MyBatis框架

一、MyBatis简介

1、MyBatis概述

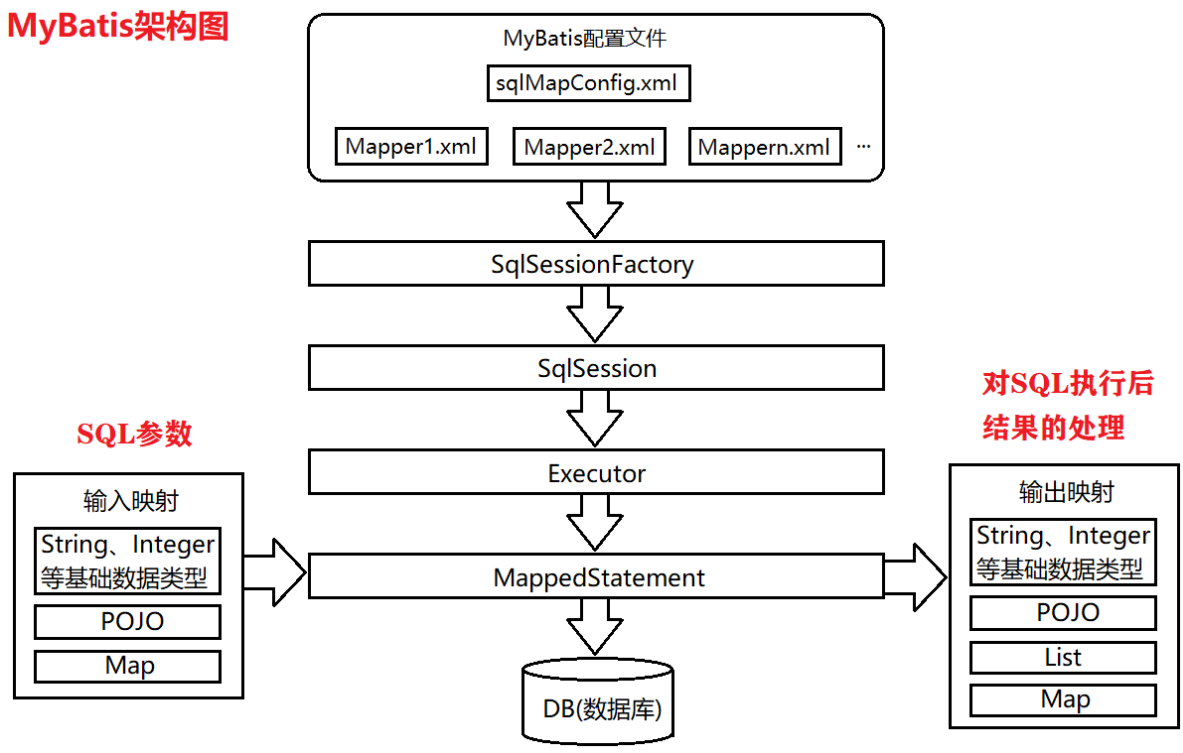
MyBatis 本是apache的一个开源项目iBatis, 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。2013年11月迁移到Github。

MyBatis是一个优秀的持久层框架，它对jdbc的操作数据库的过程进行封装，使开发者只需要关注 SQL 本身，而不需要花费精力去处理例如注册驱动、创建connection、创建statement、手动设置参数、结果集检索等jdbc繁杂的过程代码。

Mybatis通过xml或注解的方式将要执行的各种statement（statement、preparedStatemnt）配置起来，并通过java对象和statement中的sql进行映射生成最终执行的sql语句，最后由mybatis框架执行sql并将结果映射成java对象并返回

总结：mybatis对JDBC访问数据库进行了封装，简化了JDBC操作，解决了jdbc将结果映射为Java对象的麻烦

2、MyBatis架构图



- mybati-config.xml是mybatis的核心配置文件，通过配置文件（核心配置文件和映射配置文件）可以生成SqlSessionFactory，也就是SqlSession工厂
- 基于SqlSessionFactory可以生成SqlSesiion对象
- SqlSession可以发送sql语句去执行，得到返回结果，类似JDBC中的connection，是mybatis中至关重要的类
- Executor是SqlSession底层实现，用来执行sql语句
- MappedStatement也是SqlSession底层实现，用来接收输入映射（也就是sql语句中的参数），将查询的结果映射为相应的结果

二、为什么要使用MyBatis框架

思考：通过JDBC查询数据库表tb_user中的所有记录，将查询结果封装到List<User>并返回

```
public class JdbcTest {
    // 通过JDBC查询数据库表tb_user中的所有记录，将查询结果封装到List<User>并返回
    public List<User> findAll() {
        // 声明变量
        List<User> userList = new ArrayList<>();

        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;

        try {
            // 1、注册驱动
            Class.forName("com.mysql.cj.jdbc.Driver");

            // 2、获取连接
            String url = "jdbc:mysql:///hbnu?
serverTimezon=GMT&useSSL=false&characterEncoding=utf-8";
            String username = "root";
            String password = "chendikai";
            conn = DriverManager.getConnection(url, username, password);

            // 3、获取数据库操作对象
            String sql = "select * from tb_user";
            ps = conn.prepareStatement(sql);

            // 4、执行sql语句
            rs = ps.executeQuery();

            // 5、处理查询结果集
            while (rs.next()) {
                User user = new User();

                String username = rs.getString("username");
                String password = rs.getString("password");
                String email = rs.getString("email");

                user.setUsername(username);
                user.SetPassword(password);
                user.SetEmail(email);

                userList.add(user);
            }

            return userList;
        } catch(SQLException e) {
            e.printStackTrace();
        } finally {
            // 6、释放数据库资源
            rs.close();
            ps.close();
            conn.close();

        }
```

```
53      }
54  }
```

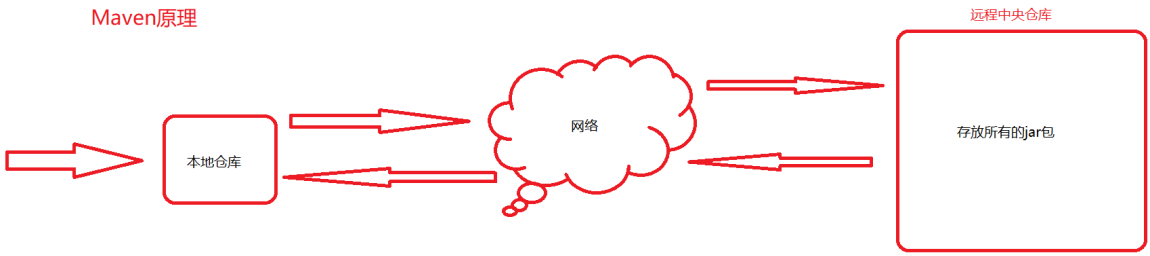
- JDBC访问数据库存在大量重复代码（比如注册驱动、获取连接、释放资源等等）
- JDBC自身不支持数据库连接池，会频繁创建连接、断开连接，这个操作比较耗资源，效率低
- sql语句是写在程序里面的，一旦sql语句发送改变，需要重新编译类
- JDBC对于结果集的处理，需要手动进行处理，有时候比较麻烦

使用MyBatis框架访问数据库

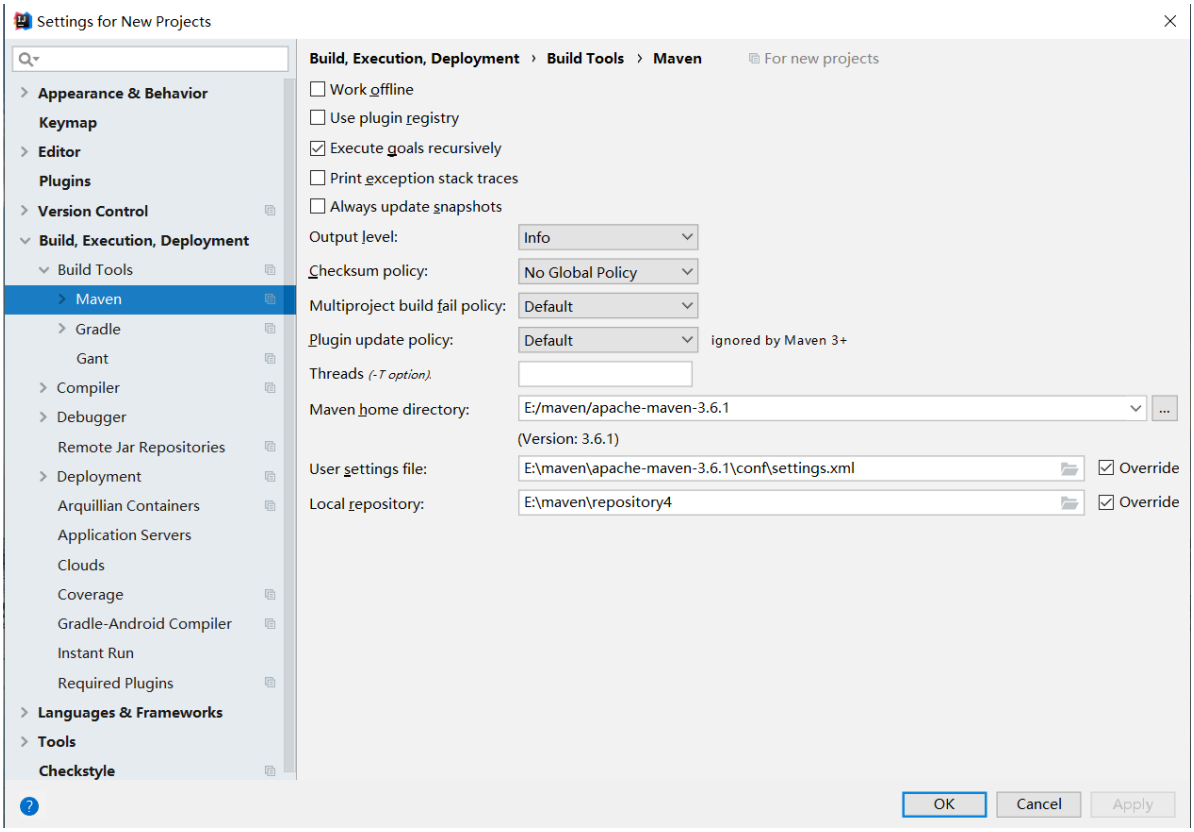
- mybatis框架对JDBC访问数据库的过程进行了封装，简化了JDBC操作
- mybatis自身支持数据连接池（还可以配置其他的数据库连接池），提高效率
- mybatis中的sql语句是在mapper配置文件中，修改sql语句只需要修改配置文件就可以了，不需要重新编译类
- mybatis对查询结果集进行了处理，可以自动将查询结果映射为相应的结果

三、MyBatis快速入门案例

1、Maven简单介绍

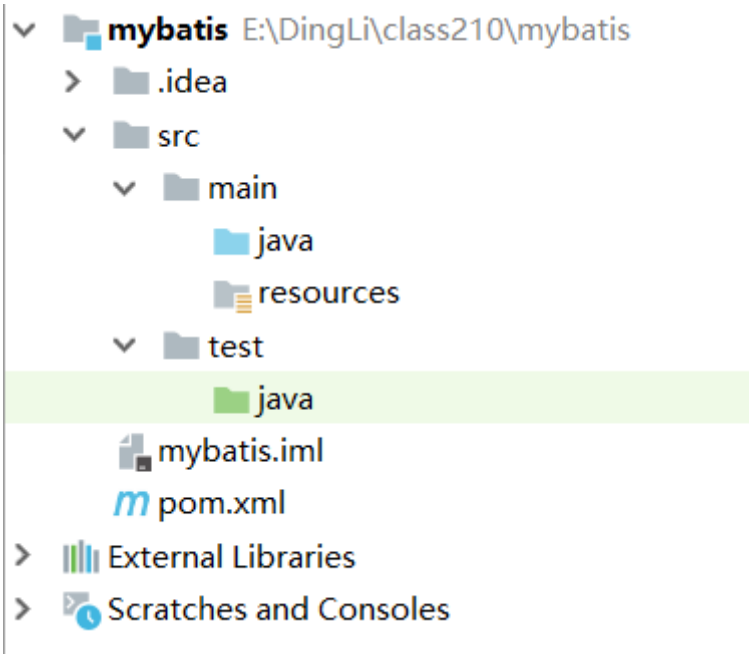


2、Maven在IDEA中的配置



3、创建Maven的简单Java项目

项目结构如下：



4、准备数据

准备数据库和数据库表

username	password	email
chendikai	123456789	chendikai@qq.com
孤独患者	789456	guduhuanzhe@163.com
湖师	hushi	hushi@qq.com
陌上杨花	chendikai	2086@qq.com

5、导入jar包

配置pom文件

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.hbnu</groupId>
    <artifactId>mybatis</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <!-- junit单元测试 -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.9</version>
        </dependency>
        <!-- mysql驱动 -->
        <dependency>
```

```
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.12</version>
        </dependency>
        <!-- mybatis -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.2.8</version>
        </dependency>
        <!-- 整合log4j -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.6.4</version>
        </dependency>

    </dependencies>
</project>
```

6、创建实体类User

```
package com.hbnu.pojo;

/**
 * @author 陈迪凯
 * @date 2021-03-03 9:12
 */
public class User {
    private String username;
    private String password;
    private String email;

    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 getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

```

```
    @Override
    public String toString() {
        return "User{" +
                "username='" + username + '\'' +
                ", password='" + password + '\'' +
                ", email='" + email + '\'' +
                '}';
    }
}
```

7、编写映射配置文件UserMapper.xml

```
<?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="UserMapper">
    <select id="findAll" resultType="com.hbnu.pojo.User">
        select * from tb_user
    </select>

</mapper>
```

8、编写核心配置文件mybatis-config.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>
    <environments default="develop">
        <environment id="develop">
            <transactionManager type="JDBC"></transactionManager>
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.cj.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql:///hbnu?
serverTimezone=GMT&amp;useSSL=false&amp;characterEncoding=utf-8"/>
                <property name="username" value="root"/>
                <property name="password" value="chendikai"/>
            </dataSource>
        </environment>

    </environments>

    <mappers>
        <mapper resource="UserMapper.xml"/>
    </mappers>
</configuration>
```

9、测试

```
package com.hbnu.pojo;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
```

```java
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;

import java.io.IOException;
import java.io.InputStream;
import java.util.List;

/**
 * @author 陈迪凯
 * @date 2021-03-10 8:14
 */
public class MyBatisTest {

    @Test
    public void findAll() throws IOException {
        // 1、通过mybatis-config.xml核心配置文件构建SqlSessionFactory
        InputStream in = Resources.getResourceAsStream("mybatis-config.xml");

        // 2、构建工厂SqlSessionFactory
        SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(in);

        // 3、通过SqlSessionFactory构建sqlSession对象，用于发送sql语句去执行，获取返回结果
        SqlSession sqlSession = sqlSessionFactory.openSession();

        // 4、执行sql语句
        String sqlId = "UserMapper.findAll";
        List<User> userList = sqlSession.selectList(sqlId);

        for (User user : userList) {
            System.out.println(user);
        }

    }
}
```