转换的元数据需要提供和管理；但是同样，这些花费要比维护手写的方案要少；而且就算是遵守ODMG规范的对象数据库依然需要类级别的元数据。ORM是随着面向对象的软件开发方法发展而产生的。面向对象的开发方法是当今企业级应用开发环境中的主流开发方法，关系数据库是企业级应用环境中永久存放数据的主流数据存储系统。对象和关系数据是业务实体的两种表现形式，业务实体在内存中表现为对象，在数据库中表现为关系数据。内存中的对象之间存在关联和继承关系，而在数据库中，关系数据无法直接表达多对多关联和继承关系。因此，对象-关系映射(ORM)系统一般以中间件的形式存在，主要实现程序对象到关系数据库数据的映射。

**1.2**目前流行的 ORM 产品(ORM框架);众多厂商和开源社区都提供了持久层框架的实现，常见的有：

Java系列：Apache OJB （http://db.apache.org/ojb/）   
      Hibernate （http://www.hibernate.org）   
      **iBatis** （http://www.ibatis.com）

三、什么是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参考资料官网：<https://mybatis.github.io/mybatis-3/zh/index.html>

四、MyBatis优缺点

**一、MyBatis框架的优点：**

1. 与JDBC相比，减少了50%以上的代码量。

2. MyBatis是最简单的持久化框架，小巧并且简单易学。

3. MyBatis相当灵活，不会对应用程序或者数据库的现有设计强加任何影响，SQL写在XML里，从程序代码中彻底分离，降低耦合度，便于统一管理和优化，并可重用。(半自动化)

4. 提供XML标签，支持编写动态SQL语句。

5. 提供映射标签，支持对象与[数据库](http://lib.csdn.net/base/mysql)的ORM字段关系映射。

**二、MyBatis框架的缺点：**

1. SQL语句的编写工作量较大，尤其是字段多、关联表多时，更是如此，对开发人员编写SQL语句的功底有一定要求。

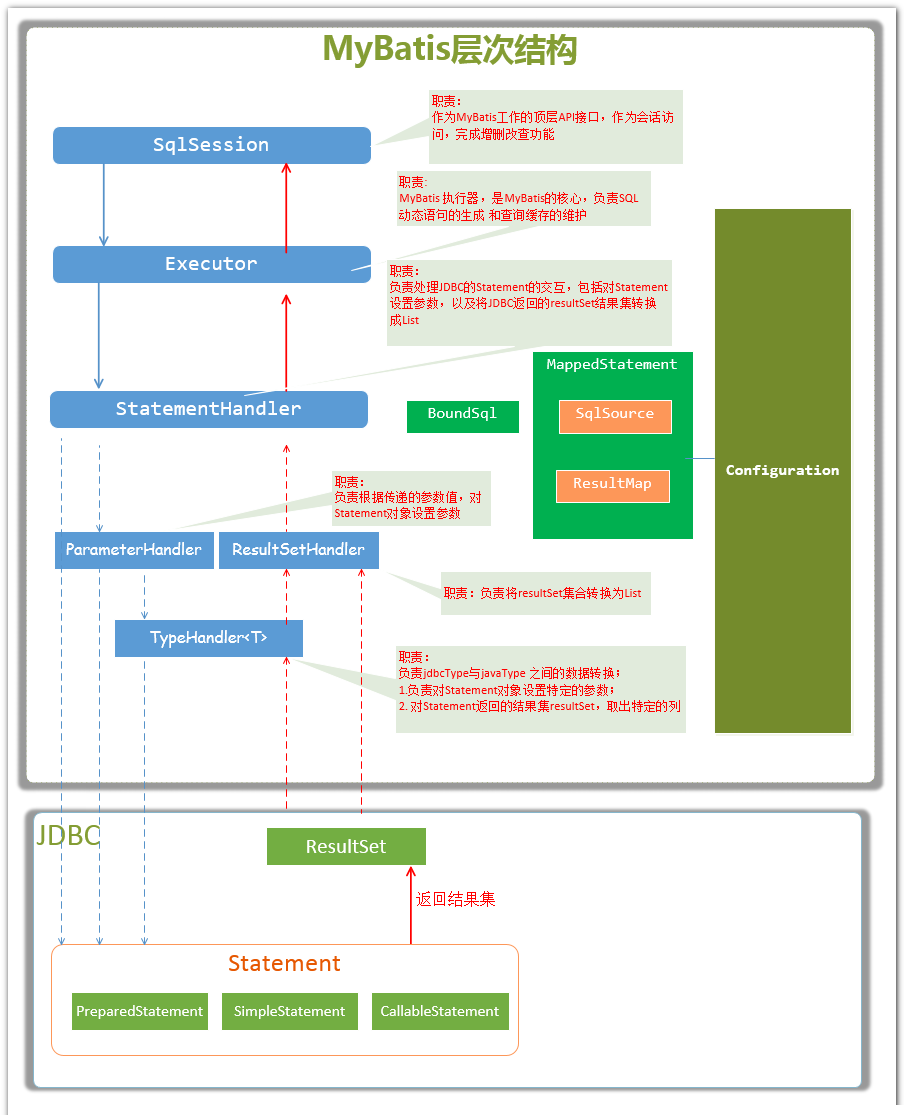
2. SQL语句依赖于数据库，导致数据库移植性差，不能随意更换数据库。

**三、MyBatis框架适用场合：**

MyBatis专注于SQL本身，是一个足够灵活的DAO层解决方案。

对性能的要求很高，或者需求变化较多的项目，如互联网项目，MyBatis将是不错的选择。

五、MyBatis的主要成员



**Configuration**        MyBatis所有的配置信息都保存在Configuration对象之中，配置文件中的大部分配置都会存储到该类中

**SqlSession**            作为MyBatis工作的主要顶层API，表示和数据库交互时的会话，完成必要数据库增删改查功能

**Executor**               MyBatis执行器，是MyBatis 调度的核心，负责SQL语句的生成和查询缓存的维护

**StatementHandler** 封装了JDBC Statement操作，负责对JDBC statement 的操作，如设置参数等

**ParameterHandler**  负责对用户传递的参数转换成JDBC Statement 所对应的数据类型

**ResultSetHandler**   负责将JDBC返回的ResultSet结果集对象转换成List类型的集合

**TypeHandler**          负责java数据类型和jdbc数据类型(也可以说是数据表列类型)之间的映射和转换

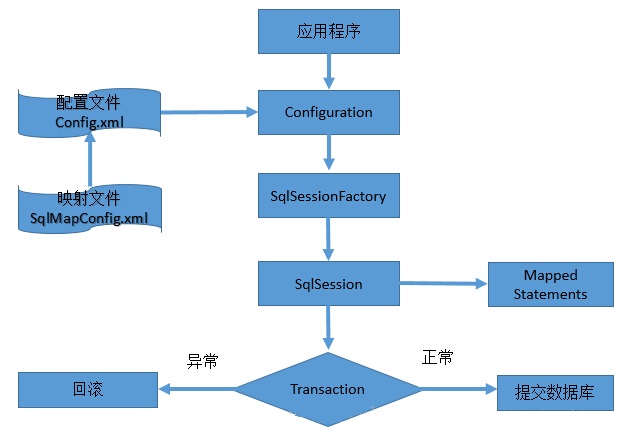
**MappedStatement**  MappedStatement维护一条<select|update|delete|insert>节点的封装

**SqlSource**              负责根据用户传递的parameterObject，动态地生成SQL语句，将信息封装到BoundSql对象中，并返回

**BoundSql**              表示动态生成的SQL语句以及相应的参数信息

以上主要成员在一次数据库操作中基本都会涉及，在SQL操作中重点需要关注的是SQL参数什么时候被设置和结果集怎么转换为JavaBean对象的，这两个过程正好对应StatementHandler和ResultSetHandler类中的处理逻辑。

六、MyBatis的原理



**原理详解：**

MyBatis应用程序根据XML配置文件创建SqlSessionFactory，SqlSessionFactory在根据配置，配置来源于两个地方，一处是配置文件，一处是Java代码的注解，获取一个SqlSession。SqlSession包含了执行sql所需要的所有方法，可以通过SqlSession实例直接运行映射的sql语句，完成对数据的增删改查和事务提交等，用完之后关闭SqlSession。

七、MyBatis核心配置文件说明

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!--  Copyright 2009-2012 the original author or authors.  Licensed under the Apache License, Version 2.0 (the "License");  you may not use this file except in compliance with the License.  You may obtain a copy of the License at  http://www.apache.org/licenses/LICENSE-2.0  Unless required by applicable law or agreed to in writing, software  distributed under the License is distributed on an "AS IS" BASIS,  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  See the License for the specific language governing permissions and  limitations under the License.  -->  <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd">  <configuration>  <!-- 引入外部文件定义的属性，供配置文件使用 -->  <!-- <properties resource="jdbc.properties"/> -->  <!-- 配置数据库环境 【default：指定某一个连接】-->  <environments default=*"development"*>    <!--  每个environment都代表一种连接，  id是唯一标识，可以配置多个environment数据库连接,  使用default来指定确定具体使用哪个连接  -->  <environment id=*"development"*>    <!-- 数据库事务处理方式：分jdbc自己来处理方式和managed处理方式 -->  <transactionManager type=*"JDBC"* />  <!-- 数据源信息 :  配置数据源的管理方式：tpye值[UNPOOLED,POOLED,JNDI]  UNPOOLED:自己管理，不用连接池  POOLED:连接池  JNDI:外部来管理(java命名与目录，一种命名服务与资源关联的技术)  -->  <dataSource type=*"UNPOOLED"*>  <!-- 数据源连接信息，从jdbc.properties文件进行读取，${变量} -->  <property name=*"driver"* value=*"com.mysql.jdbc.Driver"*/>  <property name=*"url"* value=*"jdbc:mysql://localhost:3306/boot"*/>  <property name=*"username"* value=*"root"*/>  <property name=*"password"* value=*"root"*/>  </dataSource>  </environment>  </environments>    <!-- 引入实体映射文件 -->  <mappers>  <mapper resource=*"com/yk/pojo/UserMapper.xml"*/>  </mappers>    </configuration> |