MyBatis

什么是MyBatis框架?

MyBatis是apache的一个开源项目iBatis，iBatis是MyBatis的前身。

简单来说，MyBatis是一个实现了数据持久化的开源框架，支持三种高级编程语言，Java、.NET、Ruby，MyBatis可以简单理解为对JDBC的一个封装框架。

MyBatis和Hibernate框架的区别和共同点。

MyBatis 和 Hibernate 都是 ORM 框架。

ORM：Object Relationship Mapping 对象关系映射，将面向对象的编程语言与关系型数据库进行映射，将 Java 对象与 MySQL 数据表中的记录进行映射，一个 Java 对象就会对应 MySQL 某张数据表中的某一条记录。

除了 MyBatis 、Hibernate，Spring Data MongoDB 也是一个 ORM 框架。

MyBatis 是“半自动化”的 ORM 框架，这里的“半自动化”是相对于 Hibernate 提供了全面的数据库封装机制的“全自动化” ORM实现而言。

传统的 JDBC 查询数据的方式：

- 获取 Connection。
- 创建 SQL 语句： select * from user;
- 创建 Statement 对象来执行 SQL，并且将结果封装成一个 ResultSet 对象。
- 解析 ResultSet 对象，转化成 Java 对象。

使用 Hibernate：

- 加载 Hibernate 相关配置。
- 调用 Hibernate 提供的 API 获取对象。

使用 MyBatis：

- 需要在xml文件中配置 SQL 语句。
- 加载 MyBatis 相关配置。
- 调用 MyBatis 提供的 API 获取对象。

“全自动”的 Hibernate 框架实现了 Java 对象和数据库表之间的映射，以及 SQL 的自动生成和执行，结果解析。

“半自动”的 MyBatis 框架的关注点，在于 Java 对象和 SQL 之间的映射关系。

MyBatis 消除了几乎所有的 JDBC 代码和参数的手动设置以及对结果集的检索，MyBatis 可以使用简单的 XML 或注解用于配置和映射，通过接口将 Java 对象映射成数据库表中的记录。

MyBatis 的优点：

- 大大简化了 JDBC 代码的开发。
- MyBatis 是最简单的持久化框架，小巧并且灵活，简单易学。
- MyBatis 非常灵活，不会对应用程序或者数据库的现有设计强加任何影响，SQL 写在 XML 中，从

代码中彻底分离，降低程序的耦合性，便于统一管理和优化，可重用性高。
- 提供 XML 标签，支持编写动态 SQL 语句。
- 提供映射标签，支持对象与数据库的 ORM 字段关系映射。

MyBatis 的缺点：

- SQL 语句的编写工作量较大，尤其是多字段、多表关联更是如此，对开发人员编写 SQL 语句的功底有一定要求。
- SQL 语句依赖于底层数据库，导致数据库移植性很差，一旦更换底层数据库，所有的 SQL 需要重写编写。

如何使用?

- 搭建环境，创建 Maven 工程，pom.xml 引入相关依赖。

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.5</version>
</dependency>

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.11</version>
</dependency>
```

- 新建数据表。

```
create table t_user(
    id int primary key auto_increment,
    username varchar(11),
    password varhcar(11),
    age int
);
```

- 创建对应的实体类 User

```
package com.southwind.entity;

public class User {
    private int id;
    private String username;
    private String password;
    private int age;

    public int getId() {
        return id;
    }
```

```
    public void setId(int 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 int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

- 创建 MyBatis 的配置文件 config.xml（文件名可自定义）。

```
<?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>
    <!-- 配置 MyBatis 的运行环境 -->
    <environments default="development">
        <environment id="development">
            <!-- 配置 JDBC 事务管理 -->
            <transactionManager type="JDBC"></transactionManager>
            <!-- 配置数据库连接信息 -->
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.cj.jdbc.Driver">
</property>
                <property name="url"
value="jdbc:mysql://localhost:3306/dbname?
useUnicode=true&amp;characterEncoding=UTF-8"></property>
                <property name="username" value="root"></property>
```

```
                <property name="password" value="root"></property>
            </dataSource>
        </environment>
    </environments>

</configuration>
```

- MyBatis 有两种开发方式：
  - 使用原生接口
  - Mapper 代理实现自定义接口
- 使用原生接口
  - 创建 Mapper 文件 UserMapper.xml

    namespace 通常设置为文件所在包+文件名。

    select 标签表示执行查询操作。

    id 是调用接口时关联的属性。

    parameterType 是参数数据类型。

    resultType 是返回值数据类型。

```
<?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.southwind.mapper.UserMapper">

    <select id="get" parameterType="int"
resultType="com.southwind.entity.User">
        select * from t_user where id = #{id}
    </select>

</mapper>
```

  - 在全局配置文件 config.xml 中注册 UserMapper.xml。

```
<?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>
    <!-- 配置 MyBatis 的运行环境 -->
    <environments default="development">
        <environment id="development">
            <!-- 配置 JDBC 事务管理 -->
            <transactionManager type="JDBC"></transactionManager>
            <!-- 配置数据库连接信息 -->
            <dataSource type="POOLED">
                <property name="driver"
value="com.mysql.cj.jdbc.Driver"></property>
```

```
                <property name="url"
value="jdbc:mysql://localhost:3306/mbtest?
useUnicode=true&amp;characterEncoding=UTF-8"></property>
                <property name="username" value="root"></property>
                <property name="password" value="19900310"></property>
            </dataSource>
        </environment>
    </environments>

    <!-- 注册 UserMapper.xml -->
    <mappers>
        <mapper resource="com/southwind/mapper/UserMapper.xml"></mapper>
    </mappers>

</configuration>
```

- 调用原生接口执行 SQL 语句获取结果。

```
package com.southwind.test;

import com.southwind.entity.User;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.InputStream;

public class Test {
    public static void main(String[] args) {
        //加载 MyBatis 配置文件
        InputStream inputStream =
Test.class.getClassLoader().getResourceAsStream("config.xml");
        SqlSessionFactoryBuilder sqlSessionFactoryBuilder = new
SqlSessionFactoryBuilder();
        SqlSessionFactory sqlSessionFactory =
sqlSessionFactoryBuilder.build(inputStream);
        //获取 SqlSession 对象
        SqlSession sqlSession = sqlSessionFactory.openSession();
        //调用 MyBatis 原生接口执行 SQL
        //statement 对应 UserMapper.xml 的 namespace 值 +"."+ select 标签
的 id 值
        String statement = "com.southwind.mapper.UserMapper.get";
        User user = sqlSession.selectOne(statement,1);
        System.out.println(user);
    }
}
```

- 使用 IDEA 进行开发时，需要在 pom.xml 中添加如下配置，否则 XML 文件无法解析。

```
<build>

    <resources>
        <resource>
            <directory>src/main/java</directory>
            <includes>
                <include>**/*.xml</include>
            </includes>
        </resource>
        <resource>
            <directory>src/main/resources</directory>
            <includes>
                <include>*.xml</include>
                <include>*.properties</include>
            </includes>
        </resource>
    </resources>

</build>
```

实际开发中，推荐使用第二种方式：自定义接口，但是不需要实现该接口，通过 Mapper 代理的形式来实现，具体的实现逻辑或者说要执行的 SQL 语句配置在 Mapper.xml 中。

- 创建接口。

```
package com.southwind.repository;

import com.southwind.entity.User;

import java.util.List;

public interface UserRepository {

    /**
     * 新增用户
     * @param user
     * @return
     */
    public int save(User user);

    /**
     * 删除用户
     * @param id
     * @return
     */
    public int deleteById(int id);

    /**
     * 修改用户
```

```
     * @param user
     * @return
     */
    public int update(User user);

    /**
     * 通过 id 查询用户
     * @param id
     * @return
     */
    public User findById(int id);

    /**
     * 查询全部用户
     * @return
     */
    public List<User> findAll();
}
```

- 创建 UserRepository.xml，定义接口方法对应的 SQL 语句。statement 标签根据 SQL 执行的业务可选择 insert、delete、update、select。

  MyBatis 框架会根据规则自动创建 UserRepository 接口实现类的代理对象。

  规则如下：

  - UserRepository.xml 中的 namespace 为接口的全类名。
  - UserRepository.xml 中的 statement 的 id 为接口中对应的方法名。
  - UserRepository.xml 中的 statement 的 parameterType 和接口中对应方法的参数类型一致。
  - UserRepository.xml 中的 statement 的 resultType 和接口中对应方法的返回值类型一致。

```
<?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.southwind.repository.UserRepository">

    <insert id="save" parameterType="com.southwind.entity.User">
        insert into t_user(username,password,age) values(#{username},#
{password},#{age});
    </insert>

</mapper>
```

- 在 config.xml 中注册 UserRepository.xml

```
<?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>
```

```
    <!-- 配置 MyBatis 的运行环境 -->
    <environments default="development">
        <environment id="development">
            <!-- 配置 JDBC 事务管理 -->
            <transactionManager type="JDBC"></transactionManager>
            <!-- 配置数据库连接信息 -->
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.cj.jdbc.Driver">
</property>
                <property name="url"
value="jdbc:mysql://localhost:3306/mbtest?
useUnicode=true&amp;characterEncoding=UTF-8"></property>
                <property name="username" value="root"></property>
                <property name="password" value="19900310"></property>
            </dataSource>
        </environment>
    </environments>

    <!-- 注册 UserMapper.xml -->
    <mappers>
        <mapper resource="com/southwind/repository/UserRepository.xml">
</mapper>
    </mappers>

</configuration>
```

- 测试

```
package com.southwind.test;

import com.southwind.entity.User;
import com.southwind.repository.UserRepository;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.InputStream;

public class Test {
    public static void main(String[] args) {
        //加载 MyBatis 配置文件
        InputStream inputStream =
Test.class.getClassLoader().getResourceAsStream("config.xml");
        SqlSessionFactoryBuilder sqlSessionFactoryBuilder = new
SqlSessionFactoryBuilder();
        SqlSessionFactory sqlSessionFactory =
sqlSessionFactoryBuilder.build(inputStream);
        //获取 SqlSession 对象
        SqlSession sqlSession = sqlSessionFactory.openSession();
```

```
        //调用 MyBatis 原生接口执行 SQL
        //statement 对应 UserMapper.xml 的 namespace 值 +"."+ select 标签的 id
值
//        String statement = "com.southwind.mapper.UserMapper.get";
//        User user = sqlSession.selectOne(statement,1);
//        System.out.println(user);

        //调用自定义的 UserRepository 接口
        //获取实现接口的代理对象
        UserRepository userRepository =
sqlSession.getMapper(UserRepository.class);

        //新增用户
        User user = new User();
        user.setUsername("李四");
        user.setPassword("123123");
        user.setAge(23);
        int result = userRepository.save(user);
        System.out.println(result);
        sqlSession.commit();
    }
}
```