韩顺平 MyBatis(简化数据库操作的持久层框架) - 2022 版

1 MyBatis 介绍

1.1 官方文档

1.1.1 MyBatis 中文手册 https://mybatis.org/mybatis-3/zh/index.html

https://mybatis.net.cn/

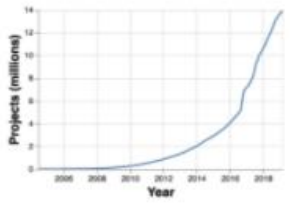
1.1.2 Maven 仓库 https://mvnrepository.com/

老韩提示：需要什么 jar 包，搜索得到对应的 maven dependency

MVNREPOSITORY

Search for groups, artifacts, categories

Indexed Artifacts (21.4M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers

Home » org.mybatis » mybatis

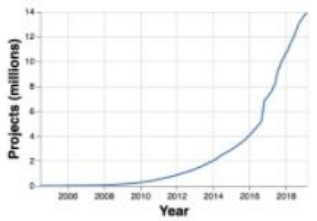
MyBatis

The MyBatis SQL mapper framework makes it easier to use a relational database with objec
objects with stored procedures or SQL statements using a XML descriptor or annotations. S
MyBatis data mapper over object relational mapping tools.

License	Apache 2.0
Categories	Object/Relational Mapping
Tags	persistence relational mapping
Used By	1,079 artifacts

Central (35) | EBIPublic (2) | ICM (1) | Alfresco (1)

Version	Repository
3.5.7	Central
3.5.6	Central
3.5.5	Central



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities

MyBatis » 3.5.7

The MyBatis SQL mapper framework makes it easier to use a relational datab
objects with stored procedures or SQL statements using a XML descriptor or
MyBatis data mapper over object relational mapping tools.

License	Apache 2.0
Categories	Object/Relational Mapping
HomePage	http://www.mybatis.org/mybatis-3
Date	(Apr 25, 2021)
Files	jar (1.7 MB) View All
Repositories	Central
Used By	1,079 artifacts

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen

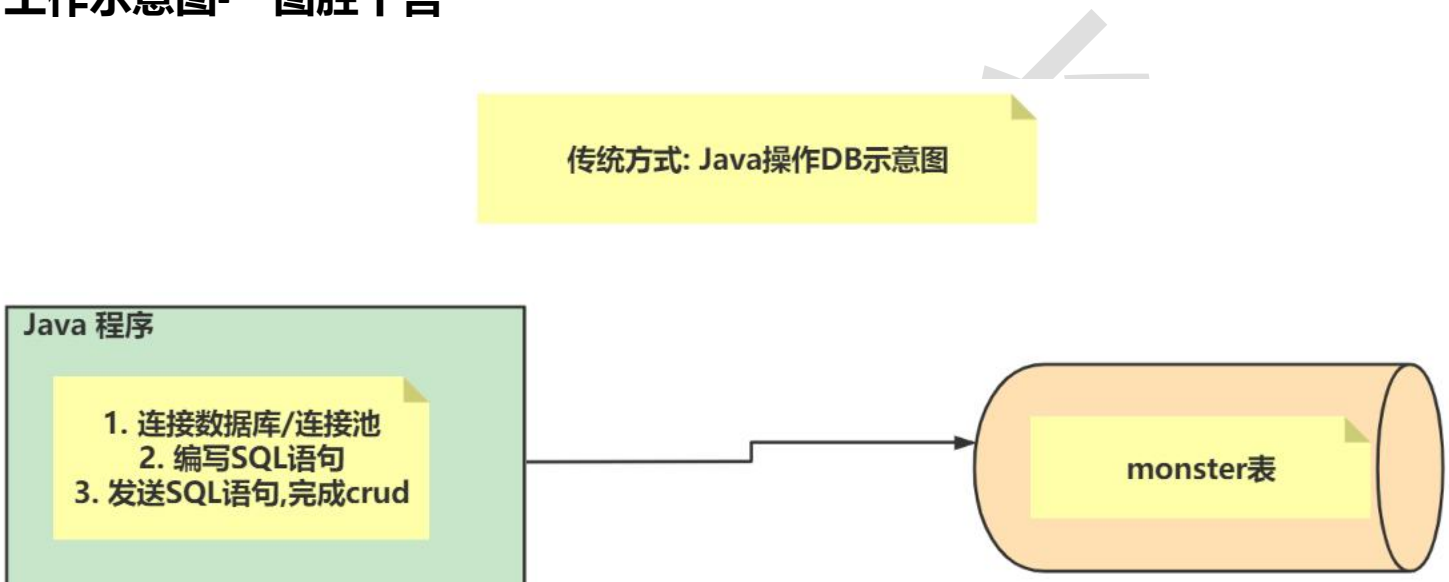
```
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.7</version>
</dependency>
```

1.2 概述

1.2.1 为什么需要 MyBatis

- 传统的 Java 程序操作 DB 分析

1、工作示意图-一图胜千言



老师分析传统方式的问题
1. 要自己去连接数据库， 这段代码是程序员编写(不一定标准)
2. 程序不是OOP的方式来操作DB(也就是说: 我们希望在程序/代码这个层面上，是以对象的形式来操作) insert(object)
3. 因为SQL语句是写在程序中， 属于硬编码，没有解耦

2、传统方式问题分析(如上)

3、引出 MyBatis

1.2.2 基本介绍

1. MyBatis 是一个持久层框架

2. 前身是 ibatis，在 ibatis3.x 时，更名为 MyBatis

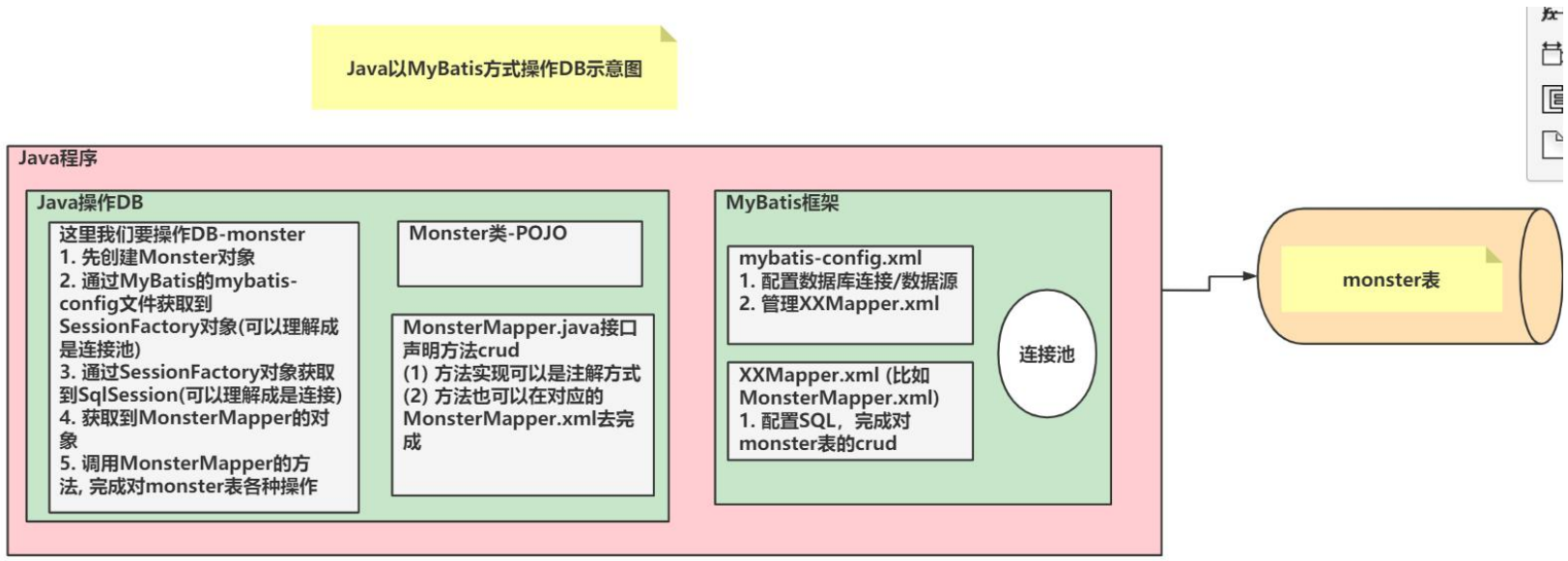
3. MyBatis 在 java 和 sql 之间提供更灵活的映射方案

4. mybatis 可以将对数据表的操作(sql,方法)等等直接剥离，写到 xml 配置文件，实现和 java 代码的解耦

5. mybatis 通过 SQL 操作 DB，建库建表的工作需要程序员完成

1.2.3 MyBatis 工作原理- 一图胜千言

- **MyBatis 工作示意图**



老师分析MyBatis方式的好处
1. 数据库的连接/连接池，只需要配置即可
2. 程序是以OOP的方式来操作DB
3. SQL语句是可以写在xml文件，实现了解耦
4. MyBatis 可以对DB操作进行优化，提高效率，比如配置缓存

2 MyBatis 快速入门

2.1 快速入门需求说明

要求: 开发一个 **MyBatis** 项目，通过 **MyBatis** 的方式可以完成对 **monster** 表的 **crud** 操作

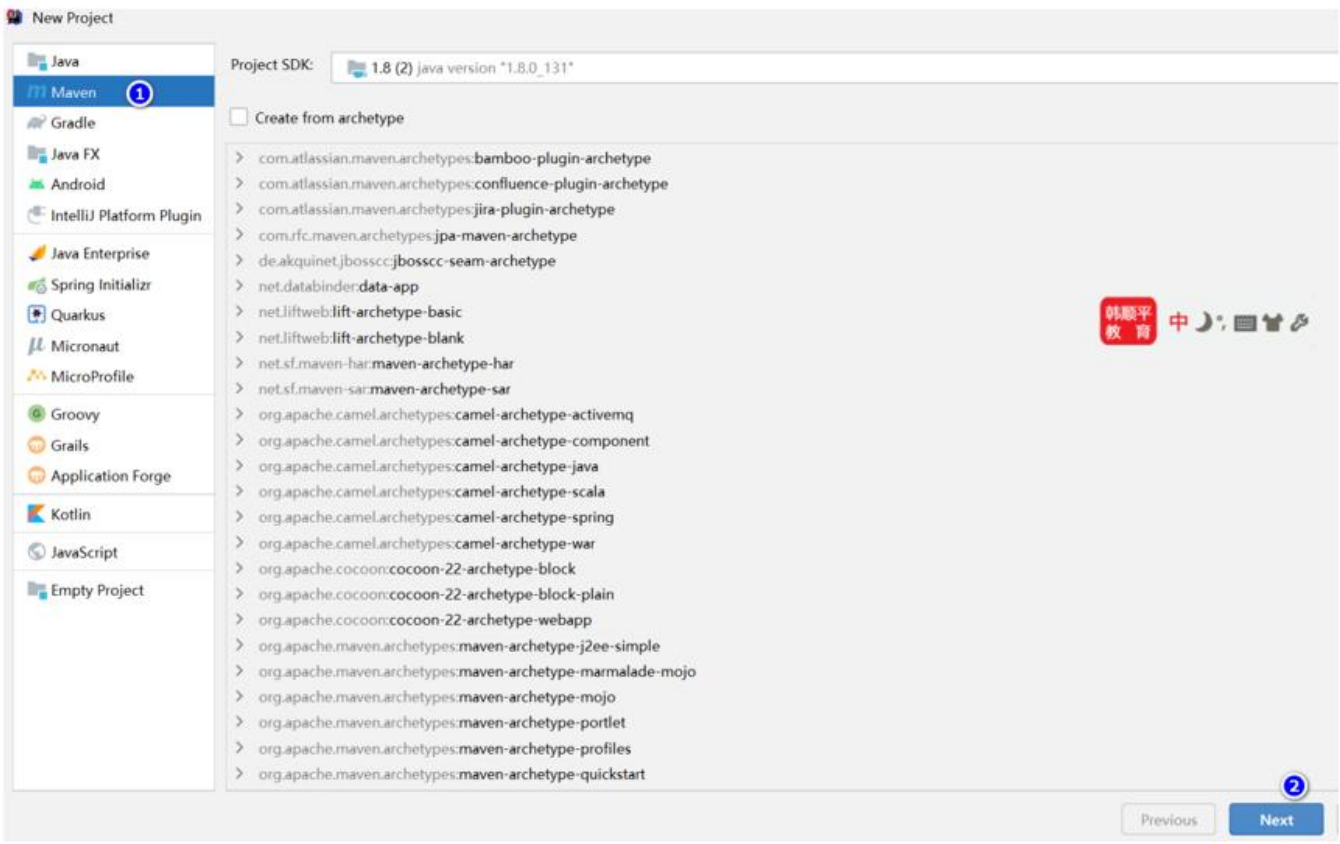
2.2 快速入门-代码实现

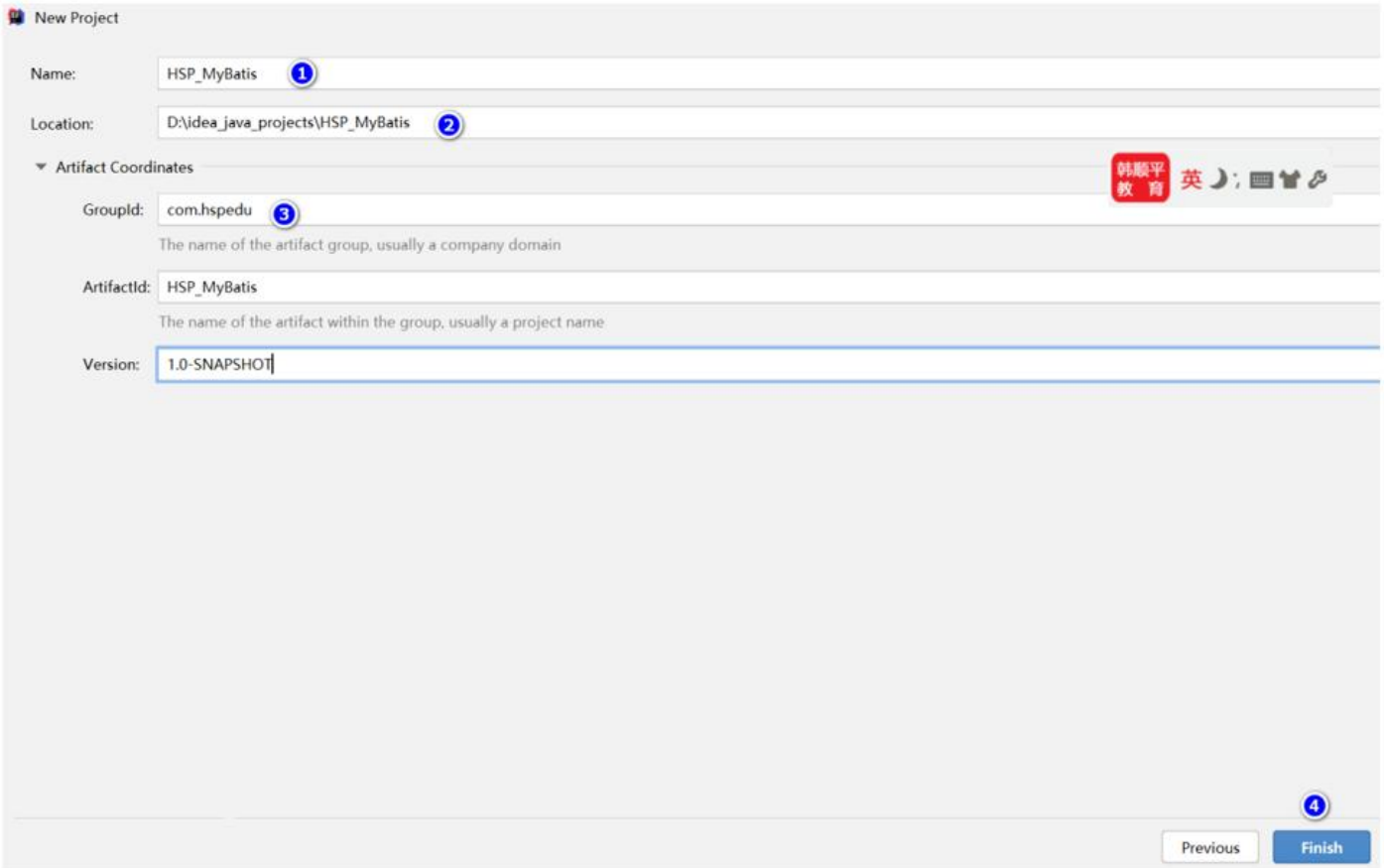
1. 创建 **mybatis** 数据库 **- monster** 表

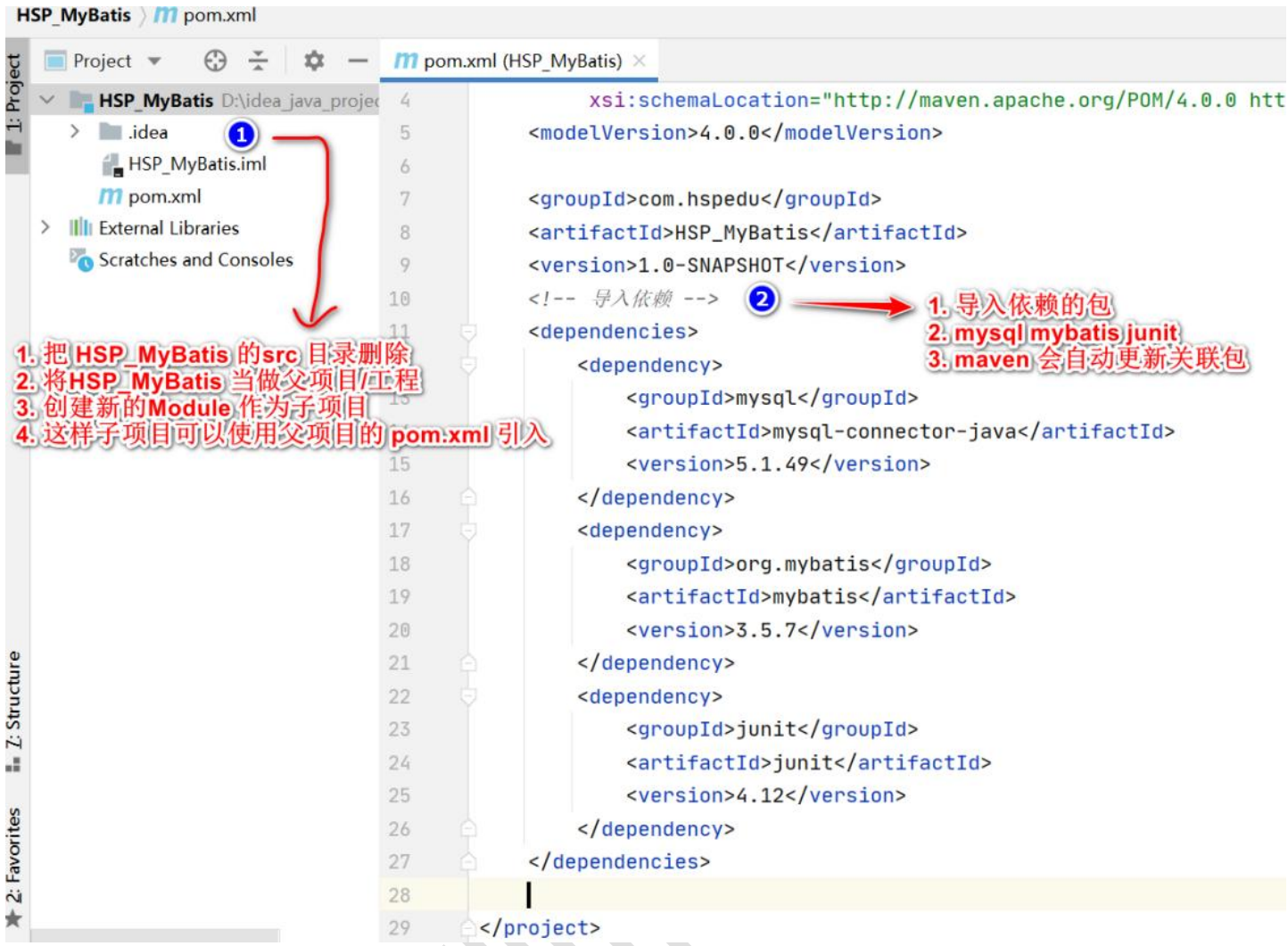
```
CREATE DATABASE `mybatis`

CREATE TABLE `monster` (
```

```
`id` INT NOT NULL AUTO_INCREMENT,

`age` INT NOT NULL,

`birthday` DATE DEFAULT NULL,

`email` VARCHAR(255) NOT NULL ,

`gender` TINYINT NOT NULL,

`name` VARCHAR(255) NOT NULL,

`salary` DOUBLE NOT NULL,

PRIMARY KEY (`id`)

) CHARSET=utf8
```

2. 创建 maven 项目， 方便项目需要 jar 包管理


New Project
Java
Maven
Gradle
Java FX
Android
IntelliJ Platform Plugin
Java Enterprise
Spring Initializr
Quarkus
Micronaut
MicroProfile
Groovy
Grails
Application Forge
Kotlin
JavaScript
Empty Project
Project SDK:
1.8 (2) java version "1.8.0_131"
Create from archetype
com.atlassian.maven.archetypes:bamboo-plugin-archetype
com.atlassian.maven.archetypes:confluence-plugin-archetype
com.atlassian.maven.archetypes:jira-plugin-archetype
com.rfc.maven.archetypes:jpa-maven-archetype
de.akquinet.jbosscc:jbosscc-seam-archetype
net.databinder:data-app
net.liftweb:lift-archetype-basic
net.liftweb:lift-archetype-blank
net.sf.maven-har:maven-archetype-har
net.sf.maven-sar:maven-archetype-sar
org.apache.camel.archetypes:camel-archetype-activemq
org.apache.camel.archetypes:camel-archetype-component
org.apache.camel.archetypes:camel-archetype-java
org.apache.camel.archetypes:camel-archetype-scala
org.apache.camel.archetypes:camel-archetype-spring
org.apache.camel.archetypes:camel-archetype-war
org.apache.cocoon:cocoon-22-archetype-block
org.apache.cocoon:cocoon-22-archetype-block-plain
org.apache.cocoon:cocoon-22-archetype-webapp
org.apache.maven.archetypes:maven-archetype-j2ee-simple
org.apache.maven.archetypes:maven-archetype-marmalade-mojo
org.apache.maven.archetypes:maven-archetype-mojo
org.apache.maven.archetypes:maven-archetype-portlet
org.apache.maven.archetypes:maven-archetype-profiles
org.apache.maven.archetypes:maven-archetype-quickstart
Previous
Next


New Project
Name:
HSP_MyBatis
Location:
D:\idea_java_projects\HSP_MyBatis
Artifact Coordinates
GroupId:
com.hspedu
The name of the artifact group, usually a company domain
ArtifactId:
HSP_MyBatis
The name of the artifact within the group, usually a project name
Version:
1.0-SNAPSHOT
Previous
Finish

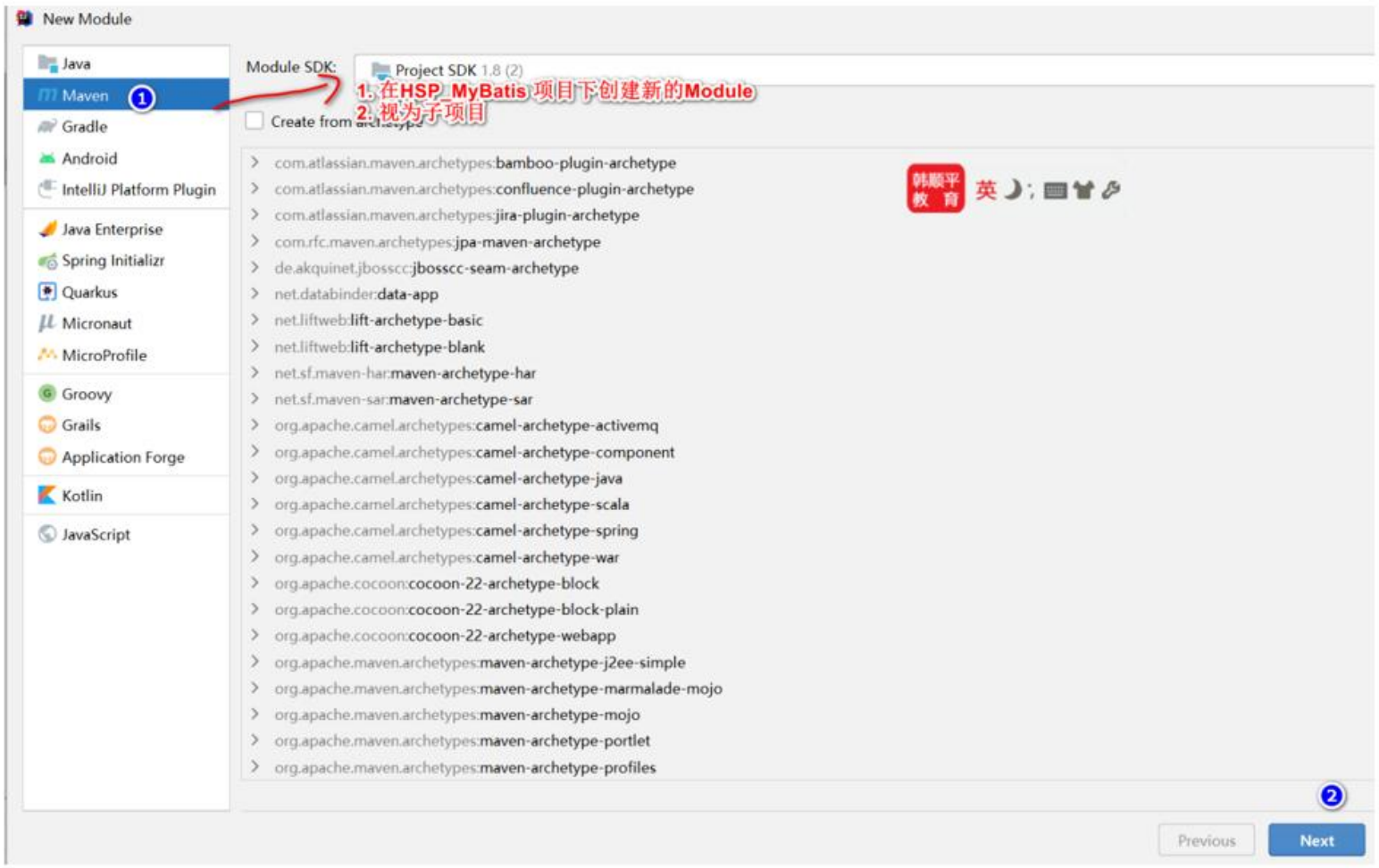


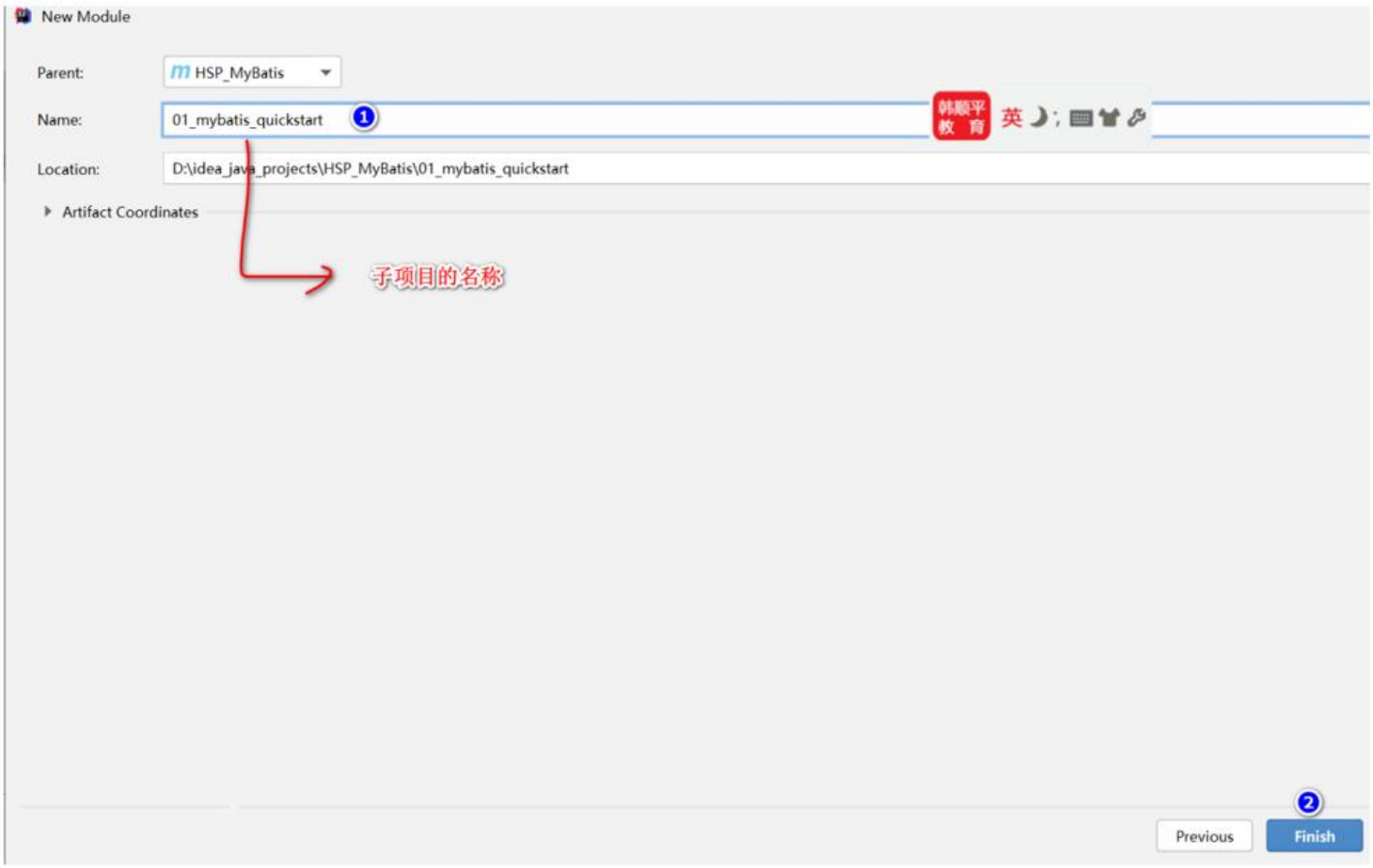
说明：将 D:\idea_java_projects\HSP_MyBatis\pom.xml 作为父项目 pom.xml 引入相关依赖

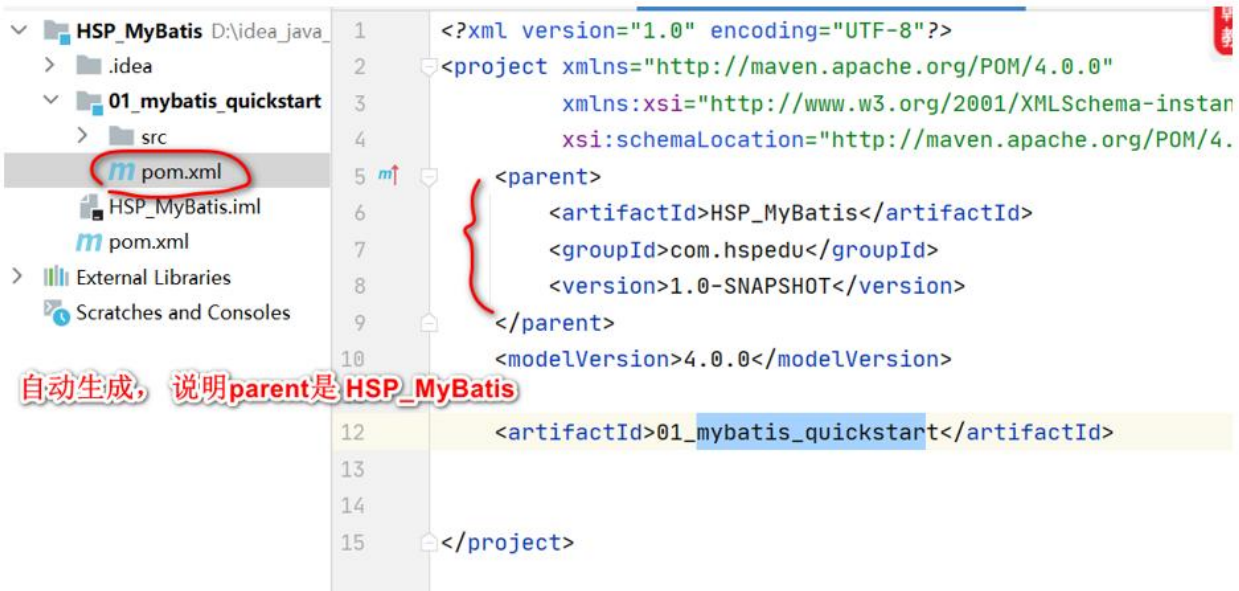
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>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

```
<groupId>com.hspedu</groupId>
<artifactId>HSP_MyBatis</artifactId>
<packaging>pom</packaging>
<version>1.0-SNAPSHOT</version>

<!-- 导入依赖 -->
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.49</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.5.7</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
```

```
        </dependency>
    </dependencies>
</project>
```







3. 创建 resources/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="development">
        <environment id="development">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://127.0.0.1:3306/mybatis?userSSL=true&amp;
                userUnicode=true&amp;characterEncoding=UTF-8"/>
                <property name="username" value="root"/>
                <property name="password" value="hsp"/>
            </dataSource>
        </environment>
    </environments>
<!--    <mappers>-->
<!--        <mapper resource="org/mybatis/example/BlogMapper.xml"/>-->
<!--    </mappers>-->
</configuration>
```

```
<?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="development">
        <environment id="development">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver"/>
                <property                                          name="url"
```

```
value="jdbc:mysql://127.0.0.1:3306/mybatis?useSSL=true&amp;
                    useUnicode=true&amp;characterEncoding=UTF-8"/>
                    <property name="username" value="root"/>
                    <property name="password" value="hsp"/>
                </dataSource>
            </environment>
        </environments>
</configuration>
```

4. 创建 D:\idea_java_projects\HSP_MyBatis\01_mybatis_quickstart\src\main\java\com\hspedu\entity\Monster.java

```
package com.hspedu.entity;

import java.util.Date;

/**
 * @author 韩顺平
 * @version 1.0
 */
//老韩解读
//1. 一个普通的 Pojo 类
```

```
//2. 使用原生态的 sql 语句查询结果还是要封装成对象
//3. 要求大家这里的实体类属性名和表名字段保持一致。
public class Monster {

    private Integer id;
    private Integer age;
    private String name;
    private String email;
    private Date birthday;
    private double salary;
    private Integer gender;
    public Monster() {
    }

    public Monster(Integer id, Integer age, String name, String email, Date birthday, double
salary, Integer gender) {
        this.id = id;
        this.age = age;
        this.name = name;
        this.email = email;
        this.birthday = birthday;
        this.salary = salary;
```

```
        this.gender = gender;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public Integer getAge() {
        return age;
    }

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

    public String getName() {
        return name;
    }
```

```
public void setName(String name) {
        this.name = name;
}

public String getEmail() {
        return email;
}

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

public Date getBirthday() {
        return birthday;
}

public void setBirthday(Date birthday) {
        this.birthday = birthday;
}

public double getSalary() {
```

```
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public Integer getGender() {
        return gender;
    }

    public void setGender(Integer gender) {
        this.gender = gender;
    }

    @Override
    public String toString() {
        return "Monster{" +
                "id=" + id +
                ", age=" + age +
                ", name='" + name + '\'' +
                ", email='" + email + '\'' +
```

```
                ", birthday=" + birthday +
                ", salary=" + salary +
                ", gender=" + gender +
                '}';
    }
}
```

5. 创建 D:\idea_java_projects\HSP_MyBatis\01_mybatis_quickstart\src\main\java\com\hspedu\mapper\MonsterMapper.java

```
package com.hspedu.mapper;

import com.hspedu.entity.Monster;
import org.apache.ibatis.annotations.Param;

import java.util.List;
import java.util.Map;

/**
 * @author 韩顺平
 * @version 1.0
 */
```

```
public interface MonsterMapper {
    //添加方法
    public void addMonster(Monster monster);
}
```

6. 创建 D:\idea_java_projects\HSP_MyBatis\01_mybatis_quickstart\src\main\java\com\hspedu\mapper\MonsterMapper.xml

```
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<!-- mapper: 表示一个映射器，文档:https://mybatis.org/mybatis-3/zh/sqlmap-xml.html#insert_update_and_delete
    1. namespace="com.hspedu.mapper.MonsterMapper"
     说明本 mapper.xml 文件是用来 映射管理 MonsterMapper 接口
     ,主要是去实现 MonsterMapper 接口声明方法
    2. select: 实现一个查询操作  insert:表示一个添加操作
    3. id="addMonster" 表示 MonsterMapper 接口 的方法名
    4. resultType="xx" 返回的结果类型，如果没有就不需要写
    5. parameterType="com.hspedu.entity.Monster" 表示该方法输入的参数类型
    6. (age,birthday,email,gender,name,salary) 表的字段名
    7. #{age},#{birthday},#{email},#{gender},#{name},#{salary} 是实体类 Monster 的属性名
```

```
    -->
<mapper namespace="com.hspedu.mapper.MonsterMapper">
<!--
    没有在 mybatis-config.xml 指定 typealiases 时，需要给 Monster 指定全类名
    <insert id="addMonster" parameterType="com.hspedu.entity.Monster"-->
<!--                    useGeneratedKeys="true" keyProperty="id"-->
<!--        >-->

    <insert id="addMonster" parameterType="com.hspedu.entity.Monster"
        useGeneratedKeys="true" keyProperty="id"
    >
        INSERT INTO monster (age,birthday,email,gender,name,salary)
        VALUES(#{age},#{birthday},#{email},#{gender},#{name},#{salary})
    </insert>


</mapper>
```

7. 修 改 D:\idea_java_projects\HSP_MyBatis\01_mybatis_quickstart\src\main\resources\mybatis-config.xml

```
</environments>
```

```
<mappers>
    <!-- 这里会引入(注册)我们的 Mapper.xml 文件 -->
    <mapper resource="com/hspedu/mapper/MonsterMapper.xml"/>
</mappers>
```

8. 创建工具类

D:\idea_java_projects\HSP_MyBatis\01_mybatis_quickstart\src\main\java\com\hspedu\util\MyBatisUtils.java

```
package com.hspedu.util;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.IOException;
import java.io.InputStream;

/**
 * @author 韩顺平
 * @version 1.0
```

```
 */
public class MyBatisUtils {
    private static SqlSessionFactory sqlSessionFactory;
    static {
        try {
            //使用 mybatis 第一步：获取 sqlSessionFactory 对象
            String resource = "mybatis-config.xml";
            InputStream inputStream = Resources.getResourceAsStream(resource);
            sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
/**
 * 老韩解读
 * 1. 获得 SqlSession 的实例
 * 2. SqlSession 提供了对数据库执行 SQL 命令所需的所有方法。
 * 3. 通过 SqlSession 实例来直接执行已映射的 SQL 语句
 * @return
 */
public static SqlSession getSqlSession() {
    return sqlSessionFactory.openSession();
```

```
    }
}
```

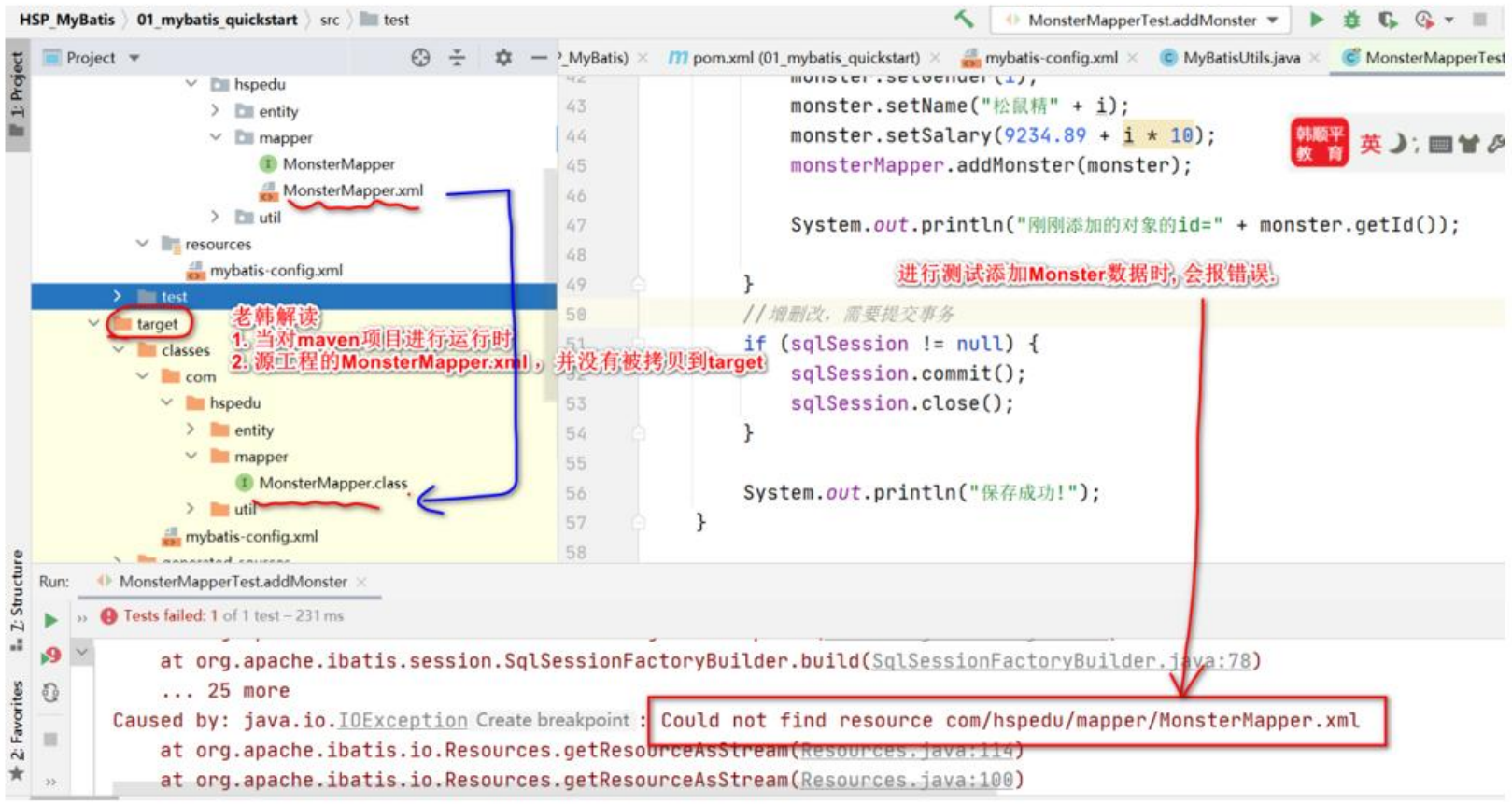
9. 创建 D:\idea_java_projects\HSP_MyBatis\01_mybatis_quickstart\src\test\java\com\hspedu\mapper\MonsterMapperTest.java

```
package com.hspedu.mapper;

import com.hspedu.entity.Monster;
import com.hspedu.util.MyBatisUtils;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Before;
import org.junit.Test;

import java.io.InputStream;
import java.util.*;

/**
 * @author 韩顺平
```

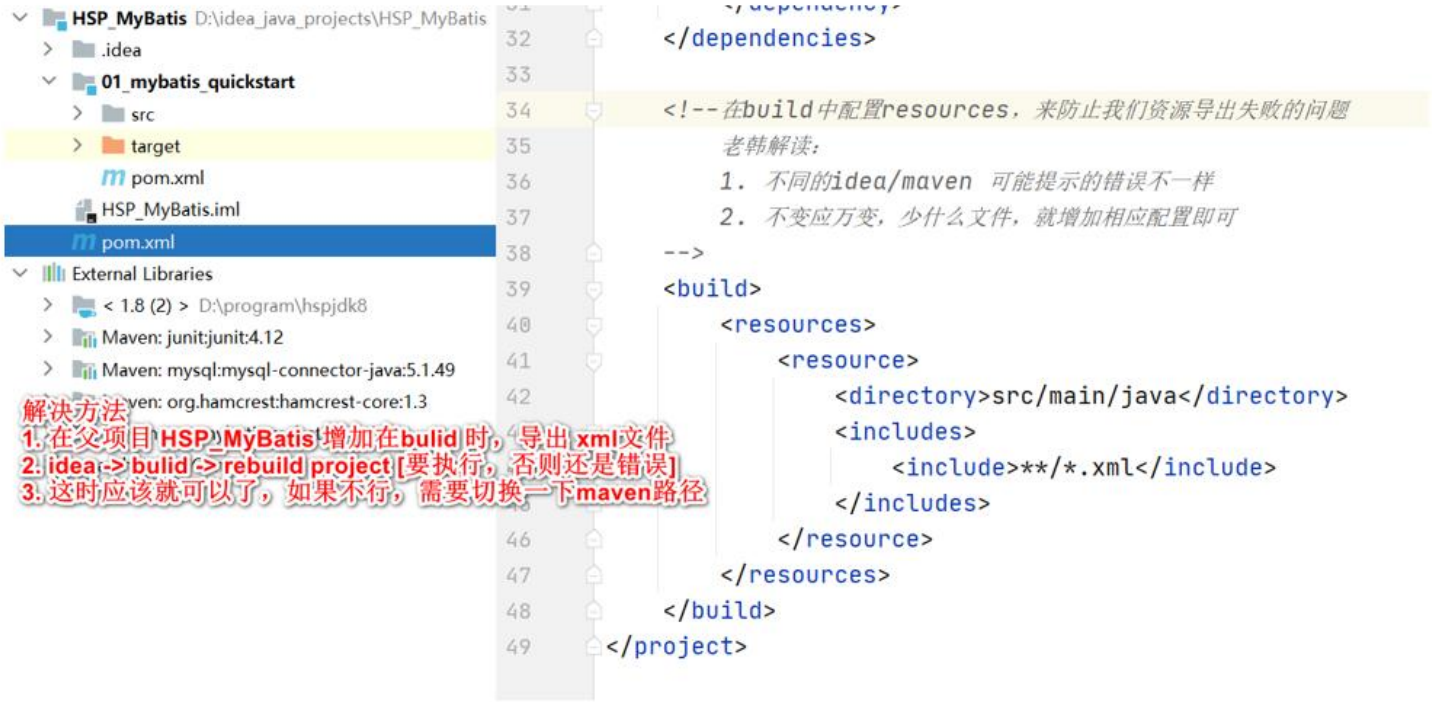
```
 * @version 1.0
 */
public class MonsterMapperTest {
    //这个是 Sql 会话,通过它可以发出 sql 语句
    private SqlSession sqlSession;
    private MonsterMapper monsterMapper;


    @Before
    public void init() throws Exception {
        //通过   SqlSessionFactory 对象获取一个 SqlSession 会话
        sqlSession = MyBatisUtils.getSqlSession();
        //获取 MonsterMapper 接口对象，该对象实现了 MonsterMapper
        monsterMapper = sqlSession.getMapper(MonsterMapper.class);
        System.out.println(monsterMapper.getClass());
}

@Test
public void addMonster() {
    for (int i = 0; i < 1; i++) {
        Monster monster = new Monster();
        monster.setAge(100 + i); monster.setBirthday(new Date());
```

```
            monster.setEmail("tn@sohu.com");
            monster.setGender(1); monster.setName("松鼠精" + i);
            monster.setSalary(9234.89 + i * 10);
            monsterMapper.addMonster(monster);
            System.out.println("刚刚添加的对象的 id=" + monster.getId());
        }
        //增删改，需要提交事务
        if (sqlSession != null) {
            sqlSession.commit();
            sqlSession.close();
        }
            System.out.println("保存成功!");
        }
    }
```

10. 测试，看看是否可以添加成功，这时会出现找不到 Xxxxmapper.xml 错误，分析了原因



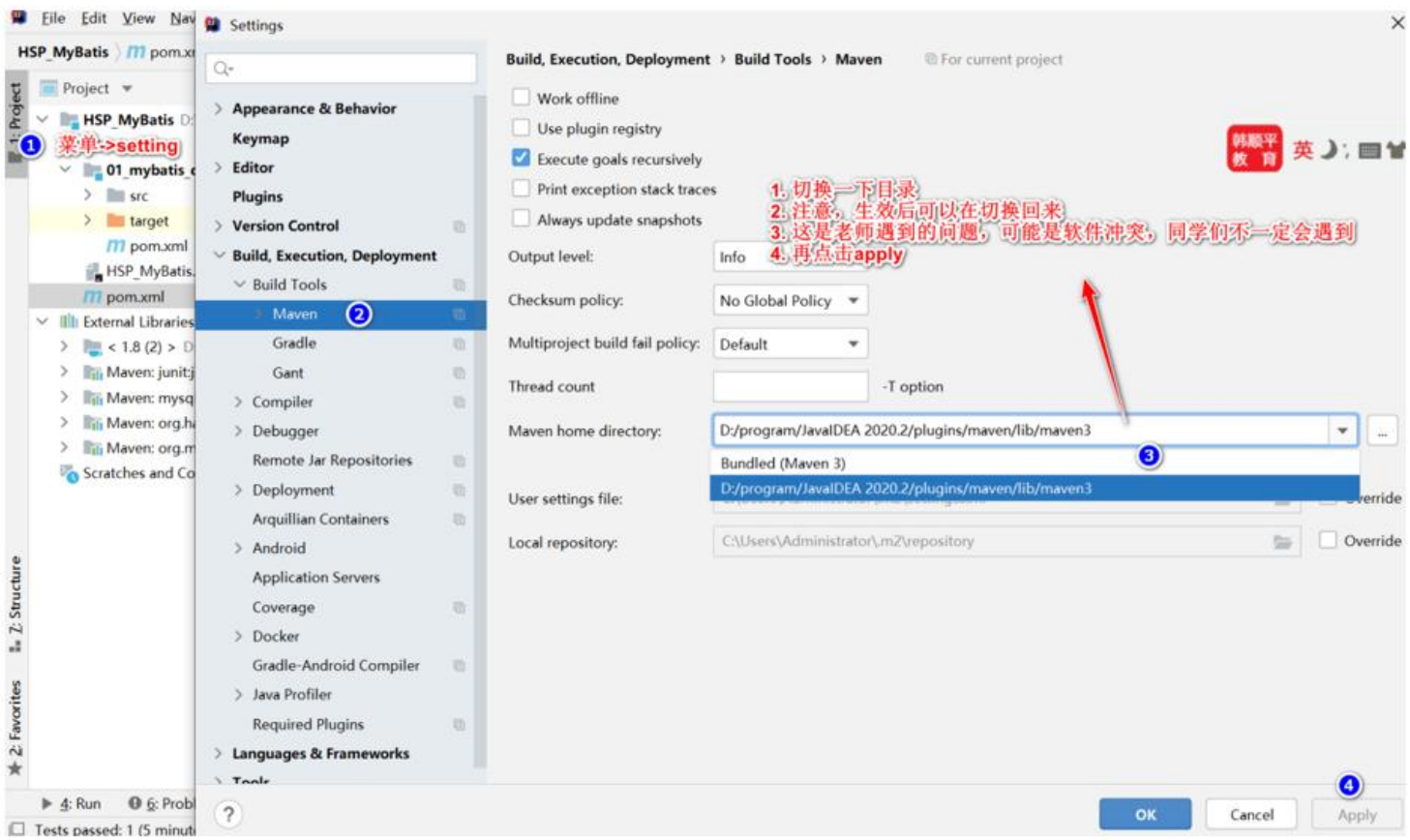
11. 解决找不到 Mapper.xml 配置文件问题，老韩提示：如果顺利，你会很快解决，不顺利，你会干到怀疑人生

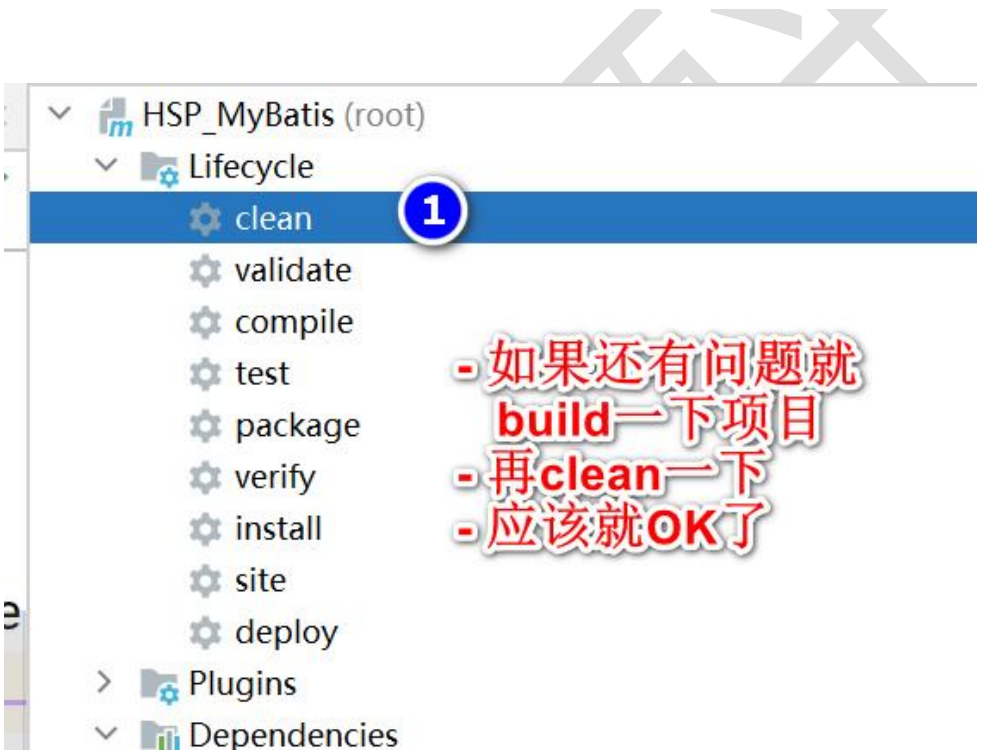


//在父工程的 pom.xml 加入 build 配置

```
</dependencies>
<!--在 build 中配置 resources，来防止我们资源导出失败的问题
      老韩解读：
      1. 不同的 idea/maven 可能提示的错误不一样
      2. 不变应万变，少什么文件，就增加相应配置即可
-->
<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>
```


菜单->setting
Settings
Build, Execution, Deployment › Build Tools › Maven
For current project
Work offline
Use plugin registry
Execute goals recursively
Print exception stack traces
Always update snapshots
1. 切换一下目录
2. 注意，生效后可以在切换回来
3. 这是老师遇到的问题，可能是软件冲突，同学们不一定会遇到
4. 再点击apply
Output level:
Info
Checksum policy:
No Global Policy
Multiproject build fail policy:
Default
Thread count
-T option
Maven home directory:
D:/program/JavaIDEA 2020.2/plugins/maven/lib/maven3
Bundled (Maven 3)
D:/program/JavaIDEA 2020.2/plugins/maven/lib/maven3
User settings file:
Local repository:
C:\Users\Administrator\.m2\repository
Override
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Build Tools
Maven
Gradle
Gant
Compiler
Debugger
Remote Jar Repositories
Deployment
Arquillian Containers
Android
Application Servers
Coverage
Docker
Gradle-Android Compiler
Java Profiler
Required Plugins
Languages & Frameworks
OK
Cancel
Apply


HSP_MyBatis (root)
Lifecycle
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
Dependencies
- 如果还有问题就
build一下项目
- 再clean一下
- 应该就OK了

```
[...\...am\...\...8\bin\j
class com.sun.proxy.$Pro
Sun Jul 18 15:13:34 CST
刚刚添加的对象的id=6
保存成功!
```

执行成功，可以查看mysql

1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

(只读)

id	age	name	salary
1	100	松鼠精0	9234.89
2	100	松鼠精0	9234.89
3	100	松鼠精0	9234.89
4	100	松鼠精0	9234.89
5	100	松鼠精0	9234.89
6	100	松鼠精0	9234.89

12. 修改 MonsterMapper.java，增加方法接口

```
//根据 id 删除一个 Monster
public void delMonster(Integer id);
```

13. 修改 MonsterMapper.xml，实现方法接口

```
<!-- 实现 delMonster 方法
   注意 parameterType="Integer" 如果是 Java 本身的数据类型直接写类名即可
-->
<delete id="delMonster" parameterType="Integer">
   DELETE FROM monster
   WHERE id=#{id}
</delete>
```

14. 完成测试，修改 D:\idea_java_projects\HSP_MyBatis\01_mybatis_quickstart\src\test\java\com\hspedu\mapper\MonsterMapperTest.java，增加测试方法

```
//测试 mybatis 删除功能
@Test
public void delMonster() {

    monsterMapper.delMonster(2);
    if (sqlSession != null) {
        sqlSession.commit();
        sqlSession.close();
    }
    System.out.println("删除 ok");
}
```

15. 修改 MonsterMapper.java，增加方法接口

```
//修改 Monster
public void updateMonster(Monster monster);

//查询-根据 id
public Monster getMonsterById(Integer id);
```

```
//查询所有的 Monster
public List<Monster> findAllMonster();
```

16. 修改 MonsterMapper.xml，实现方法接口

1) 为了配置方便，在 mybatis-config.xml 配置 Monster 的别名

```
<configuration>
    <typeAliases>
        <!-- 为某个 mapper 指定一个别名，下面可以在 XxxxxMapper.xml 做相应简化处理
-->
        <typeAlias type="com.hspedu.entity.Monster" alias="Monster"/>
    </typeAliases>
```

2) 修改 MonsterMapper.xml，实现方法接口，可以使用 Monster 别名了.

```
<!-- 实现 updateMonster 方法 -->
<update id="updateMonster" parameterType="Monster">
  UPDATE monster SET age=#{age}, birthday=#{birthday}, email = #{email},
  gender= #{gender}, name=#{name}, salary=#{salary} WHERE id=#{id}
</update>
```

```
<!-- 实现 getMonsterById 方法 -->
<select id="getMonsterById" parameterType="Integer"
    resultType="Monster">
  SELECT * FROM monster WHERE id = #{id}
</select>
<!-- 实现 findAllMonster -->
<select id="findAllMonster"
    resultType="Monster">
  SELECT * FROM monster
</select>
```

17. 完成测试，修改 MonsterMapperTest.java ，增加测试方法

```
//测试 mybatis 的修改
@Test
public void updateMonster() {
        Monster monster = new Monster();
        monster.setAge(200);
        monster.setBirthday(new Date());
        monster.setEmail("hspedu@sohu.com");
        monster.setGender(2);
        monster.setName("狐狸精");
        monster.setSalary(9234.89);
```

```
        monster.setId(2);
        monsterMapper.updateMonster(monster);
        if (sqlSession != null) {
            sqlSession.commit();
            sqlSession.close();
        }
        System.out.println("修改 ok");
    }

    //测查询单个对象
    @Test
    public void getMonsterById() {
        Monster monster = monsterMapper.getMonsterById(2);
        System.out.println(monster);
        if (sqlSession != null) {
            sqlSession.close();
        }
    }

    @Test
    public void findAllMonster() {
        List<Monster> monsterList = monsterMapper.findAllMonster();
```

```
        for (Monster monster : monsterList) {
            System.out.println(monster);
        }
        if (sqlSession != null) {
            sqlSession.close();
        }
    }
```

2.3 日志输出-查看 SQL

2.3.1 看一个需求

1. 在开发 MyBatis 程序时，比如执行测试方法，程序员往往需要查看 程序底层发给 MySQL 的 SQL 语句，到底长什么样，怎么办?

2. 解决方案: 日志输出

2.3.2 日志文档 https://mybatis.org/mybatis-3/zh/logging.html

2.3.3 配置日志 https://mybatis.org/mybatis-3/zh/configuration.html#settings

2.3.4 配置日志-具体操作

1. 查看文档 ： https://mybatis.org/mybatis-3/zh/configuration.html#settings

logPrefix	指定 MyBatis 增加到日志名称的前缀。	任何字符串
logImpl	指定 MyBatis 所用日志的具体实现，未指定时将自动查找。	SLF4J | LOG4J | LOG4J2 | JDK_LOGGING | COMMONS_LOGGING | STDOUT_LOGGING | NO_LOGGING

2. 修改 D:\java_projects\HSP_MyBatis\01_mybatis_quickstart\src\main\resources\mybatis-config.xml，加入日志输出配置，方便分析 SQL 语句

```
<configuration>


    <!-- 配置 MyBatis 自带的日志输出，还可以是其它日志比如 SLF4J | LOG4J | LOG4J2 | JDK_LOGGING 等 -->
    <settings>
        <setting name="logImpl" value="STDOUT_LOGGING"/>
    </settings>
```

//其它....

```
Opening JDBC Connection
Thu Nov 04 01:54:15 CST 2021 WARN: Establishing SSL connection without server's identity verifi
Created connection 1027007693. ①
Setting autocommit to false on JDBC Connection [com.mysql.jdbc.JDBC4Connection@3d36e4cd]
==>  Preparing: UPDATE monster SET age = ? WHERE id = ? ②
==> Parameters: 10(Integer), 4(Integer)
<==    Updates: 1
Committing JDBC Connection [com.mysql.jdbc.JDBC4Connection@3d36e4cd] ③
Resetting autocommit to true on JDBC Connection [com.mysql.jdbc.JDBC4Connection@3d36e4cd]
Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@3d36e4cd] ④
Returned connection 1027007693 to pool.⑤
操作成功~
```

2.4 课后练习

- **MyBatis 入门练习创建一个 Monk 类 - 自己创建个新的项目完成.**

要求属性

0.id 号

1.法号(nickname)

2.本领(skill)

3.级别(grade)

4.薪水(salary)

5.出生日期(要求可以保存 年-月-日 时:分:秒)

完成功能:

1.可以添加一个和尚

2. 可以修改和尚信息

3. 可以删除和尚

4. 可以取出一个和尚

5. 取出所有和尚

6. 自己完成，会有心得体会

6.入寺庙时间(entry) [只保留年-月-日]	

【提示 保留 年月日 时分秒 只需将 表字段类型 设置为 datetime 即可】

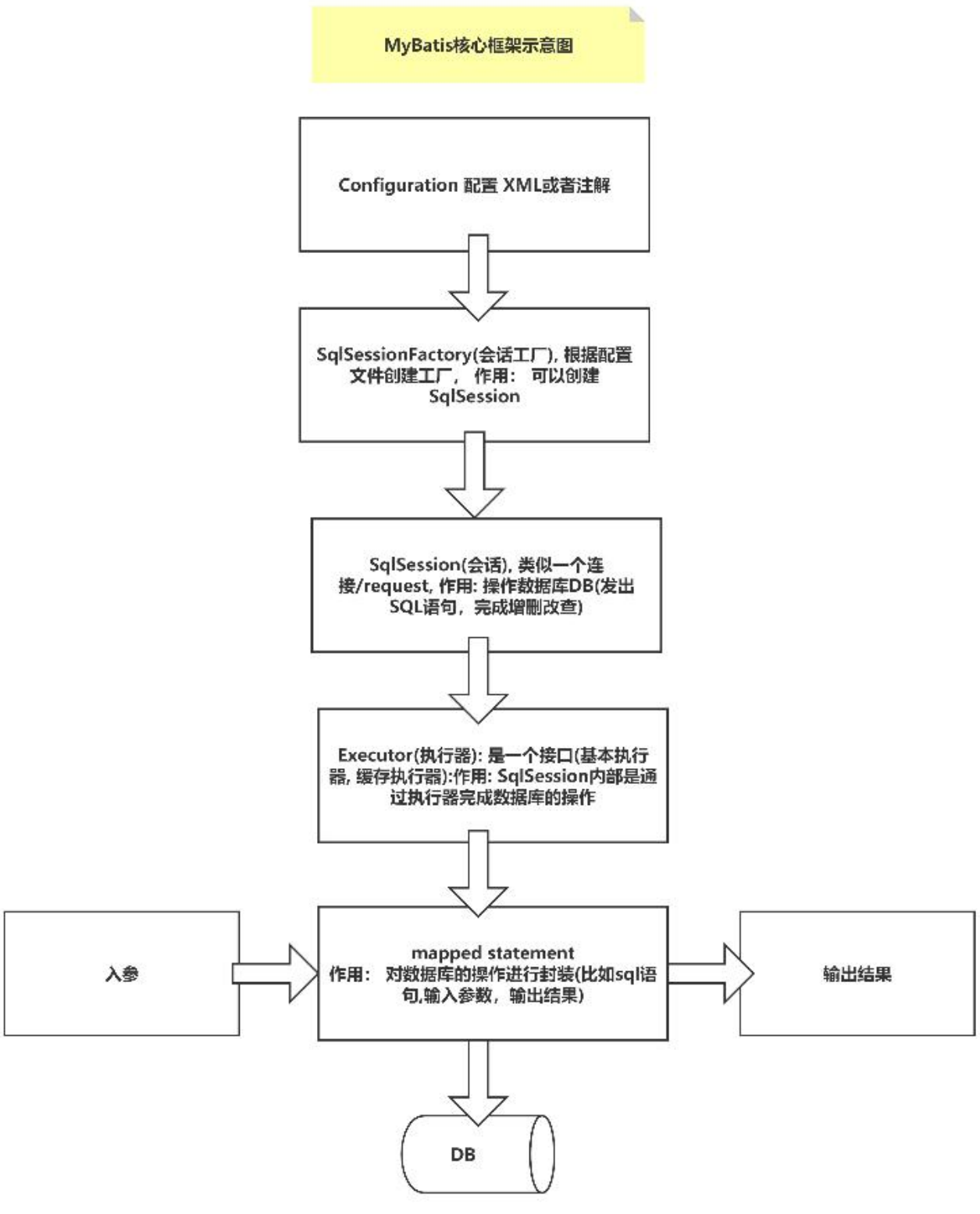
- **说明 比较简单，随堂编写即可**

3 韩顺平 自己实现 MyBatis 底层机制【老韩新增】【封装 Sqlsession 到执行器 + Mapper 接口和 Mapper.xml + MapperBean + 动态代理代理 Mapper 的方法】

3.1 MyBatis 整体架构分析

3.1.1 一图胜千言

1、Mybatis 核心框架示意图


MyBatis核心框架示意图
Configuration 配置 XML或者注解
SqlSessionFactory(会话工厂), 根据配置文件创建工厂， 作用： 可以创建SqlSession
SqlSession(会话), 类似一个连接/request, 作用: 操作数据库DB(发出SQL语句，完成增删改查)
Executor(执行器): 是一个接口(基本执行器, 缓存执行器):作用: SqlSession内部是通过执行器完成数据库的操作
入参
mapped statement
作用： 对数据库的操作进行封装(比如sql语句,输入参数，输出结果)
输出结果
DB

2、对上图的解读

1) mybatis 的核心配置文件

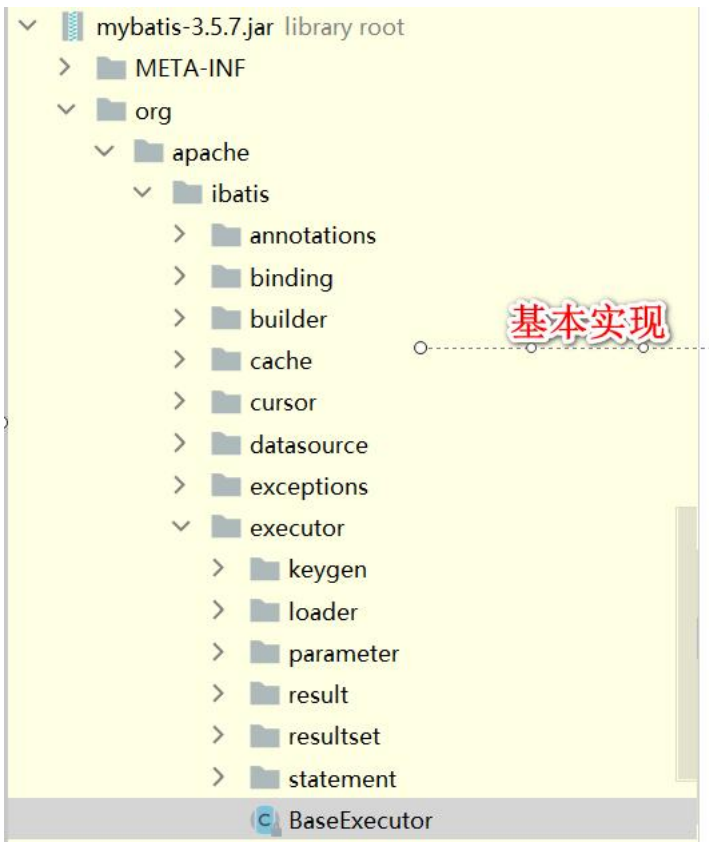
mybatis-config.xml: 进行全局配置，全局只能有一个这样的配置文件

XxxMapper.xml 配置多个 SQL，可以有多个 XxxMappe.xml 配置文件

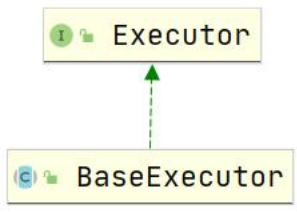
2) 通过 mybatis-config.xml 配置文件得到 SqlSessionFactory

3) 通过 SqlSessionFactory 得到 SqlSession，用 SqlSession 就可以操作数据了

4) SqlSession 底层是 Executor(执行器)，有 2 重要的实现类，有很多方法



基本实现

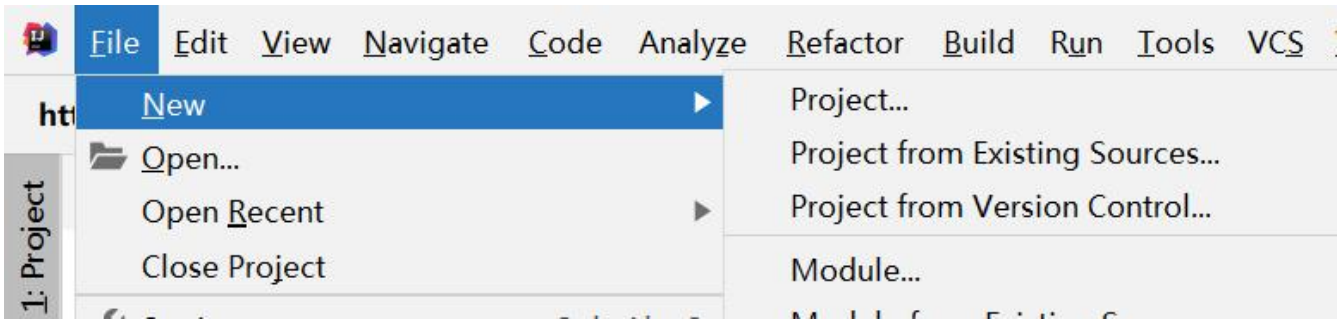


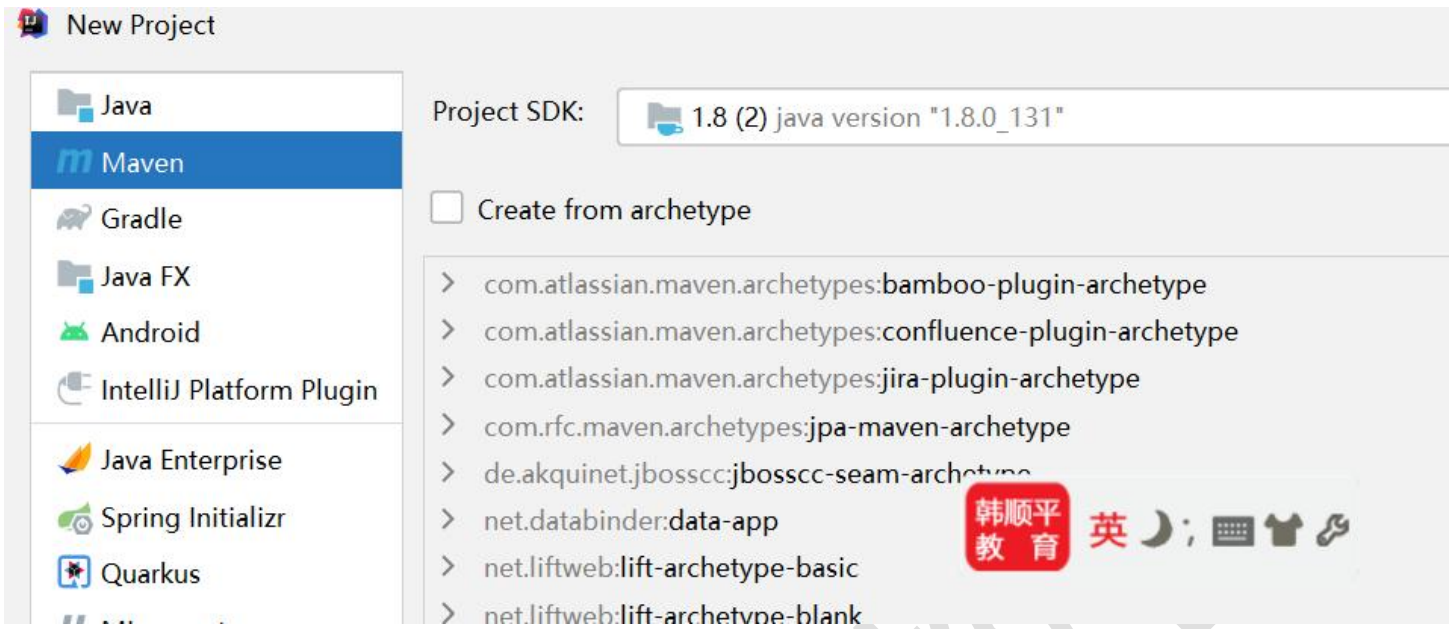
5) MappedStatement 是通过 XxxMapper.xml 中定义，生成的 statement 对象

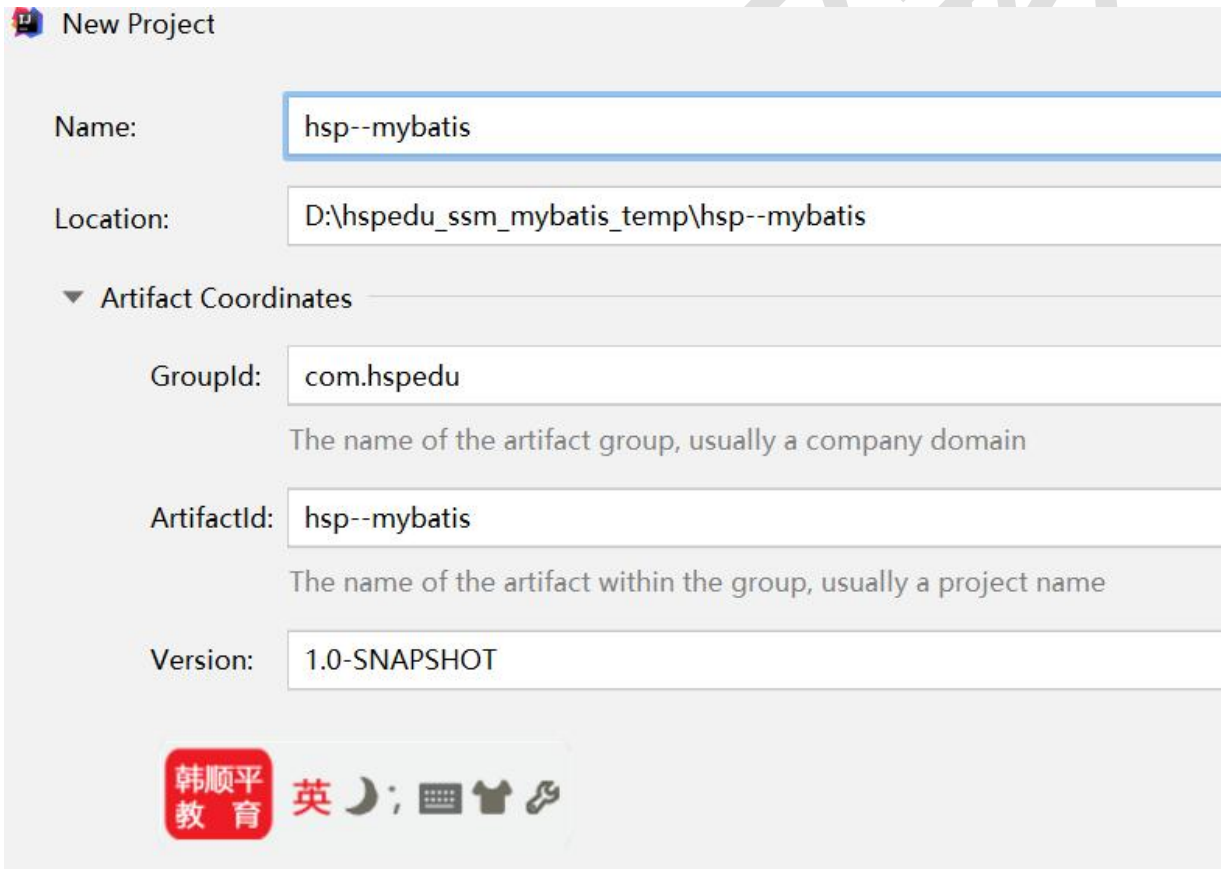
6) 参数输入执行并输出结果集，无需手动判断参数类型和参数下标位置，且自动将结果集映射为 Java 对象

3.2 搭建 MyBatis 底层机制开发环境

1、创建 Maven 项目 hsp-mybatis




New Project
Java
Maven
Gradle
Java FX
Android
IntelliJ Platform Plugin
Java Enterprise
Spring Initializr
Quarkus
Project SDK:
1.8 (2) java version "1.8.0_131"
Create from archetype
com.atlassian.maven.archetypes:bamboo-plugin-archetype
com.atlassian.maven.archetypes:confluence-plugin-archetype
com.atlassian.maven.archetypes:jira-plugin-archetype
com.rfc.maven.archetypes:jpa-maven-archetype
net.databinder:data-app
net.liftweb:lift-archetype-basic
net.liftweb:lift-archetype-blank


New Project
Name:
hsp--mybatis
Location:
D:\hspedu_ssm_mybatis_temp\hsp--mybatis
Artifact Coordinates
GroupId:
com.hspedu
The name of the artifact group, usually a company domain
ArtifactId:
hsp--mybatis
The name of the artifact within the group, usually a project name
Version:
1.0-SNAPSHOT

2、修改 D:\hspedu_ssm_mybatis_temp\hsp-mybatis\pom.xml

```
<?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.hspedu</groupId>
    <artifactId>hsp-mybatis</artifactId>
    <version>1.0-SNAPSHOT</version>

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <java.version>1.8</java.version>
</properties>

<dependencies>
    <!-- 读取 xml 文件 -->
    <dependency>
```

```
        <groupId>dom4j</groupId>
        <artifactId>dom4j</artifactId>
        <version>1.6.1</version>
    </dependency>

    <!-- MySQL -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.49</version>
    </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.20</version>
        </dependency>
    </dependencies>

</project>
```

3、创建数据库和表

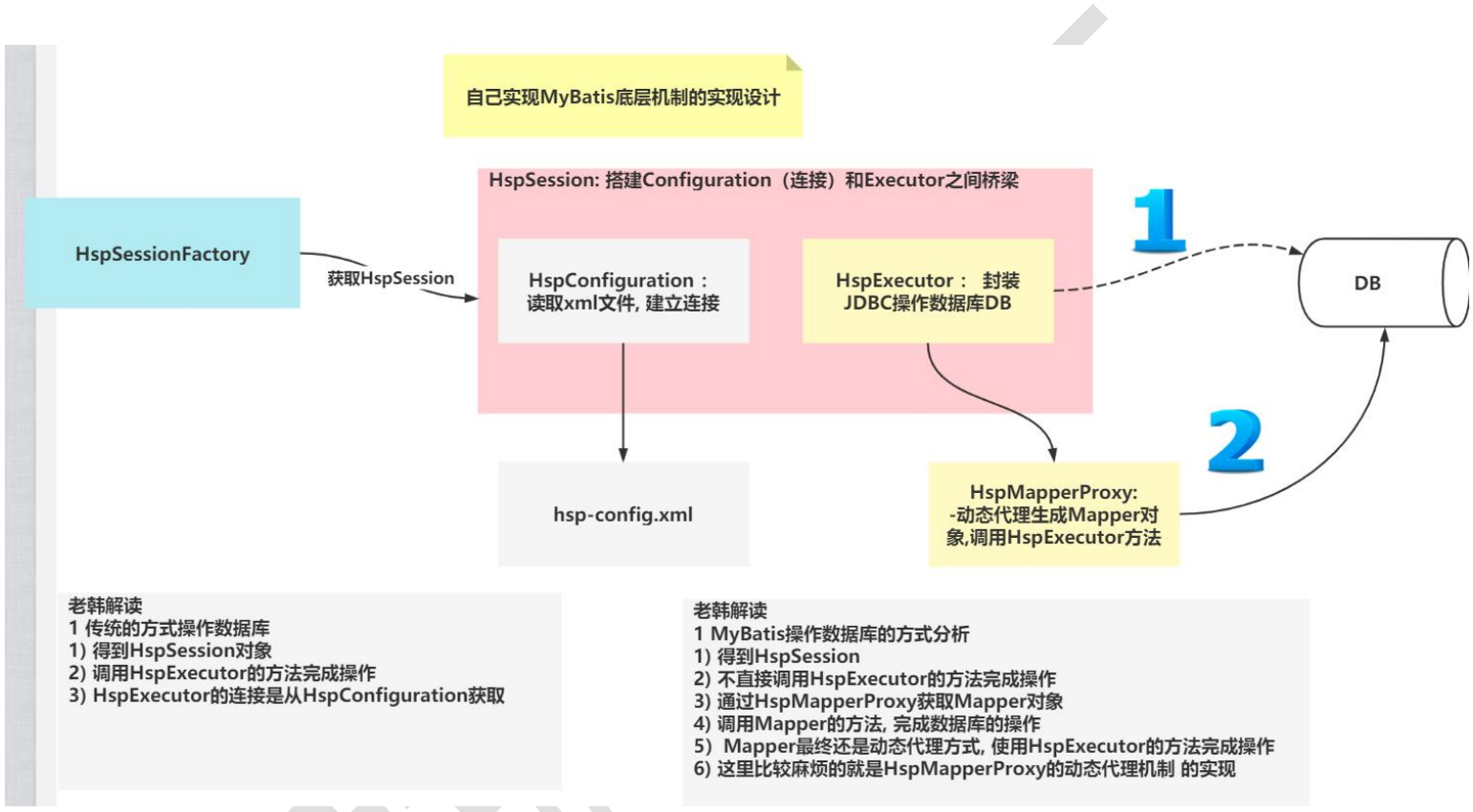
```
CREATE DATABASE `hsp_mybatis`;

USE `hsp_mybatis`;

CREATE TABLE `monster` (
    `id` INT NOT NULL AUTO_INCREMENT,
    `age` INT NOT NULL,
    `birthday` DATE DEFAULT NULL,
    `email` VARCHAR(255) NOT NULL,
    `gender` TINYINT NOT NULL,
    `name` VARCHAR(255) NOT NULL,
    `salary` DOUBLE NOT NULL,
    PRIMARY KEY (`id`)
) CHARSET=utf8

INSERT INTO `monster` VALUES(NULL, 200, '2000-11-11', 'nmw@sohu.com', 1,
'牛魔王', 8888.88)
```

4、到此：项目开发环境搭建 OK

3.3 Hsp-Mybatis 的设计思路

3.3.1 一图胜千言

1、自己写 Mybatis 的底层实现设计



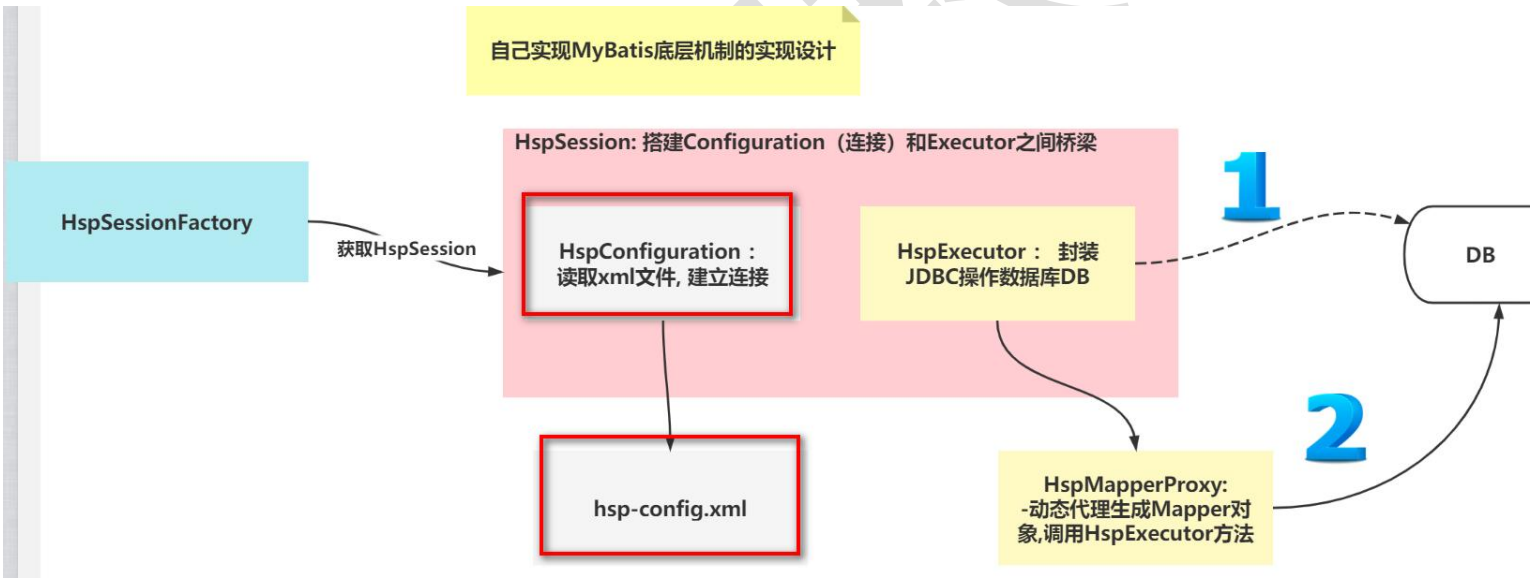
3.4 自己实现 MyBatis 底层机制 【封装 Sqlsession 到执行器 + Mapper 接口和 Mapper.xml + MapperBean + 动态代理代理 Mapper 的方法

3.4.1 实现任务阶段 **1-** 完成读取配置文件，得到数据库连

3.4.1.1 说明：通过配置文件，获取数据库连接

3.4.1.2 分析+代码实现

- 分析示意图



- **代码实现**

1. 创建

D:\hspedu_ssm_mybatis_temp\hsp-mybatis\src\main\resources\hsp_config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<database>
    <property name="driverClassName">com.mysql.jdbc.Driver</property>
    <property name="url">
jdbc:mysql://localhost:3306/hsp_mybatis_temp?useUnicode=true&amp;characterEnco
ding=utf8
</property>
    <property name="username">root</property>
    <property name="password">hsp</property>
</database>
```

2. 创建 D:\hspedu_ssm_mybatis_temp\hsp-mybatis\src\main\java\com\hspedu\hspmybatis\sqlsession\HspConfiguration.java

```
package com.hspedu.hspmybatis.sqlsession;

import org.dom4j.Document;
import org.dom4j.Element;
import org.dom4j.io.SAXReader;

import java.io.InputStream;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 * @author 韩顺平
 * @version 1.0
 * 读取与解析配置信息，并返回处理后的 Environment
 */
public class HspConfiguration {
    private static ClassLoader loader = ClassLoader.getSystemClassLoader();
    /**
     * 读取 xml 信息并处理
     */
    public Connection build(String resource) {
        try {
            InputStream stream = loader.getResourceAsStream(resource);
            SAXReader reader = new SAXReader();
            Document document = reader.read(stream);
            Element root = document.getRootElement();
            return evalDataSource(root);
        } catch (Exception e) {
```

```
                throw new RuntimeException("error occured while evaling xml " +
resource);
            }
        }

        private Connection evalDataSource(Element node) throws
ClassNotFoundException {
            if (!"database".equals(node.getName())) {
                throw new RuntimeException("root should be <database>");
            }
            String driverClassName = null;
            String url = null;
            String username = null;
            String password = null;
            //获取属性节点
            for (Object item : node.elements("property")) {
                Element i = (Element) item;
                String value = getValue(i);
                String name = i.attributeValue("name");
                if (name == null || value == null) {
                    throw
                        new RuntimeException("[database]: <property> should contain
```

```
name and value");
                }
                //赋值
                switch (name) {
                    case "url":
                        url = value;
                        break;
                    case "username":
                        username = value;
                        break;
                    case "password":
                        password = value;
                        break;
                    case "driverClassName":
                        driverClassName = value;
                        break;
                    default:
                        throw new RuntimeException("[database]: <property> unknown
name");
                }
            }
            Class.forName(driverClassName);
```

```
        Connection connection = null;
        try {
            //建立数据库链接
            connection = DriverManager.getConnection(url, username, password);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }

//获取 property 属性的值,如果有 value 值,则读取 没有设置 value,则读取内容
    private String getValue(Element node) {
        return node.hasContent() ? node.getText() : node.attributeValue("value");
    }
}
```

3.4.1.3 完成测试

1. 编写 D:\hspedu_ssm_mybatis_temp\hsp-mybatis\src\test\java\com\hspedu\test\HspMybatisTest.java

```
public class HspMybatisTest {
```

```
    public static void main(String[] args) {

        HspConfiguration hspConfiguration = new HspConfiguration();
        //获取到一个 Connection
        Connection connection = hspConfiguration.build("hsp_config.xml");
        System.out.println(connection);

    }
}
```

2. 测试效果

```
D:\program\hspjdk8\bin\java.exe ...
Wed Feb 23 20:23:59 CST 2022 WARN: Establ
com.mysql.jdbc.JDBC4Connection@4cdf35a9
```

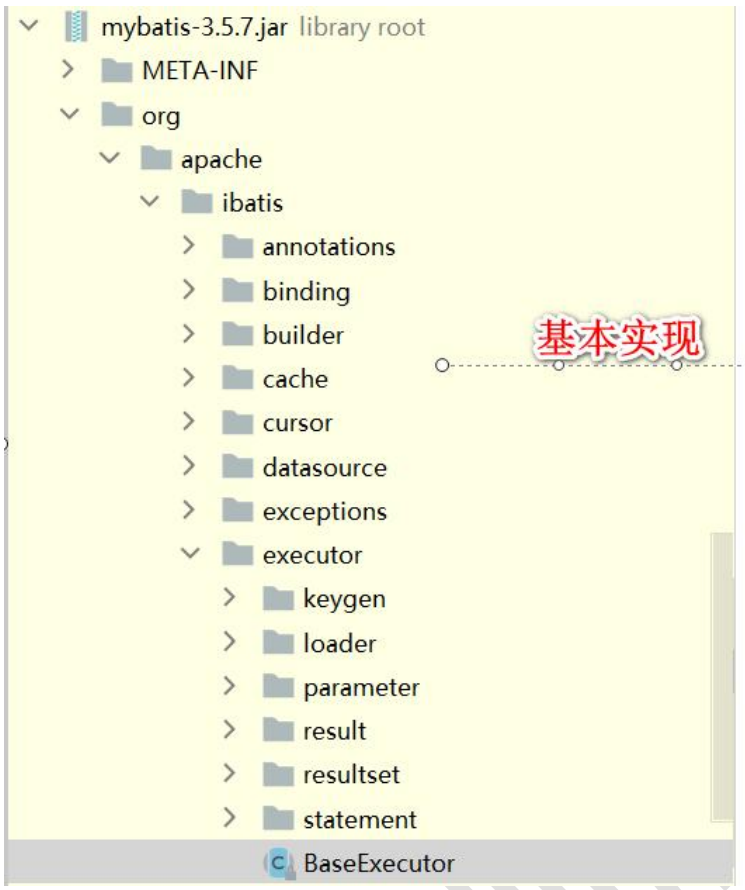
3.4.2 实现任务阶段 2- 编写执行器，输入 SQL 语句，完成操作

3.4.2.1 说明：通过实现执行器机制，对数据表操作

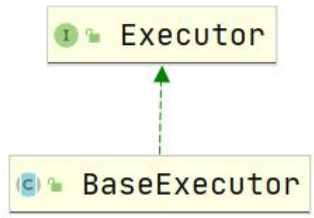
3.4.2.2 分析+代码实现

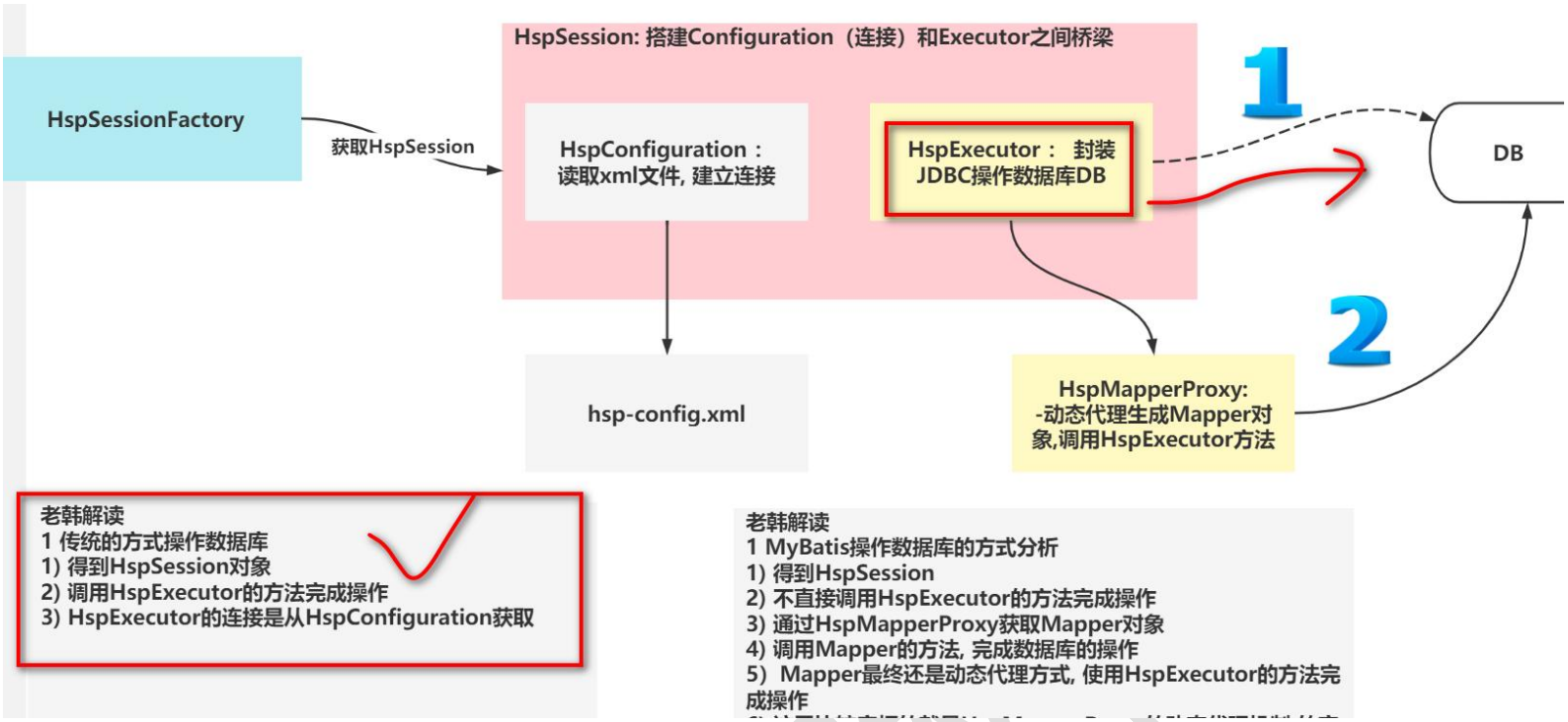
- 分析示意图

说明：我们把对数据库的操作，会封装到一套 Executor 机制中，程序具有更好的扩展性，结构更加清晰.



基本实现





- **代码实现**

1. 创建 D:\hspedu_ssm_mybatis_temp\hsp-mybatis\src\main\java\com\hspedu\entity\Monster.java

```
package com.hspedu.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;
```

```
import java.util.Date;


/**
 * @author 韩顺平
 * @version 1.0
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class Monster {
    private Integer id;
    private Integer age;
    private String name;
    private String email;
    private Date birthday;
    private double salary;
    private Integer gender;

}
```

2. 创 建

D:\hspedu_ssm_mybatis_temp\hsp-mybatis\src\main\java\com\hspedu\hspmybatis\sqlse

ssion\Executor.java

```
package com.hspedu.hspmybatis.sqlsession;


/**
 * @author 韩顺平
 * @version 1.0
 */
public interface Executor {
    public <T> T query(String statement, Object parameter);
}
```

3. 创 建 D:\hspedu_ssm_mybatis_temp\hsp-mybatis\src\main\java\com\hspedu\hspmybatis\sqlsession\HspExecutor.java

```
package com.hspedu.hspmybatis.sqlsession;


import com.hspedu.entity.Monster;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
/**
 * @author 韩顺平
 * @version 1.0
 */
public class HspExecutor implements Executor {
    private HspConfiguration hspConfiguration = new HspConfiguration();
    @Override
    public <T> T query(String sql, Object parameter) {
        Connection connection = getConnection();
        ResultSet set = null;
        PreparedStatement pre = null;
        try {
            pre = connection.prepareStatement(sql);
            //设置参数，如果参数多，可以使用数组来处理
            pre.setString(1, parameter.toString());
            set = pre.executeQuery();
            //这里可以使用，一套反射机制来创建，后面可以优化灵活.
            Monster monster = new Monster();
            //遍历结果集
            while (set.next()) {
                monster.setId(set.getInt("id"));
```

```
                monster.setName(set.getString("name"));
                monster.setEmail(set.getString("email"));
                monster.setAge(set.getInt("age"));
                monster.setGender(set.getInt("gender"));
                monster.setBirthday(set.getDate("birthday"));
                monster.setSalary(set.getDouble("salary"));
            }
            return (T) monster;
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (set != null) {
                    set.close();
                }
                if (pre != null) {
                    pre.close();
                }
                if (connection != null) {
                    connection.close();
                }
            } catch (Exception e2) {
```

```
                e2.printStackTrace();
            }
        }
        return null;
    }

    private Connection getConnection() {
        try {
            Connection connection = hspConfiguration.build("hsp_config.xml");
            return connection;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

3.4.2.3 完成测试

1. 修改 D:\hspedu_ssm_mybatis_temp\hsp-mybatis\src\test\java\com\hspedu\test\HspMybatisTest.java，增加测试方法

```
@Test
```

```
public void testHspExecutor() {


    Executor executor = new HspExecutor();
    Monster monster = executor.query("select * from monster where id = ?", 2);
    System.out.println("monster= " + monster);
}
```

2. 测试效果

```
✔ Tests passed: 1 of 1 test – 375 ms
D:\program\hspjdk8\bin\java.exe ...
Wed Feb 23 21:09:06 CST 2022 WARN: Establis
monster= Monster(id=2, age=200, name=牛魔王,

Process finished with exit code 0
```