**Mybatis实现原理深入解析 （转载）**

**原理分析之一：从JDBC到Mybatis**

**1.引言**

本文主要讲解JDBC怎么演变到Mybatis的渐变过程，重点讲解了为什么要将JDBC封装成Mybaits这样一个持久层框架。再而论述Mybatis作为一个数据持久层框架本身有待改进之处。

**2.JDBC实现查询分析**

我们先看看我们最熟悉也是最基础的通过JDBC查询数据库数据，一般需要以下七个步骤：

（1）  加载JDBC驱动

（2）  建立并获取数据库连接

（3）  创建 JDBC Statements 对象

（4）  设置SQL语句的传入参数

（5）  执行SQL语句并获得查询结果

（6）  对查询结果进行转换处理并将处理结果返回

（7）  释放相关资源（关闭Connection，关闭Statement，关闭ResultSet）

## 3.JDBC演变到Mybatis过程

   上面我们看到了实现JDBC有七个步骤，哪些步骤是可以进一步封装的，减少我们开发的代码量。

**第一步优化：连接获取和释放**

**问题描述：**

数据库连接频繁的开启和关闭本身就造成了资源的浪费，影响系统的性能。

**解决问题：**

数据库连接的获取和关闭我们可以使用数据库连接池来解决资源浪费的问题。通过连接池就可以反复利用已经建立的连接去访问数据库了。减少连接的开启和关闭的时间。

**问题描述：**

但是现在连接池多种多样，可能存在变化，有可能采用DBCP的连接池，也有可能采用容器本身的JNDI数据库连接池。

**解决问题：**

我们可以通过DataSource进行隔离解耦，我们统一从DataSource里面获取数据库连接，DataSource具体由DBCP实现还是由容器的JNDI实现都可以，所以我们将DataSource的具体实现通过让用户配置来应对变化。

**第二步优化：SQL统一存取**

**问题描述：**

我们使用JDBC进行操作数据库时，SQL语句基本都散落在各个JAVA类中，这样有三个不足之处：

第一，可读性很差，不利于维护以及做性能调优。

第二，改动Java代码需要重新编译、打包部署。

第三，不利于取出SQL在数据库客户端执行（取出后还得删掉中间的Java代码，编写好的SQL语句写好后还得通过＋号在Java进行拼凑）。

**解决问题：**

我们可以考虑不把SQL语句写到Java代码中，那么把SQL语句放到哪里呢？首先需要有一个统一存放的地方，我们可以将这些SQL语句统一集中放到配置文件或者数据库里面（以key-value的格式存放）。然后通过SQL语句的key值去获取对应的SQL语句。

既然我们将SQL语句都统一放在配置文件或者数据库中，那么这里就涉及一个SQL语句的加载问题。

**第三步优化：传入参数映射和动态SQL**

**问题描述：**

很多情况下，我们都可以通过在SQL语句中设置占位符来达到使用传入参数的目的，这种方式本身就有一定局限性，它是按照一定顺序传入参数的，要与占位符一一匹配。但是，如果我们传入的参数是不确定的（比如列表查询，根据用户填写的查询条件不同，传入查询的参数也是不同的，有时是一个参数、有时可能是三个参数），那么我们就得在后台代码中自己根据请求的传入参数去拼凑相应的SQL语句，这样的话还是避免不了在Java代码里面写SQL语句的命运。既然我们已经把SQL语句统一存放在配置文件或者数据库中了，怎么做到能够根据前台传入参数的不同，动态生成对应的SQL语句呢？

**解决问题：**

第一，我们先解决这个动态问题，按照我们正常的程序员思维是，通过if和else这类的判断来进行是最直观的，这个时候我们想到了JSTL中的<if test=””></if>这样的标签，那么，能不能将这类的标签引入到SQL语句中呢？假设可以，那么我们这里就需要一个专门的SQL解析器来解析这样的SQL语句，但是，if判断的变量来自于哪里呢？传入的值本身是可变的，那么我们得为这个值定义一个不变的变量名称，而且这个变量名称必须和对应的值要有对应关系，可以通过这个变量名称找到对应的值，这个时候我们想到了key-value的Map。解析的时候根据变量名的具体值来判断。

假如前面可以判断没有问题，那么假如判断的结果是true，那么就需要输出的标签里面的SQL片段，但是怎么解决在标签里面使用变量名称的问题呢？这里我们需要使用一种有别于SQL的语法来嵌入变量（比如使用＃变量名＃）。这样，SQL语句经过解析后就可以动态的生成符合上下文的SQL语句。

还有，怎么区分开占位符变量和非占位变量？有时候我们单单使用占位符是满足不了的，占位符只能为查询条件占位，SQL语句其他地方使用不了。这里我们可以使用#变量名#表示占位符变量，使用$变量名$表示非占位符变量。

**第四步优化：结果映射和结果缓存**

**问题描述：**

执行SQL语句、获取执行结果、对执行结果进行转换处理、释放相关资源是一整套下来的。假如是执行查询语句，那么执行SQL语句后，返回的是一个ResultSet结果集，这个时候我们就需要将ResultSet对象的数据取出来，不然等到释放资源时就取不到这些结果信息了。我们从前面的优化来看，以及将获取连接、设置传入参数、执行SQL语句、释放资源这些都封装起来了，只剩下结果处理这块还没有进行封装，如果能封装起来，每个数据库操作都不用自己写那么一大堆Java代码，直接调用一个封装的方法就可以搞定了。

**解决问题：**

我们分析一下，一般对执行结果的有哪些处理，有可能将结果不做任何处理就直接返回，也有可能将结果转换成一个JavaBean对象返回、一个Map返回、一个List返回等等，结果处理可能是多种多样的。从这里看，我们必须告诉SQL处理器两点：第一，需要返回什么类型的对象；第二，需要返回的对象的数据结构怎么跟执行的结果映射，这样才能将具体的值copy到对应的数据结构上。

    接下来，我们可以进而考虑对SQL执行结果的缓存来提升性能。缓存数据都是key-value的格式，那么这个key怎么来呢？怎么保证唯一呢？即使同一条SQL语句几次访问的过程中由于传入参数的不同，得到的执行SQL语句也是不同的。那么缓存起来的时候是多对。但是SQL语句和传入参数两部分合起来可以作为数据缓存的key值。

**第五步优化：解决重复SQL语句问题**

**问题描述：**

由于我们将所有SQL语句都放到配置文件中，这个时候会遇到一个SQL重复的问题，几个功能的SQL语句其实都差不多，有些可能是SELECT后面那段不同、有些可能是WHERE语句不同。有时候表结构改了，那么我们就需要改多个地方，不利于维护。

**解决问题：**

     当我们的代码程序出现重复代码时怎么办？将重复的代码抽离出来成为独立的一个类，然后在各个需要使用的地方进行引用。对于SQL重复的问题，我们也可以采用这种方式，通过将SQL片段模块化，将重复的SQL片段独立成一个SQL块，然后在各个SQL语句引用重复的SQL块，这样需要修改时只需要修改一处即可。

### 原理分析之二：框架整体设计

## 1.引言

本文主要讲解Mybatis的整体程序设计，理清楚框架的主要脉络。后面文章我们再详细讲解各个组件。

## 2.整体设计

### 2.1 总体流程

**(1)加载配置并初始化**

**触发条件：**加载配置文件

      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，将SQL的配置信息加载成为一个个MappedStatement对象（包括了传入参数映射配置、执行的SQL语句、结果映射配置），存储在内存中。

**(2)接收调用请求**

**触发条件：**调用Mybatis提供的API

**传入参数**：为SQL的ID和传入参数对象

**处理过程：**将请求传递给下层的请求处理层进行处理。

**(3)处理操作请求**

**触发条件：**API接口层传递请求过来

**传入参数**：为SQL的ID和传入参数对象

**处理过程：**

        (A)根据SQL的ID查找对应的MappedStatement对象。

        (B)根据传入参数对象解析MappedStatement对象，得到最终要执行的SQL和执行传入参数。

        (C)获取数据库连接，根据得到的最终SQL语句和执行传入参数到数据库执行，并得到执行结果。

        (D)根据MappedStatement对象中的结果映射配置对得到的执行结果进行转换处理，并得到最终的处理结果。

        (E)释放连接资源。

**(4)返回处理结果**

将最终的处理结果返回。

### 2.2 功能架构设计http://dl.iteye.com/upload/attachment/0065/2605/6a59d52b-db76-3737-b164-f366bb2fbce7.png

**功能架构讲解：**

我们把Mybatis的功能架构分为三层：

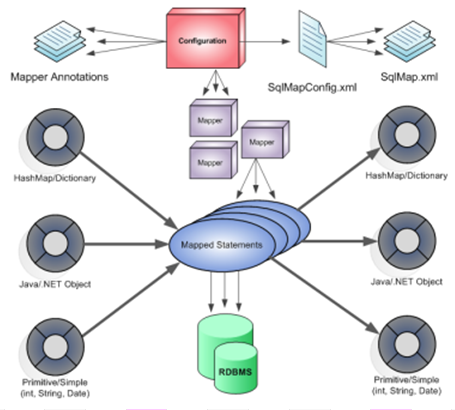
**(1)API接口层：**提供给外部使用的接口API，开发人员通过这些本地API来操纵数据库。接口层一接收到调用请求就会调用数据处理层来完成具体的数据处理。

**(2)数据处理层：**负责具体的SQL查找、SQL解析、SQL执行和执行结果映射处理等。它主要的目的是根据调用的请求完成一次数据库操作。

**(3)基础支撑层：**负责最基础的功能支撑，包括连接管理、事务管理、配置加载和缓存处理，这些都是共用的东西，将他们抽取出来作为最基础的组件。为上层的数据处理层提供最基础的支撑。

### 2.3 框架架构设计

### 



**框架架构讲解：**

(1)**加载配置：**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，将SQL的配置信息加载成为一个个MappedStatement对象（包括了传入参数映射配置、执行的SQL语句、结果映射配置），存储在内存中。

(2)**SQL解析：**当API接口层接收到调用请求时，会接收到传入SQL的ID和传入对象（可以是Map、JavaBean或者基本数据类型），Mybatis会根据SQL的ID找到对应的MappedStatement，然后根据传入参数对象对MappedStatement进行解析，解析后可以得到最终要执行的SQL语句和参数。

(3) **SQL执行：**将最终得到的SQL和参数拿到数据库进行执行，得到操作数据库的结果。

(4)**结果映射：**将操作数据库的结果按照映射的配置进行转换，可以转换成HashMap、JavaBean或者基本数据类型，并将最终结果返回。

**这里总结一下上面的过程，总共由三个步骤：**

**步骤一：**读取Ibatis的主配置文件，并将文件读成文件流形式(InputStream)。

**步骤二：**从主配置文件流中读取文件的各个节点信息并存放到Configuration对象中。读取mappers节点的引用文件，并将这些文件的各个节点信息存放到Configuration对象。

**步骤三：**根据Configuration对象的信息获取数据库连接，并设置连接的事务隔离级别等信息，将经过包装数据库连接对象SqlSession接口返回，DefaultSqlSession是SqlSession的实现类，所以这里返回的是DefaultSqlSession，SqlSession接口里面就是对外提供的各种数据库操作。

比较好的文档：

<https://www.jianshu.com/p/8d1f499063a1>

<http://c.biancheng.net/view/4304.html>