## MyBatis 核心组件 + 面试题

### MyBatis 介绍

MyBatis 是一款优秀的 ORM（Object Relational Mapping，对象关系映射）框架，它可以通过对象和数据库之间的映射，将程序中的对象自动存储到数据库中。它是 Apache 提供的一个开源项目，之前的名字叫做 iBatis，2010 年迁移到了 Google Code，并且将名字改为我们现在所熟知的 MyBatis，又于 2013 年 11 月迁移到了 Github。

MyBatis 提供了普通 SQL 查询、事务、存储过程等功能，它的优缺点如下。

**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 是一个非常优秀和灵活的数据持久化框架，适用于需求多变的互联网项目，也是当前主流的 ORM 框架。

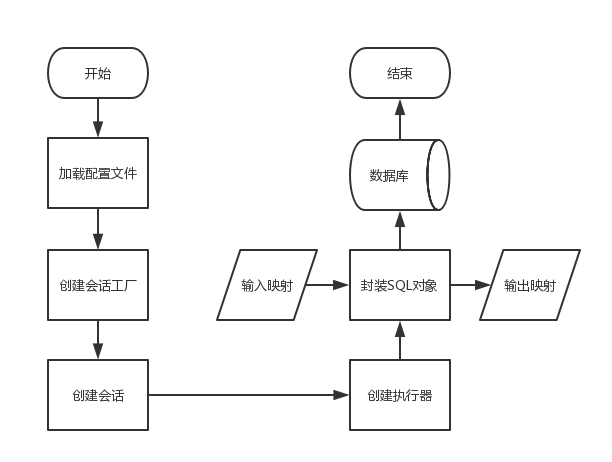
#### MyBatis 重要组件

MyBatis 中的重要组件如下：

* Mapper 配置：用于组织具体的查询业务和映射数据库的字段关系，可以使用 XML 格式或 Java 注解格式来实现；
* Mapper 接口：数据操作接口也就是通常说的 DAO 接口，要和 Mapper 配置文件中的方法一一对应；
* Executor：MyBatis 中所有的 Mapper 语句的执行都是通过 Executor 执行的；
* SqlSession：类似于 JDBC 中的 Connection，可以用 SqlSession 实例来直接执行被映射的 SQL 语句；
* SqlSessionFactory：SqlSessionFactory 是创建 SqlSession 的工厂，可以通过 SqlSession openSession() 方法创建 SqlSession 对象。

#### MyBatis 执行流程

MyBatis 完整执行流程如下图所示：



1

MyBatis 执行流程说明：

1. 首先加载 Mapper 配置的 SQL 映射文件，或者是注解的相关 SQL 内容。
2. 创建会话工厂，MyBatis 通过读取配置文件的信息来构造出会话工厂（SqlSessionFactory）。
3. 创建会话，根据会话工厂，MyBatis 就可以通过它来创建会话对象（SqlSession），会话对象是一个接口，该接口中包含了对数据库操作的增、删、改、查方法。
4. 创建执行器，因为会话对象本身不能直接操作数据库，所以它使用了一个叫做数据库执行器（Executor）的接口来帮它执行操作。
5. 封装 SQL 对象，在这一步，执行器将待处理的 SQL 信息封装到一个对象中（MappedStatement），该对象包括 SQL 语句、输入参数映射信息（Java 简单类型、HashMap 或 POJO）和输出结果映射信息（Java 简单类型、HashMap 或 POJO）。
6. 操作数据库，拥有了执行器和 SQL 信息封装对象就使用它们访问数据库了，最后再返回操作结果，结束流程。

### MyBatis XML 版

MyBatis 使用分为两个版本：XML 版和 Java 注解版。接下来我们使用 Spring Boot 结合 MyBatis 的 XML 版，来实现对数据库的基本操作，步骤如下。

#### 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

在项目添加对 MyBatis 和 MySQL 支持的依赖包，在 pom.xml 文件中添加如下代码：

<!-- 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>

mybatis-spring-boot-starter 是 MyBatis 官方帮助我们快速集成 Spring Boot 提供的一个组件包，mybatis- spring-boot-starter 2.1.0 对应 MyBatis 的版本是 3.5.2。

#### 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:  
 config-location: classpath:mybatis/mybatis-config.xml  
 mapper-locations: classpath:mybatis/mapper/\*.xml  
 type-aliases-package: com.interview.mybatislearning.model

其中：

* mybatis.config-location：配置 MyBatis 基础属性；
* mybatis.mapper-locations：配置 Mapper 对应的 XML 文件路径；
* mybatis.type-aliases-package：配置项目中实体类包路径。

注：如果配置文件使用的是 application.properties，配置内容是相同的，只是内容格式不同。

#### 4）创建实体类

public class UserEntity implements Serializable {  
 private static final long serialVersionUID = -5980266333958177104L;  
 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）创建 XML 文件

**mybatis-config.xml** （基础配置文件）：

<?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>  
 <typeAliases>  
 <typeAlias alias="Integer" type="java.lang.Integer"/>  
 <typeAlias alias="Long" type="java.lang.Long"/>  
 <typeAlias alias="HashMap" type="java.util.HashMap"/>  
 <typeAlias alias="LinkedHashMap" type="java.util.LinkedHashMap"/>  
 <typeAlias alias="ArrayList" type="java.util.ArrayList"/>  
 <typeAlias alias="LinkedList" type="java.util.LinkedList"/>  
 </typeAliases>  
</configuration>

mybatis-config.xml 主要是为常用的数据类型设置别名，用于减少类完全限定名的长度，比如：resultType="Integer" 完整示例代码如下：

<select id="getAllCount" resultType="Integer">  
 select  
 count(\*)  
 from t\_user  
</select>

**UserMapper.xml** （业务配置文件）：

<?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.interview.mybatislearning.mapper.UserMapper">  
 <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>  
 <sql id="Base\_Column\_List" >  
 id, username, password, nick\_name  
 </sql>  
 <sql id="Base\_Where\_List">  
 <if test="userName != null and userName != ''">  
 and userName = #{userName}  
 </if>  
 </sql>  
 <select id="getAll" resultMap="BaseResultMap" >  
 SELECT  
 <include refid="Base\_Column\_List" />  
 FROM t\_user  
 </select>  
 <select id="getOne" parameterType="Long" resultMap="BaseResultMap" >  
 SELECT  
 <include refid="Base\_Column\_List" />  
 FROM t\_user  
 WHERE id = #{id}  
 </select>  
 <insert id="insert" parameterType="com.interview.mybatislearning.model.UserEntity" >  
 INSERT INTO  
 t\_user  
 (username,password,nick\_name)  
 VALUES  
 (#{userName}, #{passWord}, #{nickName})  
 </insert>  
 <update id="update" parameterType="com.interview.mybatislearning.model.UserEntity" >  
 UPDATE  
 t\_user  
 SET  
 <if test="userName != null">username = #{userName},</if>  
 <if test="passWord != null">password = #{passWord},</if>  
 nick\_name = #{nickName}  
 WHERE  
 id = #{id}  
 </update>  
 <delete id="delete" parameterType="Long" >  
 DELETE FROM  
 t\_user  
 WHERE  
 id =#{id}  
 </delete>  
</mapper>

以上配置我们增加了增删改查等基础方法。

#### 6）增加 Mapper 文件

此步骤我们需要创建一个与 XML 对应的业务 Mapper 接口，代码如下：

public interface UserMapper {  
 List<UserEntity> getAll();  
 UserEntity getOne(Long id);  
 void insert(UserEntity user);  
 void update(UserEntity user);  
 void delete(Long id);  
}

#### 7）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.mybatislearning.mapper")  
public class MyBatisLearningApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(MyBatisLearningApplication.class, args);  
 }  
}

#### 8）编写测试代码

经过以上步骤之后，整个 MyBatis 的集成就算完成了。接下来我们写一个单元测试，验证一下。

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

### 总结

通过本文我们知道 MyBatis 是一个优秀和灵活的数据持久化框架，MyBatis 包含 Mapper 配置、Mapper 接口、Executor、SqlSession、SqlSessionFactory 等几个重要的组件，知道了 MyBatis 基本流程：MyBatis 首先加载 Mapper 配置和 SQL 映射文件，通过创建会话工厂得到 SqlSession 对象，再执行 SQL 语句并返回操作信息。我们也使用 XML 的方式，实现了 MyBatis 对数据库的基础操作。

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

更多java学习资料如下：

[Java高级架构师课程(总共122门课程，1460GB)](https://www.consultdog.com/index.php/front/44/)

[最强java面试视频课程(21门课程，126GB)](https://www.consultdog.com/index.php/front/78/)