## 对数据库的基本操作步骤 + 面试题

MyBatis 最初的设计是基于 XML 配置文件的，但随着 Java 的发展（Java 1.5 开始引入注解）和 MyBatis 自身的迭代升级，终于在 MyBatis 3 之后就开始支持基于注解的开发了。

下面我们使用 Spring Boot + MyBatis 注解的方式，来实现对数据库的基本操作，具体实现步骤如下。

### MyBatis 注解版

#### 1）创建数据表

drop table if exists `t\_user`;  
create table `t\_user` (  
 `id` bigint(20) not null auto\_increment comment '主键id',  
 `username` varchar(32) default null comment '用户名',  
 `password` varchar(32) default null comment '密码',  
 `nick\_name` varchar(32) default null,  
 primary key (`id`)  
) engine=innodb auto\_increment=1 default charset=utf8;

#### 2）添加依赖

<!-- https://mvnrepository.com/artifact/org.mybatis.spring.boot/mybatis-spring-boot-starter -->  
<dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <version>2.1.0</version>  
</dependency>  
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  
<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.16</version>  
</dependency>

#### 3）增加配置文件

在 application.yml 文件中添加以下内容：

spring:  
 datasource:  
 url: jdbc:mysql://localhost:3306/learndb?serverTimezone=UTC&useUnicode=true&characterEncoding=utf-8&useSSL=true  
 username: root  
 password: root  
 driver-class-name: com.mysql.cj.jdbc.Driver  
mybatis:  
 type-aliases-package: com.interview.model

#### 4）创建实体类

public class UserEntity implements Serializable {  
 private static final long serialVersionUID = -5980266333958177105L;  
 private Integer id;  
 private String userName;  
 private String passWord;  
 private String nickName;  
 public UserEntity(String userName, String passWord, String nickName) {  
 this.userName = userName;  
 this.passWord = passWord;  
 this.nickName = nickName;  
 }  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public String getUserName() {  
 return userName;  
 }  
 public void setUserName(String userName) {  
 this.userName = userName;  
 }  
 public String getPassWord() {  
 return passWord;  
 }  
 public void setPassWord(String passWord) {  
 this.passWord = passWord;  
 }  
 public String getNickName() {  
 return nickName;  
 }  
 public void setNickName(String nickName) {  
 this.nickName = nickName;  
 }  
}

#### 5）增加 Mapper 文件

public interface UserMapper {  
 @Select("select \* from t\_user")  
 @Results({  
 @Result(property = "nickName", column = "nick\_name")  
 })  
 List<UserEntity> getAll();  
  
 @Select("select \* from t\_user where id = #{id}")  
 @Results({  
 @Result(property = "nickName", column = "nick\_name")  
 })  
 UserEntity getOne(Long id);  
  
 @Insert("insert into t\_user(username,password,nick\_name) values(#{userName}, #{passWord}, #{nickName})")  
 void insert(UserEntity user);  
  
 @Update("update t\_user set username=#{userName},nick\_name=#{nickName} where id =#{id}")  
 void update(UserEntity user);  
  
 @Update({"<script> ",  
 "update t\_user ",  
 "<set>",  
 " <if test='userName != null'>userName=#{userName},</if>",  
 " <if test='nickName != null'>nick\_name=#{nickName},</if>",  
 " </set> ",  
 "where id=#{id} ",  
 "</script>"})  
 void updateUserEntity(UserEntity user);  
  
 @Delete("delete from t\_user where id =#{id}")  
 void delete(Long id);  
}

使用 @Select、@Insert、@Update、@Delete、@Results、@Result 等注解来替代 XML 配置文件。

#### 6）添加 Mapper 包扫描

在启动类中添加 @MapperScan，设置 Spring Boot 启动的时候会自动加载包路径下的 Mapper。

@SpringBootApplication  
@MapperScan("com.interview.mapper")  
public class MybatisApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(MybatisApplication.class, args);  
 }  
}

#### 7）编写测试代码

@RunWith(SpringRunner.class)  
@SpringBootTest  
public class MybatisApplicationTests {  
 @Autowired  
 private UserMapper userMapper;  
 @Test  
 public void testInsert() {  
 userMapper.insert(new UserEntity("laowang", "123456", "老王"));  
 Assert.assertEquals(1, userMapper.getAll().size());  
 }  
}

### 相关面试题

#### 1.MyBatis 有哪些优缺点？

答：MyBatis 优缺点如下：

优点：

* 相比于 JDBC 需要编写的代码更少
* 使用灵活，支持动态 SQL
* 提供映射标签，支持对象与数据库的字段关系映射

缺点：

* SQL 语句依赖于数据库，数据库移植性差
* SQL 语句编写工作量大，尤其在表、字段比较多的情况下

总体来说，MyBatis 是一个非常不错的持久层解决方案，它专注于 SQL 本身，非常灵活，适用于需求变化较多的互联网项目，也是当前国内主流的 ORM 框架。

#### 2.以下不属于 MyBatis 优点的是？

A：可以灵活的编辑 SQL 语句  
B：很好的支持不同数据库之间的迁移  
C：能够很好的和 Spring 框架集成  
D：提供映射标签支持对象和数据库的字段映射

答：B

题目解析：因为 MyBatis 需要自己编写 SQL 语句，但每个数据库的 SQL 语句有略有差异，所以 MyBatis 不能很好的支持不同数据库之间的迁移。

#### 3.MyBatis 和 Hibernate 有哪些不同？

答：MyBatis 和 Hibernate 都是非常优秀的 ORM 框架，它们的区别如下：

* 灵活性：MyBatis 更加灵活，自己可以写 SQL 语句，使用起来比较方便；
* 可移植性：MyBatis 有很多自己写的 SQL，因为每个数据库的 SQL 可以不相同，所以可移植性比较差；
* 开发效率：Hibernate 对 SQL 语句做了封装，让开发者可以直接使用，因此开发效率更高；
* 学习和使用门槛：MyBatis 入门比较简单，使用门槛也更低。

#### 4.“#”和“$”有什么区别？

答：“#”是预编译处理，“$”是字符替换。 在使用“#”时，MyBatis 会将 SQL 中的参数替换成“?”，配合 PreparedStatement 的 set 方法赋值，这样可以有效的防止 SQL 注入，保证程序的运行安全。

#### 5.在 MyBatis 中怎么解决实体类属性名和表字段名不一致的问题？

答：通常的解决方案有以下两种方式。

① 在 SQL 语句中重命名为实体类的属性名，可参考以下配置：

<select id="selectorder" parametertype="int" resultetype="com.interview.order">  
 select order\_id id, order\_no orderno form order where order\_id=#{id};  
</select>

② 通过 <resultMap> 映射对应关系，可参考以下配置：

<resultMap id="BaseResultMap" type="com.interview.mybatislearning.model.UserEntity" >  
 <id column="id" property="id" jdbcType="BIGINT" />  
 <result column="username" property="userName" jdbcType="VARCHAR" />  
 <result column="password" property="passWord" jdbcType="VARCHAR" />  
 <result column="nick\_name" property="nickName" jdbcType="VARCHAR" />  
</resultMap>  
 <select id="getAll" resultMap="BaseResultMap">  
 select \* from t\_user  
</select>

#### 6.在 MyBatis 中如何实现 like 查询？

答：可以在 Java 代码中添加 SQL 通配符来实现 like 查询，这样也可以有效的防治 SQL 注入，具体实现如下：

Java 代码：

String name = "%wang%":  
List<User> list = mapper.likeName(name);

Mapper 配置：

<select id="likeName">  
 select \* form t\_user where name like #{name};  
</select>

#### 7.MyBatis 有几种分页方式？

答：MyBatis 的分页方式有以下两种：

* 逻辑分页，使用 MyBatis 自带的 RowBounds 进行分页，它是一次性查询很多数据，然后在数据中再进行检索；
* 物理分页，自己手写 SQL 分页或使用分页插件 PageHelper，去数据库查询指定条数的分页数据形式。

#### 8.RowBounds 是一次性查询全部结果吗？为什么？

答：RowBounds 表面是在“所有”数据中检索数据，其实并非是一次性查询出所有数据。因为 MyBatis 是对 JDBC 的封装，在 JDBC 驱动中有一个 Fetch Size 的配置，它规定了每次最多从数据库查询多少条数据，假如你要查询更多数据，它会在执行 next() 的时候，去查询更多的数据。 就好比你去自动取款机取 10000 元，但取款机每次最多能取 2500 元，要取 4 次才能把钱取完。对于 JDBC 来说也是一样，这样做的好处是可以有效的防止内存溢出。

#### 9.为什么阿里巴巴不允许使用 HashMap 或 Hashtable 作为查询结果集直接输出？

答：因为使用 HashMap 或 Hashtable 作为查询结果集直接输出，会导致值类型不可控，给调用人员造成困扰，给系统带来更多不稳定的因素。

#### 10.什么是动态 SQL？

答：动态 SQL 是指可以根据不同的参数信息来动态拼接的不确定的 SQL 叫做动态 SQL，MyBatis 动态 SQL 的主要元素有：if、choose/when/otherwise、trim、where、set、foreach 等。 以 if 标签的使用为例：

<select id="findUser" parameterType="com.interview.entity.User" resultType="com.interview.entity.User">  
 select \* from t\_user where  
 <if test="id!=null">  
 id = #{id}  
 </if>  
 <if test="username!=null">  
 and username = #{username}  
 </if>  
 <if test="password!=null">  
 and password = #{password}  
 </if>  
</select>

#### 11.为什么不建议在程序中滥用事务？

答：因为事务的滥用会影响数据的 QPS（每秒查询率），另外使用事务的地方还要考虑各方面回滚的方案，如缓存回滚、搜索引擎回滚、消息补偿、统计修正等。

#### 12.如何开启 MyBatis 的延迟加载？

答：只需要在 mybatis-config.xml 设置 <setting name="lazyLoadingEnabled" value="true"/> 即可打开延迟缓存功能，完整配置文件如下：

<configuration>  
 <settings>  
 <!-- 开启延迟加载 -->  
 <setting name="lazyLoadingEnabled" value="true"/>  
 </settings>  
</configuration>

#### 13.什么是 MyBatis 的一级缓存和二级缓存？

答：MyBatis 缓存如下：

* 一级缓存是 SqlSession 级别的，是 MyBatis 自带的缓存功能，并且无法关闭，因此当有两个 SqlSession 访问相同的 SQL 时，一级缓存也不会生效，需要查询两次数据库；
* 二级缓存是 Mapper 级别的，只要是同一个 Mapper，无论使用多少个 SqlSession 来操作，数据都是共享的，多个不同的 SqlSession 可以共用二级缓存，MyBatis 二级缓存默认是关闭的，需要使用时可手动开启，二级缓存也可以使用第三方的缓存，比如，使用 Ehcache 作为二级缓存。

手动开启二级缓存，配置如下：

<configuration>  
 <settings>  
 <!-- 开启二级缓存 -->  
 <setting name="cacheEnabled" value="true"/>  
 </settings>  
</configuration>

#### 14.如何设置 Ehcache 为 MyBatis 的二级缓存？

答：可直接在 XML 中配置开启 EhcacheCache，代码如下：

<mapper namespace="com.interview.repository.ClassesReposirory">   
 <!-- 开启二级缓存 -->  
 <cache type="org.mybatis.caches.ehcache.EhcacheCache" >  
 <!-- 缓存创建以后，最后一次访问缓存的时间至失效的时间间隔 -->  
 <property name="timeToIdleSeconds" value="3600"/>  
 <!-- 缓存自创建时间起至失效的时间间隔-->  
 <property name="timeToLiveSeconds" value="3600"/>  
 <!-- 缓存回收策略，LRU 移除近期最少使用的对象 -->  
 <property name="memoryStoreEvictionPolicy" value="LRU"/>  
 </cache>  
  
 <select id="findById" parameterType="java.lang.Long" resultType="com.interview.entity.Classes">  
 select \* from classes where id = #{id}  
 </select>  
</mapper>

#### 15.MyBatis 有哪些拦截器？如何实现拦截功能？

答：MyBatis 提供的连接器有以下 4 种。

* Executor：拦截内部执行器，它负责调用 StatementHandler 操作数据库，并把结果集通过 ResultSetHandler 进行自动映射，另外它还处理了二级缓存的操作。
* StatementHandler：拦截 SQL 语法构建的处理，它是 MyBatis 直接和数据库执行 SQL 脚本的对象，另外它也实现了 MyBatis 的一级缓存。
* ParameterHandler：拦截参数的处理。
* ResultSetHandler：拦截结果集的处理。

拦截功能具体实现如下：

@Intercepts({@Signature(type = Executor.class, method = "query",  
 args = {MappedStatement.class, Object.class, RowBounds.class, ResultHandler.class})})  
public class TestInterceptor implements Interceptor {  
 public Object intercept(Invocation invocation) throws Throwable {  
 Object target = invocation.getTarget(); //被代理对象  
 Method method = invocation.getMethod(); //代理方法  
 Object[] args = invocation.getArgs(); //方法参数  
 // 方法拦截前执行代码块  
 Object result = invocation.proceed();  
 // 方法拦截后执行代码块  
 return result;  
 }  
 public Object plugin(Object target) {  
 return Plugin.wrap(target, this);  
 }  
}

### 总结

通过本文可以看出 MyBatis 注解版和 XML 版的主要区别是 Mapper 中的代码，注解版把之前在 XML 的 SQL 实现，全部都提到 Mapper 中了，这样就省去了配置 XML 的麻烦。

[点击此处下载本文源码](https://github.com/vipstone/java-%20interview/tree/master/mybatislearning)

## 更多资源下载交流请加微信：Morstrong,加入永久会员,网盘更新更快捷！

# 本资源由微信公众号：光明顶一号，提供支持