**主要功能：**

* typehandlers类型处理器（日期相互转换）
* objectfactory在对象实例化时进行一些操作
* **动态sql语句可以引入sql片段**
* **高级映射(1v1、1vn、nvn)**
* **缓存结构**

MyBatis创建结果对象的新实例时，会使用一个对象工厂（ObjectFactory）实例来完成

默认的对象工厂需要做的仅仅是实例化目标类

要么通过默认构造方法，要么在参数映射存在的时候通过参数构造方法来实例化

**resultType**

只有查询出来的列名和pojo中的属性名一致，该列才可以映射成功。

* 查询出来的列名和pojo中的属性名全部不一致，没有创建pojo对象。
* 查询出来的列名和pojo中的属性至少有一个一致，就会创建pojo对象。

**resultMap**

如果查询出来的列名和pojo的属性名不一致，定义一个resultMap对列名和pojo属性名之间作一个映射关系

属性：

* Association：包装类中的一个属性是其他包装类对象
* Collection： 包含list等集合属性
* Discriminator：根据某个字段的值决定使用哪个结果集

MyBatis的每一个查询映射的返回类型都是ResultMap，只是当提供的返回类型属性是resultType的时候，MyBatis对自动的给把对应的值赋给resultType所指定对象的属性

**${}和#{}**

#将传入的数据都当成一个字符串，会对自动传入的数据加一个双引号

如：order by #user\_id#，如果传入的值是111,那么解析成sql时的值为order by "111", 如果传入的值是id，则解析成的sql为order by "id"

能够很大程度防止sql注入

$将传入的数据直接显示生成在sql中

如：order by $user\_id$，如果传入的值是id，则解析成的sql为order by id.

Sql注入：改变sql语句，执行不期望操作

<https://blog.csdn.net/weixin_41607190/article/details/82818804>

**缓存结构**

**作用：用于减轻数据压力，提高数据库性能**

一级缓存**是SqlSession级别的缓存**

**在操作数据库时需要构造 sqlSession对象，在对象中有一个HashMap用于存储缓存数据。不同的sqlSession之间的缓存数据区域（HashMap）是互相不影响的**

**前提是SqlSession没关闭**

实际开发中，MyBatis通常和Spring进行整合开发。Spring将事务放到Service中管理，每一个service中的sqlsession是不同的，这是通过mybatis-spring中的MapperScannerConfigurer创建sqlsession自动注入到service中的。 每次查询之后都要进行关闭sqlSession，关闭之后数据被清空。

spring整合mybatis之后，由spring管理的sqlSeesion在sql方法(增删改查等操作)执行完毕后就自行关闭了sqlSession，不需要我们对其进行手动关闭

所以spring整合之后，如果没有事务，一级缓存是没有意义的

如果开启二级缓存，关闭sqlsession后，会把该sqlsession一级缓存中的数据添加到namespace的二级缓存中。这样，缓存在sqlsession关闭之后依然存在

Mybatis默认开启一级缓存

sql语句且向sql中传递的参数也相同

二级缓存**是mapper级别的缓存**

* **多个SqlSession去操作同一个Mapper的sql语句，多个SqlSession去操作数据库得到数据会存在二级缓存区域，多个SqlSession可以共用二级缓存，二级缓存是跨SqlSession的**
* **其作用域是mapper的同一个namespace, 每一个namespace的mapper都有一个二级缓存区域**

**不同的sqlSession两次执行相同namespace下的**sql语句且向sql中传递参数也相同**即最终执行相同的sql语句，第一次执行完毕会将数据库中查询的数据写到缓存（内存），第二次会从缓存中获取数据将不再从数据库查询，从而提高查询效率**

mybatis的缓存是基于[namespace:sql语句:参数]来进行缓存的，意思就是，SqlSession的HashMap存储缓存数据时，是使用[namespace:sql:参数]作为key，查询返回的语句作为value保存的。

例如：1242243203:1146242777:winclpt.bean.userMapper.getUser:0:2147483647:select \* from user where id=?:19

当发起请求查询数据库的时候会先产看二级缓存是否开启，如果开启会从二级缓存中查看，如果没有开启会先从mybatis一级中的缓存（HashMap）中去查询，如果不存在那么将会执行sql语句

**特例：**

**sqlSession去执行commit操作（执行插入、更新、删除），清空当前SqlSession/Mapper中的一/二级缓存，这样做的目的为了让缓存中存储的是最新的信息，避免脏读**

**重要组件**

SqlSessionFactoryBuilder**则可以从XML配置文件或一个Configuration（配置）的实例构建出SqlSessionFactory的实例.（单例线程安全）（一创建完工厂即可回收）**

SqlSessionFactory **作用是创建SqlSession，每次应用程序需要访问数据库，我们通过它创建一个SqlSession，SqlSessionFactory在整Mybatis整个生命周期中（每个数据库对应一个SqlSessionFactory）是单例产生的；**

SqlSession**相当于JDBC的一个Connection对象，它的底层封装了JDBC连接,可以用SqlSession实例来直接执行被映射的SQL语句**

**实例不能被共享,同时SqlSession也是线程不安全的**

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**

**JDBC（java数据库连接）**

**缺点:**

* 数据库配置信息（密码…）、Sql语句、传入参数硬编码，变化时要修改源代码然后重新编译

将Sql语句配置在XXXXmapper.xml文件中与java代码分离

* 遍历结果集（getString(“表字段名“)）字段名改变

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

* 数据库的频繁连接和断开db.close();系统资源浪费;

解决：在SqlMapConfig.xml中配置数据连接池，使用连接池管理数据库连接

**Cache接口方法**

* String getId()：mybatis缓存操作对象的标识符。一个mapper对应一个mybatis的缓存操作对象。
* void putObject(Object key, Object value)：将查询结果塞入缓存。
* Object getObject(Object key)：从缓存中获取被缓存的查询结果。
* Object removeObject(Object key)：从缓存中删除对应的key、value。只有在回滚时触发。一般我们也可以不用实现
* void clear()：发生更新时，清除缓存。
* int getSize()：可选实现。返回缓存的数量。
* ReadWriteLock getReadWriteLock()：可选实现。用于实现原子性的缓存操作。