1. MyBatis面试题

1. 用过MyBatis吧，说一下MyBatis流程

1. 通过SqlSessionFactoryBuilder创建SqlSessionFactory：

   在SqlSessionFactoryBuilder的build()方法中使用XMLConfigBuilder用来解析配置文件MyBatis-config.xml，并将数据存放到Configuration对象中，然后创建并返回一个DefautSqlSessionFactory。

2. 通过SqlSessionFactory创建SqlSession，用于执行sql语句；
3. 通过SqlSession拿到Mapper对象的代理；
4. 通过MapperProxy调用Mapper中相应的方法；

2. MyBatis中的#{}和${}有什么区别？

- #{}内部使用预编译PreparedStatement机制执行SQL;
- ${}内部采用Statement机制将参数和SQL拼装在一起发送执行；
- 如果是字段位置，建议使用#{}
- 如果是表名或字段名位置，建议使用${}

```
SELECT xx
  FROM ${tableName}
 WHERE id = #{id}
 GROUP BY ${columnName}
 ORDER BY ${columnName}
```

3. 请说一下MyBatis的插件运行原理？怎么编写一个插件？

MyBatis使用JDK的动态代理，为需要拦截的接口生成代理对象以实现接口方法拦截功能，MyBatis仅可以编写针对ParameterHandler、ResultSetHandler、StatementHandler、Executor这4种接口的插件，每当执行这4种接口对象的方法时，就会进入拦截方法，具体就是InvocationHandler的invoke()方法，当然，它只会拦截那些你指定需要拦截的方法。

编写一个插件需要实现MyBatis的Interceptor接口并复写intercept()方法，然后在给插件编写注解，指定要拦截哪一个接口的哪些方法即可，最后需要在配置文件中配置编写的插件。

4. 通常一个Xml映射文件，都会写一个Mapper接口与之对应，这个接口的工作原理是什么？Mapper接口里的方法能重载吗？

1. Mapper接口的工作原理是JDK动态代理，MyBatis运行时会使用JDK动态代理为Dao接口生成代理proxy对象，代理对象proxy会拦截接口方法，转而执行MappedStatement所代表的sql，然后将sql执行结果返回；
2. Mapper接口里的方法，是不能重载的，因为是全限名+方法名的保存和寻找策略。
3. Mapper接口的全限名，是映射文件中的namespace的值，接口的方法名，就是映射文件中MappedStatement的id值，接口方法内的参数，就是传递给sql的参数；
4. Mapper接口是没有实现类的，当调用接口方法时，接口全限名+方法名拼接字符串作为key值，可唯一定位一个MappedStatement；

5. Xml映射文件中，除了常见的select|insert|updae|delete标签之外，还有哪些标签？

1. 还有很多其他的标签：

   `<resultMap>`、`<parameterMap>`、`<sql>`、`<include>`、`<selectKey>`

2. 另外，还有动态sql标签:

   `<trim>`、`<where>`、`<set>`、`<foreach>`、`<if>`、`<choose>`、`<when>`、`<otherwise>`、`<bind>` 等

   `<sql>` 为sql片段标签，通过 `<include>` 标签引入sql片段

   `<selectKey>` 为不支持自增的主键生成策略标签。

6. MyBatis都有哪些动态sql？

- MyBatis动态sql可以让我们在Xml映射文件内，以标签的形式编写动态sql，完成逻辑判断和动态拼接sql的功能；
- MyBatis提供了9种动态sql标签 `<trim>、<where>、<set>、<foreach>、<if>、<choose>、<when>、<otherwise>、<bind>` ；

7. MyBatis批量添加用的什么标签？动态标签foreach有哪些属性？

1. `<foreach>标签` ；
2. `<foreach>` 标签的属性有
    - collection：指定输入对象中集合属性
    - index：这个属性用来指定用来访问迭代集合下标的名称。如：index="myIndex"，则#{myIndex}用来访问当前迭代的下标。
    - item：每次遍历生成的对象
    - open：开始遍历时拼接的串
    - close：结束遍历时两个对象需要拼接的串
    - separator：用来分割foreach元素迭代的每个元素；

8. MyBatis是如何将sql执行结果封装为目标对象并返回的？MyBatis怎么生成实体类的？都有哪些映射形式？

1. 第一种是使用 `<resultMap>` 标签，逐一定义列名和对象属性名之间的映射关系；
2. 第二种是使用sql列的别名功能，将列别名书写为对象属性名；
3. 有了列名与属性名的映射关系后，MyBatis通过反射创建对象，同时给对象的属性逐一赋值并返回，那些找不到映射关系的属性，是无法完成赋值的。

9. MyBatis的Xml映射文件中，不同的Xml映射文件，id是否可以重复？

- 不同的Xml映射文件，如果配置了namespace，那么id可以重复；
- 由于namespace不是必须的，如果没有配置namespace，那么id不能重复；

  附：如果没有配置namespace，不能重复的原因：

  namespace + id是作为Map<String, MappedStatement>的key使用的，如果没有namespace，只剩下id，这时id重复会导致数据互相覆盖。

10. 为什么说MyBatis是半自动ORM映射工具？它与全自动的区别在哪里？

MyBatis在查询关联对象或关联集合对象时，需要手动编写sql来完成，所以称之为半自动ORM映射工具。

11.如果MyBatis写sql查询表中空字段会怎么样？有什么解决方案？

MyBatis使用resultMap来映射查询结果中的列，如果查询结果中包含空值的列，则MyBatis在映射的时候，不会映射这个字段

例如：查询name，sex，age，数据库中的age字段没有值，MyBatis返回的map中只映射了name和sex字段，而age字段则没有包含。

MyBatis不知道你传入的null参数对应的jdbc类型是什么，因为在MyBatis看来，null在数据库中可以为多种类型；

- 解决方案1：传入该参数的地方写明jdbc类型即可

  #{myNullParameter, jdbcType=VARCHAR}
- 解决方案2：使用MyBatis config配置

```
<configuration>
  <settings>
    <setting name="callSettersOnNulls" value="true"/>
  </settings>
</configuration>

<!-- 这时，如果是整合spring，需要在配置SqlSessionFactoryBean的地方引入这个配置文件 -->
```

```
<bean id="sqlSessionFactory"
class="org.MyBatis.spring.SqlSessionFactoryBean">
   <property name="dataSource" ref="dataSource" />
   <property name="configLocation" value="classpath:MyBatis-
config.xml" />
   <property name="mapperLocations" value="classpath:mappers/*.xml"
/>
   <property name="typeAliasesPackage" value="com.etoak.bean" />
</bean>
另外，配合springboot时的处理方式：在application.yml中配置
 mybatis:
   configuration:
     call-setters-on-nulls: true
```

- 方案三：整合Spring的时候，不使用MyBatis-config.xml，只修改SqlSessionFactoryBean的属性configurationProperties即可

```
<bean id="sqlSessionFactory"
class="org.MyBatis.spring.SqlSessionFactoryBean">
   <property name="dataSource" ref="dataSource" />
   <property name="mapperxmlLocations"
value="classpath:mappers/*.xml" />
   <!-- 不配置configLocation  -->
   <property name="configurationProperties">
     <props>
        <prop key="cacheEnabled">true</prop>
        <prop key="callSettersOnNulls">true</prop>
     </props>
   </property>
   <property name="typeAliasesPackage" value="com.etoak.bean" />
</bean>
```

- 解决方案四：如果想要配置空字段的默认值，创建一个类，实现MyBatis的TypeHandler接口

```
public class NullFieldHandler implements TypeHandler {
  @Override
  public String getResult(ResultSet rs, String columnName) throws
SQLException {
```

```
    return (rs.getString(columnName) == null) ? "" :
rs.getString(columnName);
  }

  @Override
  public String getResult(ResultSet rs, int columnIndex) throws
SQLException {
    return (rs.getString(columnIndex) == null) ? "" :
rs.getString(columnIndex);
  }

  @Override
  public String getResult(CallableStatement cs, int columnIndex)
throws SQLException {
    return (cs.getString(columnIndex) == null) ? "" :
cs.getString(columnIndex);
  }

  @Override
  public void setParameter(PreparedStatement ps, int arg1, String
str, JdbcType jdbcType) throws SQLException {
  }
}
```

在resultMap中使用，即可配置该字段的默认值(上述代码中age的默认值为"")

```
<resultMap id="list" type="java.util.LinkedHashMap">
  <result property="name" column="name" />
  <result property="sex" column="sex" />
  <result property="age" column="age"
          typeHandler="com.demo.EmptyStringIfNull"/>
</resultMap>
```

- 解决方案五：这种比较直接，直接给对象属性设置初始化值；

12. 说一下SqlsessionFactoryBean的作用？SqlSessionFactoryBean和SqlSessionFactory有什么区别？

- 在Spring整合MyBatis的时，我们需要配值SqlSessionFactoryBean来充当SqlSessionFactory；
- SqlSessionFactoryBean实现了Spring的FactoryBean接口，实现这个接口的时候，泛型处就是写的SqlSessionFactory，所以说实际使用SqlSessionFactoryBean时候，实际上在spring ioc容器中的Bean是SqlSessionFactory，当把这个bean注入到Spring中去了以后，IOC容器中的其他类型就可以拿到SqlSession实例了，进而就可执行SQL任务了。

  附：SqlSessionFactoryBean的定义

```
public class SqlSessionFactoryBean implements
FactoryBean<SqlSessionFactory>,
InitializingBean,
ApplicationListener<ApplicationEvent> {

}
```

13. 实体类属性和表字段不一致怎么处理？

目前有三种方案可以选择：

- 解决方案1：设定显示列的别名，让显示列的别名与实体类的属性名相同；

```
<select id="queryUserList" resultType="user">
  select name AS userName from t_user
</select>
```

- 解决方案2：使用resultMap自定义封装规则，它的优点是可以被重复的使用，弥补了方案1的缺点；

```
<resultMap type="user" id="userMap">
  <result column="name" property="userName"/>
</resultMap>
```

- 解决方案3：使用Map集合封装结果集中的数据，简单，可以重复使用，又无需额外维护封装规则，但是破坏了ORM的映射规则；不推荐

```
<select id="queryUserList" resultType="map">
  select * from t_user
</select>
```

14. 了解MyBatis的缓存吗？简单说一下。

MyBatis提供了一级缓存和二级缓存的支持；

一级缓存:

- 基于PerpetualCache的HashMap本地缓存，其存储作用域为SqlSession，当SqlSession flush或close之后，该Session中的所有Cache 就将清空；
- 另外不同的SqlSession之间的缓存数据是互相不影响的；

二级缓存：

- 二机缓存开启方式是在Mapper的配置文件配置；
- 与一级缓存其机制相同，不同在于其存储作用域为Mapper，并且可自定义存储源，如Ehcache；
- 二级缓存是跨SqlSession的，多个SqlSession可以共用二级缓存；
- 进行了“添加、更新和删除”操作后，会刷新对应的缓存的数据。

MyBatis查询顺序：二级缓存→一级缓存→数据库

MyBatis主要提供了以下几个刷新和置换策略：

- LRU：（ Least Recently Used ），最近最少使用算法，即如果缓存中容量已经满了，会将缓存中最近做少被使用的缓存记录清除掉，然后添加新的记录；

- FIFO：（ First in first out ），先进先出算法，如果缓存中的容量已经满了，那么会将最先进入缓存中的数据清除掉；
- Scheduled：指定时间间隔清空算法，该算法会以指定的某一个时间间隔将Cache缓存中的数据清空；
- 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，更积极地移除基于垃圾收集器状态和弱引用规则的对象。

15. 说一下MyBatis的selectKey标签的使用

selectKey这个标签主要用于解决“添加数据”时不支持主键自动生成的问题，他可以很随意的设置生成主键的方式；
selectKey需要注意order属性

- MySql这类支持自动增长类型的数据库中，order需要设置为after才会取到正确的值。
- Oracle这样取序列的情况，order需要设置为before，否则会报错。

16. 说一下MyBatis和Hibernate的区别

1. Hibernate是全自动，而MyBatis是半自动

   Hibernate完全可以通过对象关系模型实现对数据库的操作，拥有完整的JavaBean对象与数据库的映射结构来自动生成sql。

   MyBatis仅有基本的字段映射，对象数据以及对象实际关系仍然需要通过手写sql来实现和管理。

2. Hibernate数据库移植性大于MyBatis

   Hibernate通过它强大的映射结构和hql语言，大大降低了对象与数据库(oracle、mysql等)的耦合性；

   MyBatis由于需要手写sql，因此与数据库的耦合性取决于写sql的方式，如果sql不具通用性而用了很多某数据库特性的sql语句的话，移植性会降低很多，成本高。

3. Hibernate拥有完整的日志系统，MyBatis则欠缺一些

   Hibernate日志系统非常健全，包括：sql记录、关系异常、优化警告、缓存提示、脏数据警告等；

   MyBatis则除了基本记录功能外，其它功能薄弱很多。

4. MyBatis相比Hibernate需要关心很多细节

Hibernate配置要比MyBatis复杂的多，学习成本也比MyBatis高。

5. Sql直接优化上，MyBatis要比Hibernate方便很多
6. MyBatis的sql都是写在xml里，因此优化sql比hibernate方便很多。Hibernate的sql很多都是自动生成的，无法直接维护sql;