# ORM设计

## 设计原理

## 功能架构设计



**功能架构讲解：**

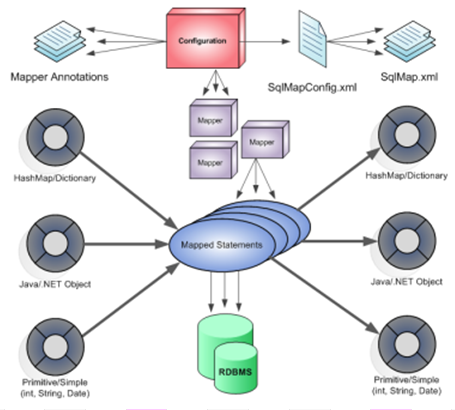
我们把Mybatis的功能架构分为三层：

**(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。

## 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。

## 性能指标

[Mybatis](http://code.google.com/p/mybatis/)比之于Spring JDBC的三大优势：

* 一可以联合查询多个表然后把结果转换为复杂对象，如Order->OrderItem->Product。 見[官方中文文档result map](http://www.mybatis.org/core/zh/sqlmap-xml.html#Result_Maps)
* 二是复杂SQL可以定义于XML文件，可自由换行缩进，且支持一些语法，简化SQL的拼装, 見 [官方中文文档dynamic sql](http://www.mybatis.org/core/zh/dynamic-sql.html)
* 三還是在XML裡，可以根據不同的數據庫，定義不同語法的SQL，見 [官方中文文档dynamic sql](http://www.mybatis.org/core/zh/dynamic-sql.html)的末尾

## 接口设计

值得留意是Mybatis+Spring的结合，最新的官方默认做法已经很潮了，不再需要Config.xml， 也不用在编写Dao实现(只需要DAO接口)。

对默认做法还做了两个合理化的改进，第一是DAO们不再叫Mapper而是直接叫回DAO，第二是Mapper.xml们不放在和Entity一样深的目录，放回一个浅的目录。

## 接口说明

整个方案是这样的：

1. 在applicationContext.xml中，SqlSessionFactoryBean先指明type Aliases的扫描目录，将所有entity目录配进去，就省掉了用Config.xml的aliase配置。
2. 在applicationContext.xml中，SqlSessionFactoryBean再指明mapper xml文件的目录，这样就不用将mapper文件放到与entity一样深的目录了。
3. 在applicationContext.xml中，MapperScannerConfigurer指明需要扫描的DAO接口的目录，动态创建其实现类。  
   如果DAO接口在多个目录中，简单的做法是创建一个MyBatisDAO的annotation，所有DAO都用这个annotation标识，然后用annotations属性指定它，那么package配到项目顶层package就可以了。
4. DAO接口，方法名必须与Mapper.xml中sql的名称一样。

Mapper.xml文件，nameSpace属性，必须是DAO接口的完整Class名。