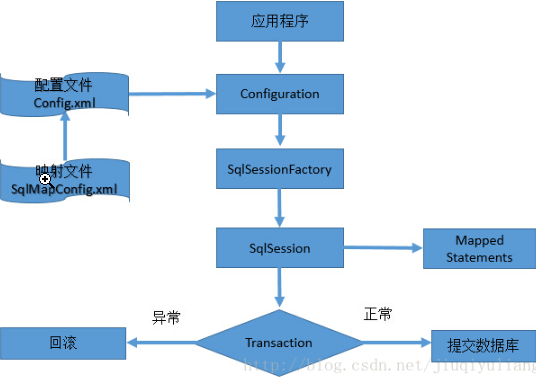
**MyBatis是个啥？**

MyBatis是一个持久层框架，支持定制化sql，存储过程以及高级映射，MyBatis几乎避免了所有的JDBC代码和手动设置参数，以及获取结果集，MyBatis可以是用简单的XML或注解来配置和映射原生信息，将接口和java的POJOs（plain old java objects普通的java对象），映射成数据库中的记录

JDBC（java database conntivity ，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java语言编写的类和接口组成，JDBC提供了一种基准，据此可以构建更高级的工具和接口，是数据开发人员能够编写数据库应用程序，同时JDBC也是一个商标名。

API（application programming interface，应用程序编程接口）是一些预定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。



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。

MyBatis 是支持定制化 SQL、存储过程以及高级映射的优秀的持久层框架，其主要就完成2件事情：

1. 封装JDBC操作
2. 利用反射打通Java类与SQL语句之间的相互转换

MyBatis的主要设计目的就是让我们对执行SQL语句时对输入输出的数据管理更加方便，所以方便地写出SQL和方便地获取SQL的执行结果才是MyBatis的核心竞争力

**sqlMapConfig.xml**

*<?***xml version="1.0" encoding="UTF-8"** *?>***<!DOCTYPE configuration  
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd"*>***<**configuration**>

<settings>

<setting name="cacheEnabled" value="true"/>开启二级缓存

<setting name="lazyLoadingEnabled" value="false"/>懒加载

<!--<setting name="logImpl" value="STDOUT\_LOGGING"/> &lt;!&ndash; 打印日志信息 &ndash;&gt;-->

</settings>

<**typeAliases**>给类起一个别名  
 *<!--<typeAlias type="com.lshs.entry.User" alias="user" />-->  
 <!--<typeAlias type="com.lshs.mapper.UserMapper" alias="UserMapper" />-->* <**package name="com.lshs.entry"** />  
 *<!--<package name="com.lshs.mapper" />-->* </**typeAliases**>  
 <**environments default="mybatis"**>配置数据源  
 <**environment id="mybatis"**>  
 <**transactionManager type="JDBC"**/>配置事务  
 <**dataSource type="POOLED"**>  
 <**property name="driver" value="com.mysql.jdbc.Driver"**/>  
 <**property name="url" value="jdbc:mysql://localhost:3306/mybatis?useCode=true&amp;character=utf-8"**/>  
 <**property name="username" value="root"**/>  
 <**property name="password" value="ls0611"**/>  
 </**dataSource**>  
 </**environment**>  
 </**environments**>  
 <**mappers**>  
 *<!--<mapper resource="com/lshs/mybatis/User.xml"/>-->  
 <!--<mapper class="com.lshs.mapper.UserMapper" />-->* <**package name="com.lshs.mapper"** /> 加载mapping.xml文件  
 </**mappers**>  
</**configuration**>

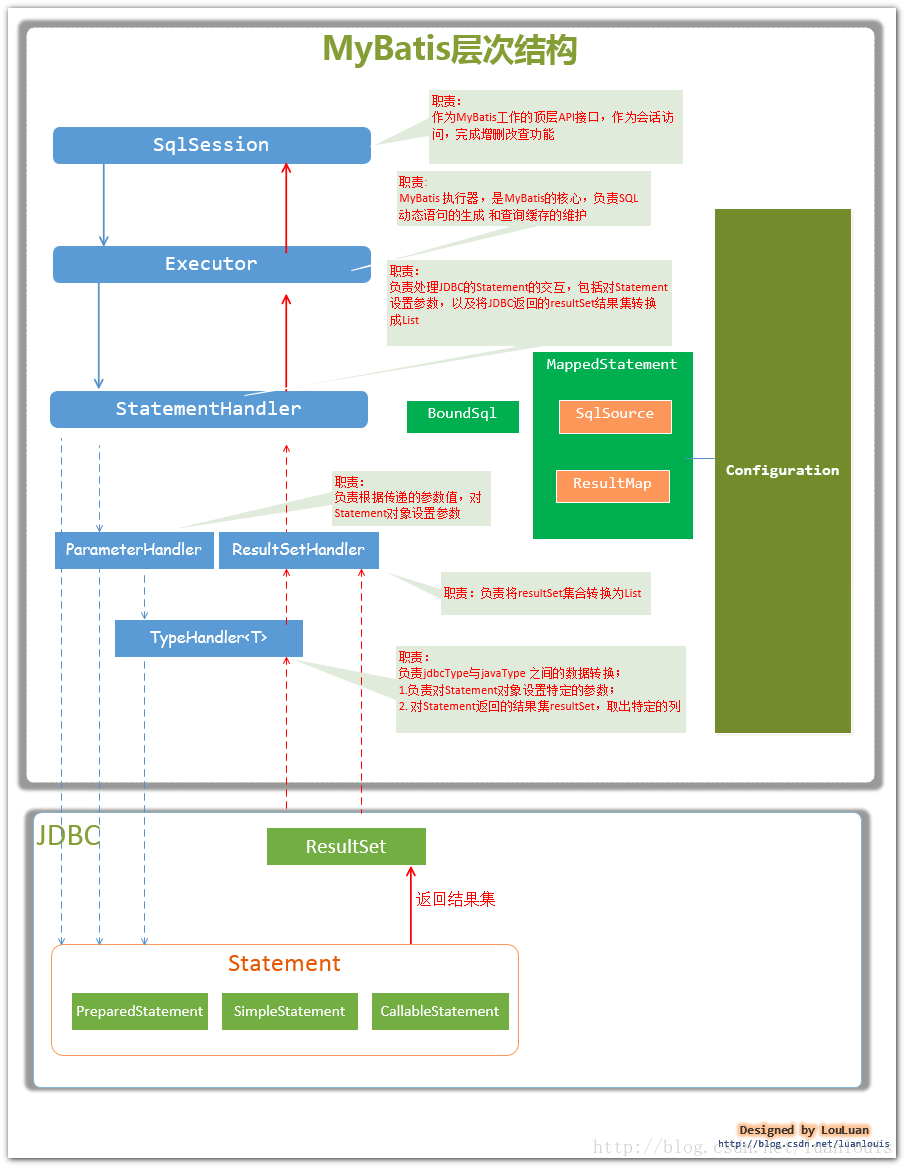
**XXXMapper.xml**

*<?***xml version="1.0" encoding="UTF-8"** *?>***<!DOCTYPE mapper  
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>***<**mapper namespace="com.lshs.mapper.UserMapper"**>这个必须要有  
  
 <**select id="findAll" resultType="User"**>封装sql语句  
 select *\** from user  
 </**select**>  
  
 <**select id="findUserById" resultType="User" parameterType="Integer"**>  
 select *\** from user where id = #{id}  
 </**select**>  
  
 <**select id="findUserByName" parameterType="String" resultType="User"**>  
 select *\** from user where username= #{username}  
 </**select**>  
  
 <**select id="findUserByLikeName" parameterType="String" resultType="com.lshs.entry.User"**>  
 /\*select \* from user where username like '%${value}%'\*/  
 select *\** from user where username like "%"#{username}"%"  
 </**select**>  
*<!--获得id的办法-->* <**insert id="save" parameterType="com.lshs.entry.User"**>  
 <**selectKey keyColumn="id" keyProperty="id" resultType="int" order="AFTER"**>  
 select last\_insert\_id()  
 </**selectKey**>  
 insert into user (username,birthday,sex,address) values(#{username},#{birthday},#{sex},#{address})  
 </**insert**>  
 *<!--不获得id的-->* <**insert id="save2" parameterType="com.lshs.entry.User"**>  
 insert into user values(null,#{username},#{birthday},#{sex},#{address})  
 </**insert**>  
  
 <**delete id="deleteById" parameterType="int"**>  
 delete from user where id = #{id}  
 </**delete**>  
  
 <**update id="update" parameterType="com.lshs.entry.User"**>  
 update user set username=#{username} where id=#{id}  
 </**update**>  
</**mapper**>

**MyBatis的主要成员**

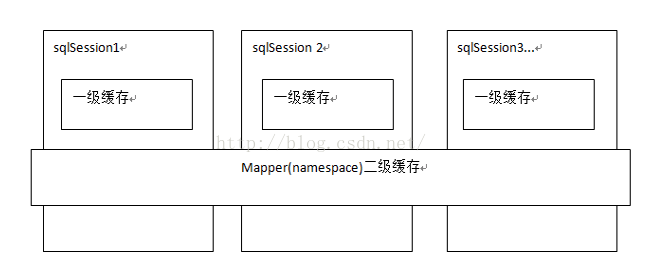
* Configuration        MyBatis所有的配置信息都保存在Configuration对象之中，配置文件中的大部分配置都会存储到该类中
* SqlSession            作为MyBatis工作的主要顶层API，表示和数据库交互时的会话，完成必要数据库增删改查功能
* Executor               MyBatis执行器，是MyBatis 调度的核心，负责SQL语句的生成和查询缓存的维护
* StatementHandler 封装了JDBC Statement操作，负责对JDBC statement 的操作，如设置参数等
* ParameterHandler  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。



**MyBatis缓存**

MyBatis提供查询缓存，用于减轻数据库压力，提高性能。MyBatis提供了一级缓存和二级缓存。



一级缓存是SqlSession级别的缓存，每个SqlSession对象都有一个哈希表用于缓存数据，不同SqlSession对象之间缓存不共享。同一个SqlSession对象对象执行2遍相同的SQL查询，在第一次查询执行完毕后将结果缓存起来，这样第二遍查询就不用向数据库查询了，直接返回缓存结果即可。MyBatis默认是开启一级缓存的。

二级缓存是mapper级别的缓存，二级缓存是跨SqlSession的，多个SqlSession对象可以共享同一个二级缓存。不同的SqlSession对象执行两次相同的SQL语句，第一次会将查询结果进行缓存，第二次查询直接返回二级缓存中的结果即可。MyBatis默认是不开启二级缓存的，可以在配置文件中使用如下配置来开启二级缓存：

<settings>

<setting name="cacheEnabled" value="true"/>

</settings>

当SQL语句进行更新操作(删除/添加/更新)时，会清空对应的缓存，保证缓存中存储的都是最新的数据。MyBatis的二级缓存对细粒度的数据级别的缓存实现不友好，比如如下需求：对商品信息进行缓存，由于商品信息查询访问量大，但是要求用户每次都能查询最新的商品信息，此时如果使用mybatis的二级缓存就无法实现当一个商品变化时只刷新该商品的缓存信息而不刷新其它商品的信息，因为mybaits的二级缓存区域以mapper为单位划分，当一个商品信息变化会将所有商品信息的缓存数据全部清空。解决此类问题需要在业务层根据需求对数据有针对性缓存，具体业务具体实现。