传统JDBC

什么是JDBC

- 是一种用于执行SQL语句的Java API
- JDBC是java访问数据库的基石
- 其他持久化技术都是对jdbc的封装
- Hibernate, MyBatis等(底层依然是JDBC)

JDBC的作用

JDBC规范了:

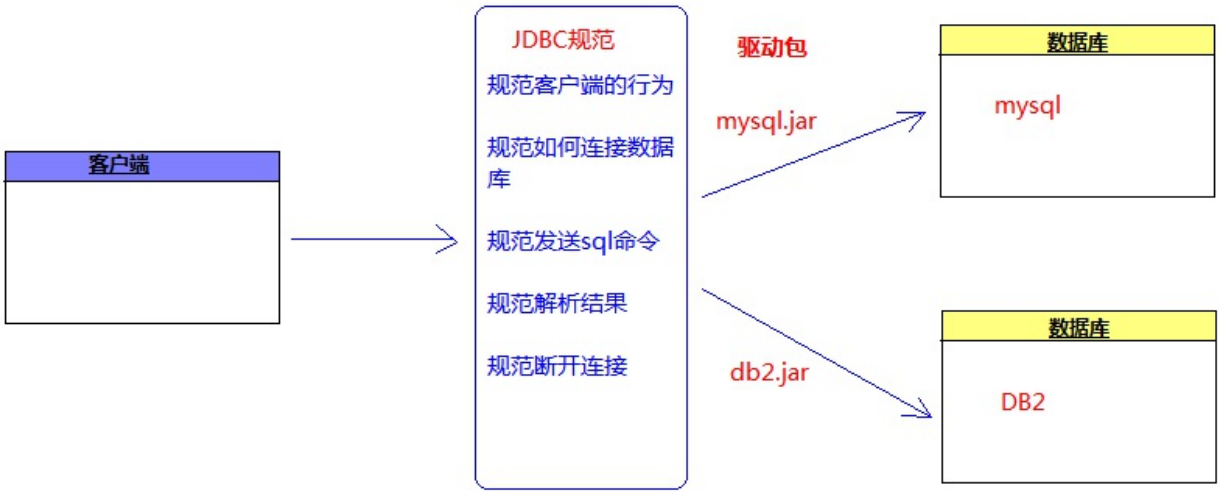
- 如何连接数据库
- 如何发送sql命令
- 如何解析结果集
- 如何断开连接等等操作.

进一步来说:

- 为多种关系数据库提供统一访问
- 为开发者屏蔽了一些细节问题
- 接口的实现由各个数据库厂商来完成--驱动包

因此:

- 遵循了JDBC规范来写代码
- 修改了数据库
- 仅仅需要修改驱动包即可



JDBC操作模板

贾琏欲执事:

1. 加载注册驱动
2. 获取数据库连接对象

3. 获取预编译语句对象
4. 执行sql命令
5. 释放资源

```
@Test
public void testCreateTable() throws Exception {
    String sql = "CREATE TABLE `t_student`(`id` BIGINT PRIMARY KEY AUTO_INCREMENT,"
            + " `name` VARCHAR(255), `age` INT)";
    // 贾琏欲执事
    // 1. 加载注册驱动
    Class.forName("com.mysql.jdbc.Driver");
    // 2. 获取数据库连接对象
    Connection conn = DriverManager
            .getConnection("jdbc:mysql://localhost:3306/javaweb", "root", "admin");
    // 3. 获取语句对象
    Statement st = conn.createStatement();
    // 4. 执行sql命令
    st.executeUpdate(sql);
    // 5. 释放资源,习惯性的后开的先关.
    st.close();
    conn.close();
}
```

```
@Test
public void testGet() throws Exception {
    // 获取id为2的学生信息
    String sql = "select * from t_student where id=?";
    // 贾琏欲执事
    Class.forName("com.mysql.jdbc.Driver");
    Connection conn = DriverManager.getConnection("jdbc:mysql:///javaweb",
            "root", "admin");
    PreparedStatement ps = conn.prepareStatement(sql);
    // 执行之前一定不要忘记给占位符设置值.
    ps.setObject(1, 3L);
    // 一定不要调用父类有参数的方法
    ResultSet rs = ps.executeQuery();
    if (rs.next()) {
        System.out.println(rs.getObject(1));
        System.out.println(rs.getObject(2));
        System.out.println(rs.getObject(3));
    }
    JDBCUtil.close(rs, ps, conn);
}
```

JDBC弊端

1. 开发工作量较大。

```
我们需要先连接，然后处理JDBC底层事务，处理数据类型，
还需要操作Connection对象、Statement/PreparedStatement对象、ResultSet对象去拿数据，
并且还要关闭这些资源，
难以写出高质量、易维护的代码。
```

2. 资源浪费、影响数据库的性能。

```
使用JDBC操作数据库，使用时才连接，不使用就释放，频繁的对数据库操作
```

3. 硬编码,不利于后期系统的维护。

```
不管是程序中的SQL语句、向PreparedStatement对象设置参数还是从ResultSet遍历结果集数据
```

4. 于是ORM框架便出现了，但是归根结底，ORM框架还是基于JDBC编程，只不过是对JDBC的封装形式、强度和方式不同罢了

Hibernate

过渡

- 这个框架大家可能在看面试题的时候会经常看到
- 但是又没有实际敲过代码
- 所以今天准备一个小的演示项目
- 通过Hibernate实现CRUD

hibernate作用

- hibernate是建立在若干POJO通过XML映射文件或注解提供的规则映射到数据库表中的，也就是说，我们可以通过POJO直接操作数据库的数据，它提供的是一种全表映射的模型。

结构分析

```
映射配置文件:主要就是描述POJO与数据库中表的映射关系。
全局配置文件:hibernate.cfg.xml
model
dao
test
```

优点

```
1对JDBC的代码进行了封装，使我们的编程更简便了，不用写SQL语句，提高了开发效率。
2消除了代码的映射规则，也无需在管理数据库连接，全部被分离到了XML或注解里面去配置。
3hibernate使用的是HQL语言，它支持方言配置，方便数据库移植。
4一个会话中，不需要操作多个对象，只要操作Session即可，
关闭资源也只要关闭一个Session即可，当然在Spring的集成使用下这些都会交由Spring来管理
```

缺点

```
1当我们更新时将发送所有的字段，而当我们查询时它也会将我们不想要查询的字段也查询出来，这即是全表映射所带来的麻烦。
2由于有第一点缺点，也就导致了我们无法根据不同的条件组装我们需要的SQL语句。
3hibernate并不能很好的支持存储过程，一大遗憾。
4对多表关联和复杂SQL查询支持稍差，还是要自己写SQL语句，返回的结果，需要自己组装为POJO。
5虽然hibernate使用的是HQL语言查询，但是性能不高。
而当我们的数据量很大或是大型系统时，必定需要优化SQL语句，同样由于第一点缺点，hibernate无法做到。
6由于hibernate的高门槛，要完全掌握并不简单，所以对于一个开始并不熟悉hibernate开发的人，学习时间稍长，开发速度稍慢。
```

Mybatis

由于hibernate自身的缺陷，因此又出现了一个新的ORM框架，即mybatis。
这是一个半自动化的框架，何谓半自动，因为它需要手工编写POJO、SQL和映射关系。

优缺点：
1.sql语句与代码分离，存放于xml配置文件中：
优点：便于维护管理，不用在java代码中找这些语句；

缺点： JDBC方式可以用用打断点的方式调试，但是Mybatis不能，需要通过log4j日志输出日志信息帮助调试，然后在配置文件中修改。

2.用逻辑标签控制动态SQL的拼接：
优点：用标签代替编写逻辑代码；

缺点：拼接复杂SQL语句时，没有代码灵活，拼写比较复杂。

3.查询的结果集与java对象自动映射：
优点：保证名称相同，配置好映射关系即可自动映射或者，不配置映射关系，通过配置列名=字段名也可完成自动映射。
缺点：对开发人员所写的SQL依赖很强。

4.编写原生SQL：
优点：接近JDBC，比较灵活。

缺点：对SQL语句依赖程度很高；并且属于半自动，数据库移植比较麻烦，比如mysql数据库编程Oracle数据库，部分的sql语句需要调整。

选择

hibernate作为一个流行且自动化的ORM框架，编程简单，无需编写SQL语句，
可以使用代码生成工具生成entity及dao层代码，并提供强大的缓存、级联和日志功能。
但是由于其自动化缺陷，导致我们无法控制SQL语句，
使得性能无法得到有效优化，并且其不适应存储过程，
所以hibernate也只适应于对性能要求不是太高且场景不太复杂的系统。

而mybatis作为一个半自动化的框架，由于其高度灵活，
自由控制SQL语句，支持动态SQL语句和存储过程，
同样也可以通过代码生成工具生成entity和dao层，当
然如果自己封装模板的话，对于entity中属性和方法名以及dao、service、controller层简易的增删改查都可以自动生成。
对于大型移动互联网项目以及需要考虑查询性能优化的都可以采用mybatis这种方式。

---

jdbc

传统的JDBC的代码缺陷：
    1.JDBC的代码是重复的.
        抽取模板方法.
    2.结果集的解析是写死了.
        把javabean的属性名作为列名,获取数据之后设置到属性上.(内省)
        前提:属性名和列名要一一匹配.
    3.SQL和java代码没有分离,出现了硬编码.
        可以将SQL写在配置文件中.
我们解决JDBC的缺陷的最优方式,就是mybatis.遵循了ORM思想的框架.

hibernate

```
优点:不用写sql
缺点:全表映射,
```

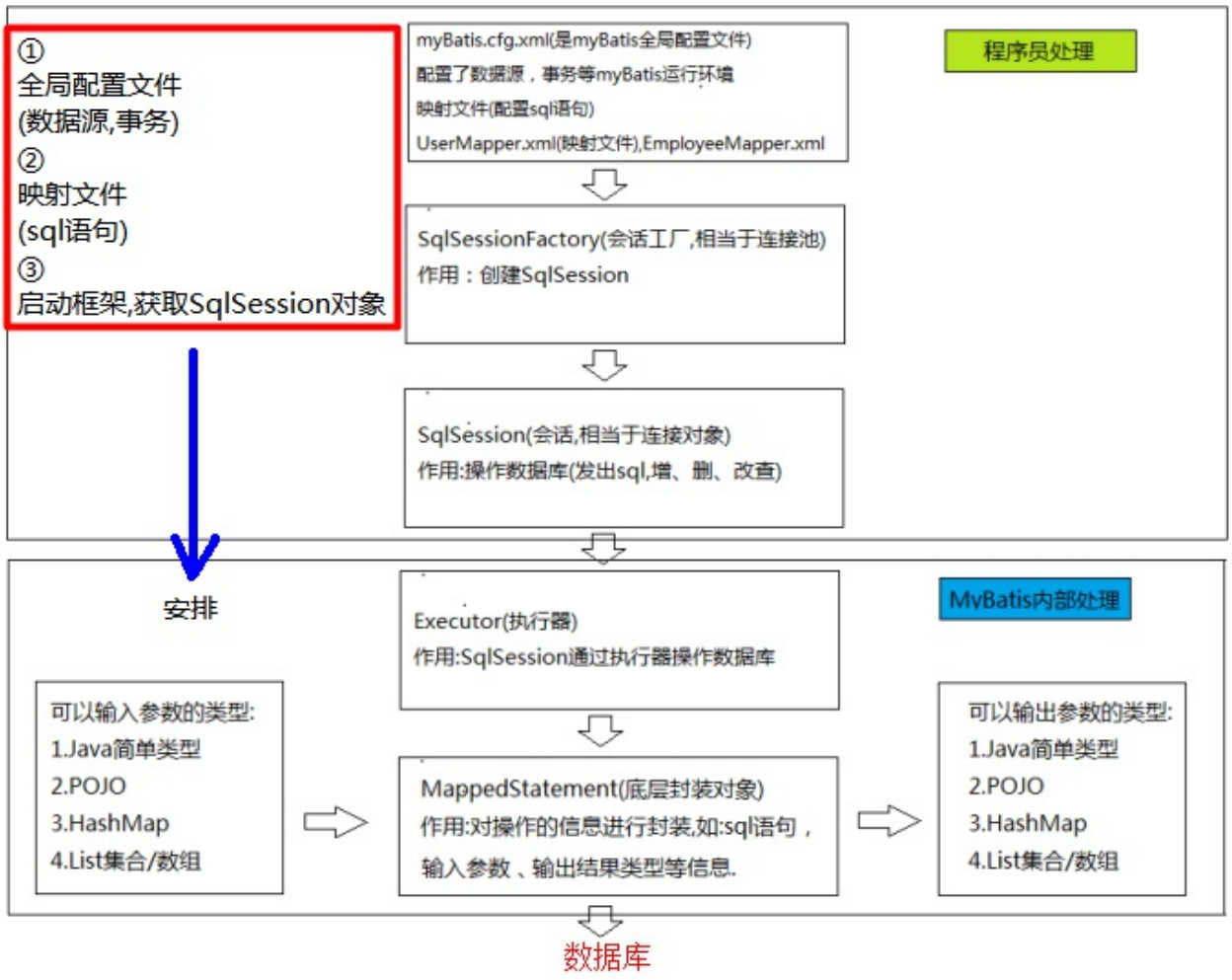
mybatis

```
jdbc和hibernate的折中方式,前身是ibatis,是半自动框架
通过读取mybatis.xml配置文件(包含信息:连接池,事务,关联映射文件)
映射文件包含:sql,pojo,映射关系



生成一个SqlSessionFactory
需要对数据库进行操作时候,通过factory获取一个会话session对象
通过session,告知框架想执行的sql的id和参数
框架就会据此去拼接并执行对应的动态sql,
如果是查询的话,则会根据设定好的resultType/resultMap封装数据
并将封装好的数据返回给调用者
调用者需使用完后,需要关闭session,将连接返回给连接池;

mapper代理
```



贾琏欲执释:
https://www.jianshu.com/p/cedd74292384

Hibernate有三种状态：transient(瞬时状态)，persistent(持久化状态)以及detached(游离状态)
https://www.cnblogs.com/jyh317/p/3666566.html

从jdbc到hibernate,mybatis
https://www.cnblogs.com/lhw1994/p/6759815.html

各自区别
https://www.cnblogs.com/rzqz/p/7266092.html