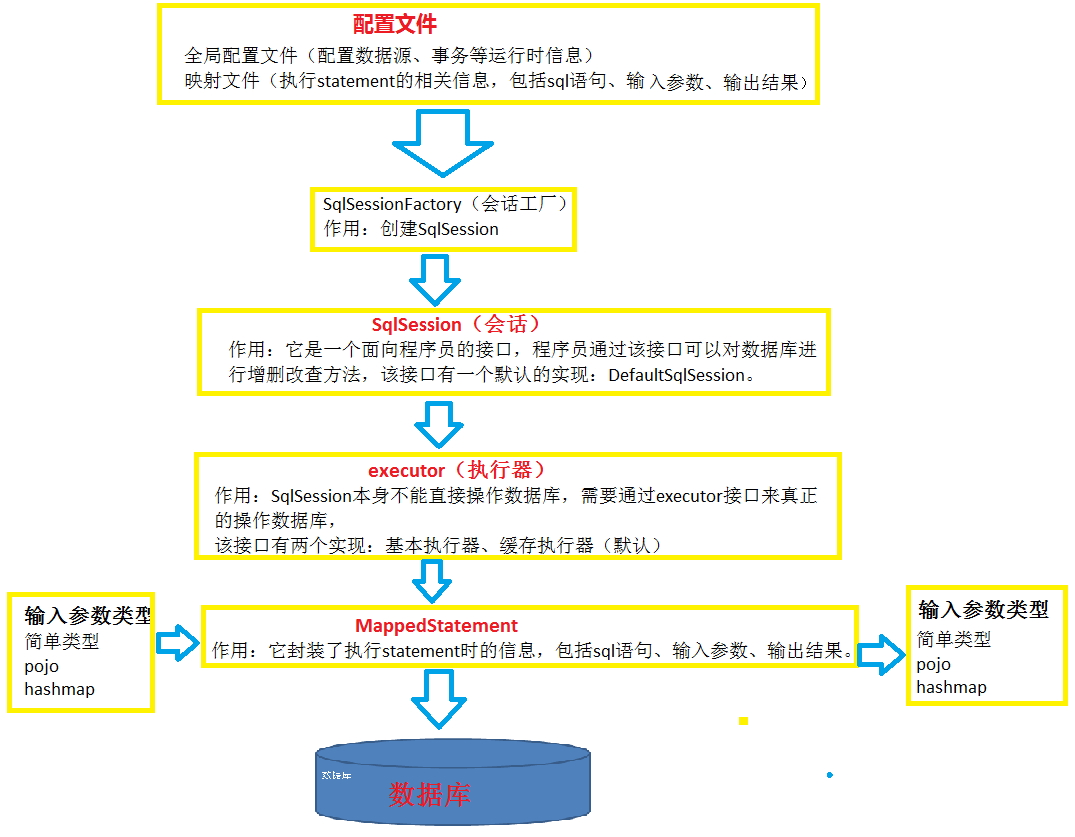
# Mybatis

## 概述

mybatis就是一个封装来jdbc的持久层框架，它和hibernate都属于ORM框架，但是具体的说，hibernate是一个完全的orm框架，而mybatis是一个不完全的orm框架。

Mybatis让程序员**只关注sql本身**，而不需要去关注如连接的创建、statement的创建等操作。

## mybatis架构(重点)



SqlMapConfig.xml，此文件作为mybatis的全局配置文件，配置了mybatis的运行环境等信息。

mapper.xml文件即sql映射文件，文件中配置了操作数据库的sql语句。此文件需要在SqlMapConfig.xml中加载。

通过mybatis环境等配置信息构造SqlSessionFactory即会话工厂

由会话工厂创建sqlSession即会话，操作数据库需要通过sqlSession进行。

mybatis底层自定义了Executor执行器接口操作数据库，Executor接口有两个实现，一个是基本执行器、一个是缓存执行器。

Mapped Statement

1.mybatis一个底层封装对象，它包装了mybatis配置信息及sql映射信息等。

mapper.xml文件中一个sql对应一个Mapped Statement对象，sql的id即是Mapped statement的id。

Mapped Statement对sql执行输入参数进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql前将输入的java对象映射至sql中

2.输入参数映射就是jdbc编程中对preparedStatement设置参数。

3.输出结果映射过程相当于jdbc编程中对结果的解析处理过程。

# jdbc与mybatis

企业开发中，根据项目大小、特点进行技术选型 ，jdbc操作数据库时效率是很高的，jdbc也是技术选型的参考。

## jdbc程序

需要数据库的驱动包：



**public static void** main(String[] args) {

Connection connection = **null**;

PreparedStatement preparedStatement = **null**;

ResultSet resultSet = **null**;

**try** {

//1、加载数据库驱动

Class.*forName*("com.mysql.jdbc.Driver");

//2、通过驱动管理类获取数据库链接

connection = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/mybatis?characterEncoding=utf-8", "root", "root");

//3、定义sql语句 ?表示占位符

String sql = "select \* from user where username = ?";

//4、获取预处理statement

preparedStatement = connection.prepareStatement(sql);

//5、设置参数，第一个参数为sql语句中参数的序号（从1开始），第二个参数为设置的参数值

preparedStatement.setString(1, "王五");

//6、向数据库发出sql执行查询，查询出结果集

resultSet = preparedStatement.executeQuery();

//7、遍历查询结果集

**while**(resultSet.next()){

User user

System.*out*.println(resultSet.getString("id")+" "+resultSet.getString("username"));

}

} **catch** (Exception e) {

e.printStackTrace();

}**finally**{

//8、释放资源

**if**(resultSet!=**null**){

**try** {

resultSet.close();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**if**(preparedStatement!=**null**){

**try** {

preparedStatement.close();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**if**(connection!=**null**){

**try** {

connection.close();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

## jdbc问题总结

1、数据库连接频繁的创建和关闭，缺点浪费数据库的资源，影响操作效率

设想：使用数据库连接池

2、sql语句是硬编码，如果需求变更需要修改sql，就需要修改java代码，需要重新编译，系统不易维护。

设想：将sql语句 统一配置在文件中，修改sql不需要修改java代码。

3、通过preparedStatement向占位符设置参数，存在硬编码（ 参数位置，参数）问题。系统不易维护。

设想：将sql中的占位符及对应的参数类型配置在配置文件中，能够自动输入 映射。

4、遍历查询结果集存在硬编码（列名）。

设想：自动进行sql查询结果向java对象的映射（输出映射）。

## Mybatis解决jdbc编程的问题

1. 数据库链接创建、释放频繁造成系统资源浪费从而影响系统性能，如果使用数据库链接池可解决此问题。

解决：在SqlMapConfig.xml中配置数据链接池，使用连接池管理数据库链接。

1. Sql语句写在代码中造成代码不易维护，实际应用sql变化的可能较大，sql变动需要改变java代码。

解决：将Sql语句配置在XXXXmapper.xml文件中与java代码分离。

1. 向sql语句传参数麻烦，因为sql语句的where条件不一定，可能多也可能少，占位符需要和参数一一对应。

解决：Mybatis自动将java对象映射至sql语句，通过statement中的parameterType定义输入参数的类型。

1. 对结果集解析麻烦，sql变化导致解析代码变化，且解析前需要遍历，如果能将数据库记录封装成pojo对象解析比较方便。

解决：Mybatis自动将sql执行结果映射至java对象，通过statement中的resultType定义输出结果的类型。

# mybatis与hibernate

## 区别

### orm

Mybatis技术特点：

半orm框架（Mybatis在查询关联对象或关联集合对象时，需要手动编写sql来完 成），直接编写SQL语句，把Sql和java代码独立出来；

Hibernate技术特点：

标准的orm框架（使用Hibernate查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取），程序员不需要编写SQL语句。

### 数据库相关性

Mybatis技术特点：

不能支持数据库无关性，即数据库发生变更，要写多套代码进行支持，移植性不好。

Hibernate技术特点：

具有良好的数据库无关性，即数据库发生变化的话，代码无需再次编写

### 学习

Mybatis技术特点：

学习门槛低，只要有SQL基础；但需要写编写结果映射。

Hibernate技术特点：

学习门槛高，需要对数据关系模型有良好的基础，而且在设置OR映射的时候，需要考虑好性能和对象模型的权衡

### Sql优化代码维护

Mybatis技术特点：

写SQL语句相对于比较灵活,并且容易调试，代码维护性更好

Hibernate技术特点：

程序员不能自主的去进行SQL性能优化。

## 应用场景

Mybatis应用场景：

需求多变的互联网项目，例如电商项目。

Hibernate应用场景：

需求明确、业务固定的项目，例如OA项目、ERP项目等。一般适合对进度要求的比较高的中小型项目.

SqlMapConfig.xml（mybatis全局配置文件，名称不固定）

运行环境（数据源、事务）

mapper.xml(配置sql语句), mapper.xml.....

SqlSessionFactory（会话工厂）

作用：创建SqlSession

SqlSession（向面用户的接口，数据库操作方法）

作用：操作数据库

Executor（数据库操作的执行器）

Executor是一个接口，有两个实现 （默认执行器和缓存执行器）

mysql

MappedStatement（mybatis的封装对象）

封装：sql语句(占位符)

输入 映射

java简单类型

pojo类型

hashmap

输出 映射

java简单类型

pojo类型

hashmap