Hibernate是全自动全映射ORM（Object Relation Mapping）框架，编程人员无法使用SQL语句进行优化只能通过HQL语句进行优化，Hibernate使用方便

MyBatis可以说是半自动化轻量级的ORM，相对于Hibernate是将SQL语句的编写任务交给了编程人员方便了语句的优化

MyBatis的全局配置文件配置数据库连接池和事务管理器等，一般名称为mybatis-config.xml，没有全局配置文件也可以通过实例化的方式进行数据库连接，配置文件的一般格式（配置文件中的标签有严格顺序的）：

<?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>

<properties url="资源文件全路径名称（引用网络路径或者磁盘路径下的资源）" resource="资源文件全路径名称（引用类路径下的资源）"></properties>

<settings>

<setting name="mapUnderscoreToCamelCase" value="true"/>（是否设置开启自动驼峰命名规则，即MyBatis自动识别符合驼峰规范的数据库列名与相似的实体类属性名）

<setting name="jdbcTypeForNull" value="NULL"/>（设置传递SQL语句是属性值为空时的传递类型，默认OTHER）

<setting name="lazyLoadingEnabled" value="true"/>（设置使用延时加载，与映射文件SQL语句分步查询结合使用）

<setting name="aggressiveLazyLoading" value="false"/>（设置侵入延时加载，即多个关联属性使用其中一个是否延时加载其他关联的属性）

<setting name="cacheEnabled" value="true"/>（设置是否使用全局二级缓存，设置后需要在SQL映射文件中配置缓存）

<setting name="localCacheScope" value="session/statement"/>（设置本地缓存即一级缓存作用域，statement禁用一级缓存）

</settings>

<typeAliases>

<typeAlias type="指定需要设置别名的全路径类名" alias="别名（不写默认是类名小写，别名不严格区分大小写）"/>设置java类型的别名

<package name="指定需要设置别名的全路径包名"/>（以包的方式进行批量设置类别名，可以在指定类前使用注解@Alias("别名")进行设置别名，MyBatis为常用基本类型名的别名一般为前方带\_或者类名小写，在自定义设置类别名时不能与MyBatis设置的重复）

</typeAliases>

<environments default="环境标识id（MyBatis可以配置多个环境，指定默认使用的环境）">

[<environment id="环境唯一标识">

<transactionManager type="事务管理器的类型（JDBC（JdbcTransactionFactory）|MANAGED（ManagedTransactionFactory））" />（可以自定义事务管理器，需要实现TransactionFactory接口）

<dataSource type="数据源类型（UNPOOLED（无连接池UnpooledDataSourceFactory）|POOLED（连接池PooledDataSourceFactory）|JNDI（JndiDataSourceFactory））">可以自定义数据源类型，需要实现DataSourceFactory接口）

<property name="driver" value="数据库驱动名称（可以使用${资源的属性名}取出引用资源的属性名对应的属性值）" />

<property name="url" value="连接数据库的url地址" />

<property name="username" value="用户名" />

<property name="password" value="密码" />

</dataSource>

</environment>]

</environments>

<databaseIdProvider type="DB\_VENDOR（VendorDatabaseProvider，使用getDatabaseProductName()得到数据库厂商的标识，）">（支持多数据库SQL语句）

[<property name="MySQL/ Oracle / SQL Server " value="mysql/ oracle / sqlserver "/>]（为数据库厂商标识设置别名）

</databaseIdProvider>

<mappers>（将SQL映射文件注册到MyBatis全局配置文件中）

[<mapper url="资源文件全路径名称（引用网络路径或者磁盘路径下的资源）" resource="资源文件全路径名称（引用类路径下的资源）" class="引用映射文件对应的注册接口，有映射文件需要名称与接口名一致并且在同一目录下，没有映射文件可以在接口方法名使用注解方式（@Select("SQL语句")一般重要复杂的接口写SQL配置文件，不重要简单的为了开发方便可以使用注解）" />]

<package name="SQL映射文件所在的包名"/>（以包的方式进行批量注册映射文件，注解方式的接口可以正常使用，映射文件的方式需要保证映射文件与对应接口在同一目录下）

</mappers>

</configuration>

MyBatis的SQL语句的映射配置文件一般格式：

<?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="名称空间（用于区分不同SQL语句，一般是对应接口的全路径类名）">

<cache-ref namespace="指定名称空间的接口全类名"/>（引用指定名称空间相同的缓存）

<cache eviction="缓存回收策略（LRU最近最少使用的：移除最长时间不被使用的对象/FIFO先进先出：移除最先进入缓存的对象/SOFT软引用：移除基于垃圾回收器状态和软引用规则的对象/WEAK弱引用：更积极移除基于垃圾回收器状态和软引用规则的对象，默认LRU）" flushInterval="缓存刷新间隔毫秒值（缓存指定时间清空一次，默认不清空）" readOnly="true/false缓存的数据只允许读取，MyBatis会将数据在缓存的引用交给用户，速度快不安全/获取的数据允许被修改，MyBatis李勇序列化和反序列化复制一份数据给用户，安全速度慢，默认false" size="缓存存放多少元素" type="指定自定义的缓存的全类名"></cache>（SQL配置使用二级缓存）

<resultMap type="JavaBean类型全包名" id="自定义结果映射唯一标识">（自定义JavaBean映射的数据库表规则，与select标签中的resultMap属性对应使用，不设置的属性或列映射默认自动根据名称封装）

<id column="数据库结果列名" property="JavaBean属性名"/>（设置主键映射）

<result column="数据库结果列名" property=" JavaBean属性名"/>（设置属性映射）

<result column="数据库结果列名" property=" JavaBean对象属性名.属性名"/>（联合查询设置JavaBean中对象属性的结果集级联属性封装）

<association property="JavaBean的对象属性名" javaType="JavaBean的对象属性的类型全包名" select="映射文件SQL语句的唯一标识全包名id" column="数据库结果列名（指定以数据库列名为select标签查询语句的参数进行查询，多个参数需要封装成Map传递{需要参数名=结果集列名[,需要参数名=结果集列名]}）" fetchType="lazy/eager（延时/立即加载，设置指定的SQL语句是否延时加载）">（指定映射的JavaBean的对象属性，javaType/ select指定JavaBean的对象属性对应的结果集是根据SQL关联查询/分步查询）

<id column="数据库结果列名" property="JavaBean对象属性的属性名"/>（设置主键映射）

<result column="数据库结果列名" property=" JavaBean对象属性的属性名"/>（设置属性映射）

</association>

<collection property=" JavaBean的集合对象属性名" ofType="指定集合里的元素的类型全包名" select="映射文件SQL语句的唯一标识全包名id" column="数据库结果列名（指定以数据库列名为select标签查询语句的参数进行查询），多个参数需要封装成Map传递{需要参数名=结果集列名[,需要参数名=结果集列名]}）" fetchType="lazy/eager（延时/立即加载）">（定义关联集合类型的属性封装规则）

<id column="数据库结果列名" property="JavaBean集合对象属性的属性名"/>（设置主键映射）

<result column="数据库结果列名" property=" JavaBean集合对象属性的属性名"/>（设置属性映射）

</collection>

<discriminator javaType="列名对应的Java类名" column="指定结果中判断的列名">（鉴别器，MyBatis根据指定列名的值进行指定的结果集封装）

<case value="列名可能的值" resultType/resultMap="指定封装的结果类型">

<id/><result/>

<association property=""></association>

<collection property=""></collection>

</case>

</discriminator>

</resultMap>

<sql id="SQL语句片段名">SQL语句字符串</sql>（一般设置常用SQL语句片段）

<select id="SQL语句的唯一标识（一般为对应接口的方法名）" resultType="SQL语句执行的结果返回类型（返回多条数据即集合时返回类型是集合中元素的类型，在对应接口方法前使用注解@MapKey("返回类型的属性名")指定返回为以属性名的值为键值key的值的Map集合的类型，默认返回List集合，返回一条数据封装成集合时返回类型为集合类名小写，）" databaseId="指定SQL语句执行的数据库厂商别名" resultMap="自定义结果映射唯一标识" useCache="是否使用二级缓存，默认true" flushCache="是否清除一级二级缓存（默认false，改变数据库数据的操作默认true）">

<include refid=" SQL语句片段名">（引用指定的SQL语句片段）

<property name="属性名" value="属性值"/>（引用SQL语句片段附带自定义属性参数，取值只能使用${}）

</include>

<bind name="变量名" value="OGNL字符串拼接的表达式"/>（将OGNL表达式绑定到指定变量名中）

select \* from tbl\_employee where id= #{属性名称}（#{属性名称}从传递的参数中取出对应的属性名称的值）

<where>（where标签会在拼接后的整体字符串前where，并且只能自动去除拼接字符中前部多余的and/or，不使用where标签也可以在SQL语句where后加1=1其他拼接字符串前都加and）

[< if test="判断表达式（格式为OGNL表达式，特殊符号需要写转义字符）">（动态判断拼装SQL语句）

列名= #{属性名}/（and/or列名= #{属性名}）

</if>]

<choose>

[<when test="判断表达式">（依次从上向下判断，判断表达式符合要求拼接指定字符串，只会拼接其中一个）

拼接的SQL语句字符串

</when>]

<otherwise>

拼接的SQL语句字符串

</otherwise>

</choose>

</where>

<trim prefix="拼接后整体字符串前加指定字符串" prefixOverrides="去掉整体字符串前部多余的指定字符串" suffix="拼接后整体字符串后加指定字符串" suffixOverrides="去掉整体字符串后部多余的指定字符串">（trim标签能实现where标签和set标签的功能）

[< if test="判断表达式（格式为OGNL表达式，特殊符号需要写转义字符）">（动态判断拼装SQL语句）

列名= #{属性名}（拼接的SQL语句字符串）

</if>]

</trim>

<foreach collection="指定需要遍历的集合元素" item="遍历的当前元素变量" index="索引/键值key（遍历集合是List/Map）" separator="每个元素之间的分隔符" open="遍历所有结果后前部加指定字符串（一般是where 列名 in(）" close="遍历所有结果后前部加指定字符串（一般是））">

拼接的SQL语句字符串

</foreach>

<if test="\_databaseId==''数据库厂商标识设置的别名">（MyBatis映射文件SQL语句的内置参数\_databaseId当前使用的数据库厂商标识设置的别名）

<if test="\_parameter!=null">（MyBatis映射文件SQL语句的内置参数\_ parameter对应接口方法的传递的参数对象）

拼接的SQL语句字符串

</if>

</if>

</select>

<insert id=" SQL语句的唯一标识" parameterType="参数类型（可省略）" useGeneratedKeys="true（指定使用主键自增获取策略）" keyProperty="实体类的属性名即JavaBean的属性名（指定主键属性，获取的主键自增值会返回到指定属性名中）" databaseId="指定SQL语句执行的数据库厂商别名">（MySQL支持主键自增以及自增的主键值获取，MyBatis一样使用statement.getGeneratedKeys()获取自增主键的值）

insert into 表名(列名[,列名])values(#{属性名},[#{属性名}])

</insert>

<insert id=" SQL语句的唯一标识" parameterType="参数类型" databaseId="指定SQL语句执行的数据库厂商别名">

<selectKey keyProperty="查询出的主键值封装到指定JavaBean类的属性名" order=" selectKey的SQL语句执行在insert的SQL语句之前/后（BEFORE/AFTER）" resultType="查询的结果的返回类型">

select 序列名.nextval/序列名.currval from dual

</selectKey>（数据库的主键使用序列策略自增的插入语句，例如Oracle需要先查询序列值然后插入数据/先插入数据然后查询当前序列值）

insert into 表名(列名[,列名])values(#{属性名}/序列名.nextval,[#{属性名}])

<foreach collection="指定需要遍历的集合元素（需要在对应接口方法参数名前使用注解@Param("集合别名")指定别名）" item="遍历的当前元素变量名" separator=",">（批量插入多条数据）

(#{变量名.属性名}[,#{变量名.属性名}])

</foreach>

<foreach collection="指定需要遍历的集合元素（需要在对应接口方法参数名前使用注解@Param("集合别名")指定别名）" item="遍历的当前元素变量名" separator=";">（使用多条插入SQL语句批量插入多条数据，需要在连接数据库的url后设置属性allowMultiQueries=true）

insert into 表名(列名[,列名])values(#{变量名.属性名}[,#{变量名.属性名}])

</foreach>

<foreach collection="指定需要遍历的集合元素" item="遍历的当前元素变量" open="begin" close="end;">（Oracle数据库不支持values后有多条数据，Oracle可以通过begin与and之间有多条insert语句方式插入多条数据）

insert into 表名(列名[,列名])values(#{变量名.属性名}[,#{变量名.属性名}]);

</foreach>

</insert>

<insert id=" SQL语句的唯一标识" parameterType="参数类型" databaseId="指定SQL语句执行的数据库厂商别名">

insert into 表名(列名[,列名])

<foreach collection="指定需要遍历的集合元素（需要在对应接口方法参数名前使用注解@Param("集合别名")指定别名）" item="遍历的当前元素变量名" separator="union" open=" select列名[,列名] from(" close=")">（Oracle使用中间表方式批量插入多条数据）

select#{变量名.属性名}[,#{变量名.属性名}] from dual

</foreach>

</insert>

<update id=" SQL语句的唯一标识" parameterType="参数类型" databaseId="指定SQL语句执行的数据库厂商别名">

update 表名set 列名=#{属性名}[,列名=#{属性名}] where列名=#{属性名}

<set>（set标签会自动将拼接后的字符串后部多余的,去掉）

<if test="判断表达式">

拼接的SQL语句字符串

</if>

</set>

</update>

<delete id=" SQL语句的唯一标识" parameterType="参数类型" databaseId="指定SQL语句执行的数据库厂商别名">

delete from 表名 where列名=#{属性名}

</delete>

</mapper>

MyBatis的SQL映射文件需要注册到MyBatis的全局配置文件中

InputStream inputStream = Resources.getResourceAsStream("MyBatis全局配置文件全路径名称（一般为mybatis-config.xml）");

SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);获取SQL语句执行会话工厂的实例

SqlSession 会话名称=sqlSessionFactory.openSession();获取不自动提交的SQL语句执行的会话

会话名称.selectOne("SQL语句映射文件的唯一标识id（一般为了区分唯一的SQL语句，id前加上名称空间名）", SQL语句需要的参数);使用会话名称执行指定的SQL语句映射文件中的SQL语句并返回结果

会话名称.commit();SQL会话进行增加删除修改等操作需要提交

会话名称.close();关闭SQL语句执行的会话

MyBatis执行SQL语句的常用方法是使用接口式编程，用接口的方式为SQL语句描述传递的参数，通过将SQL映射文件的名称空间指定为接口的全路径类名来将接口类与相应的SQL映射文件进行绑定，通过将SQL映射文件中的SQL语句的唯一标识id指定为接口类中的方法名来进行SQL语句绑定，接口与SQL映射文件动态绑定后，MyBatis会为接口创建一个动态代理对象，代理对象进行对应SQL语句的执行

MyBatis映射文件对应的接口方法允许的返回类型（Integer/Long/Boolean/void）有返回值会返回操作成功的数量值

MyBatis映射文件对应的接口方法中的参数传递规则：

单个参数，MyBatis不做任何处理，接口方法对应的映射文件SQL语句中#{任意参数名}

单个参数特殊处理，单个参数是Collection（List、Set）类型或者数组Array，传入后MyBatis会封装成Map，接口方法对应的映射文件SQL语句中#{类型名小写[第几个元素（从0开始）]}进行取值

多个参数，MyBatis会将多个参数封装到一个map中，键值key是参数的索引（从0开始，即从0到参数个数减1）或者从param1到param参数个数，value为传入的参数值，接口方法对应的映射文件SQL语句中#{}是从map中获取指定key的值

多个参数的key可以设置命名参数指定封装参数的map的key，在接口方法的传入参数名前加入注解@Param("指定参数map的键值key")，接口方法对应的映射文件SQL语句中#{指定参数map的键值key} 获取对应的参数值

多个参数是POJO数据模型，可以直接传入POJO，接口方法对应的映射文件SQL语句中#{属性名}进行取值

多个参数不是POJO数据模型，可以封装成Map进行传入，接口方法对应的映射文件SQL语句中#{指定参数map的键值key}

多个参数不是POJO数据模型也可以使用TO（Transfer Object）数据传输对象

映射文件SQL语句中#{属性名}取值是以预编译的形式，将参数设置到SQL语句中，使用占位符?拼装SQL语句，PreparedStatement防止SQL注入漏洞

映射文件SQL语句中${属性名}取值是将取出的值直接拼装到SQL语句中，有安全问题

映射文件SQL语句中#{属性名,jdbcType=NULL}指定对应的SQL语句将属性名为空的值传递给数据库时以NULL类型传递，不设置默认是OTHER类型，Oracle不支持OTHER类型，也可以在全局配置中设置<setting name="jdbcTypeForNull" value="NULL"/>

接口类名 接口类对象=SQL会话.getMapper(接口类.class);获取SQL映射文件对应接口类对象

返回数据类型名 返回数据名=接口类对象.SQL语句对应的接口类方法(参数);

SqlSession与Connection类一样式非线程安全的，所以不能写成类的成员变量被多个对象进行共享

Eclipse的所有源码文件夹会合并到类路径下即bin目录下，多个源码文件夹中有相同的包名的文件最后会整合到同一目录下，只是在Eclipse的工程视觉上不同

MyBatis一级缓存（SqlSession级别的缓存）也称本地缓存，默认开启一级缓存，不能关闭

一级缓存失效情况：SqlSession不同、查询条件不同、查询之间有执行了改变数据库数据的操作（增删改）、手动清楚SqlSession缓存

SqlSession.clearCache();清楚SqlSession的缓存即一级缓存

二级缓存（全局缓存）是基于namespace级别的缓存，需要手动配置和开启，为了提高扩展性，MyBatis定义了缓存接口Cache，实现缓存接口可以自定义二级缓存，使用二级缓存需要在全局配置文件开启二级缓存，SQL映射文件中配置使用缓存，并且数据模型POJO需要实现序列化接口，查询的数据会默认被放到一级缓存中，只有一级缓存关闭时才会将一级缓存中的数据放到二级缓存里

MyBatis查询时使用缓存顺序，先去二级缓存，再去一级缓存，最后去数据库查询。

MyBatis的缓存是保存在对应的Map对象里