Day01

Git

1. 登录 http://git.seehope.net/ 注册账号
2. 安装 git ： https://git-scm.com/

是什么

Git 是一个版本控制系统。

Git 核心

Git 有三种状态：

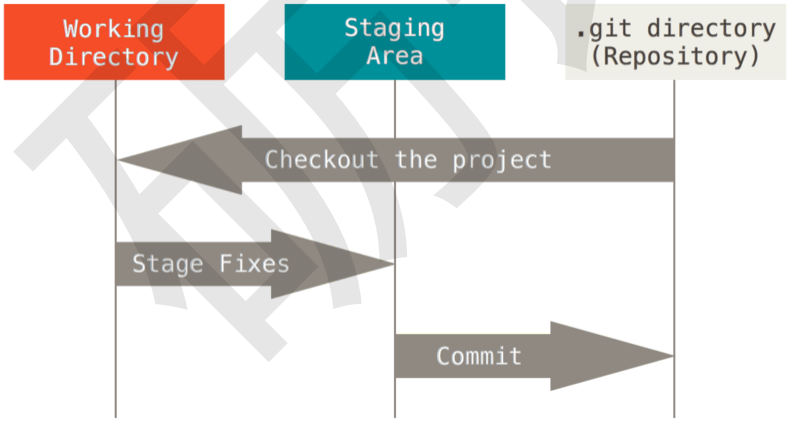
- 已提交（committed）：已修改表示修改了文件，但还没保存到数据库中。
- 已修改（modified）：已提交表示数据已经安全地保存在本地数据库中。
- 已暂存（staged）：已暂存表示对一个已修改文件的当前版本做了标记，使之包含在下次提交的快照中。

Shell 复制代码

```shell
# 查看状态
git status
# 修改状态为 staged
git add Hello.java
# 修改状态为 committed
git commit -m "feature: 新增 Hello.java"
```

Git 有三个阶段：

- 工作区（Working Directory）：
- 暂存区（Staging Area）：
- Git 目录（.git directory Repository）：



.gitignore文件

指定哪些文件不被 git 跟踪。

本地操作案例

1. 在git.seehope.net 注册：用户名（邮箱）和密码
2. 安装git
3. 打开git base
    a. 设置邮箱和用户名

```shell
$ git config --global user.email "xxx@qq.com"
$ git config --global user.name "xxx"
# 进入某个盘
cd /d/
# 如果目录有空格，需要加\
cd Second\ Stage
# 新增文件
mdkir test-git && cd test-git
# 1.初始化Git仓库
git init
# 2.添加文件
touch Hello.java
# 3.添加到暂存区
git add *
# 4.提交到Git仓库
git commit -m "新增Hello文件"
# 2,3,4步骤不断循环。。。
```

推送远程仓库案例

1. 在git.seehope.net 中新增仓库
2. 拉取（克隆）仓库到本地

```shell
git clone http://git.seehope.net/liufuxu/test-push.git
```

3. 进入克隆到本地的仓库文件夹中

```shell
cd test-push
```

4. 进行跟本地git一样的操作

```shell
touch Hello.java
git add *
git commit -m "新增Hello文件"
```

5. 推送到远程仓库

```shell
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 227 bytes | 227.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: . Processing 1 references
remote: Processed 1 references in total
To http://git.seehope.net/liufuxu/test-push.git
 * [new branch]      master -> master
```

6. 添加 .gitignore 文件

Shell 复制代码

```
# 没有把 *.class 添加 .gitignore 文件时，提示 .class 没被跟踪。
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Hello.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Hello.class

no changes added to commit (use "git add" and/or "git commit -a")

# 添加 .gitignore 文件
$ echo "*.class" >> .gitignore

# 此时，git不再提示 Hell.class 文件未被跟踪
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Hello.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

$ git commit -m "修改了Hello.java，增加了.gitignore 文件"
[master aadbc5b] 修改了Hello.java，增加了.gitignore 文件
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore

$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 6 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 429 bytes | 429.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote: . Processing 1 references
```

```
remote: Processed 1 references in total
To http://git.seehope.net/liufuxu/test-push.git
   7e0b597..aadbc5b  master -> master
```

撤回

Plain Text 复制代码

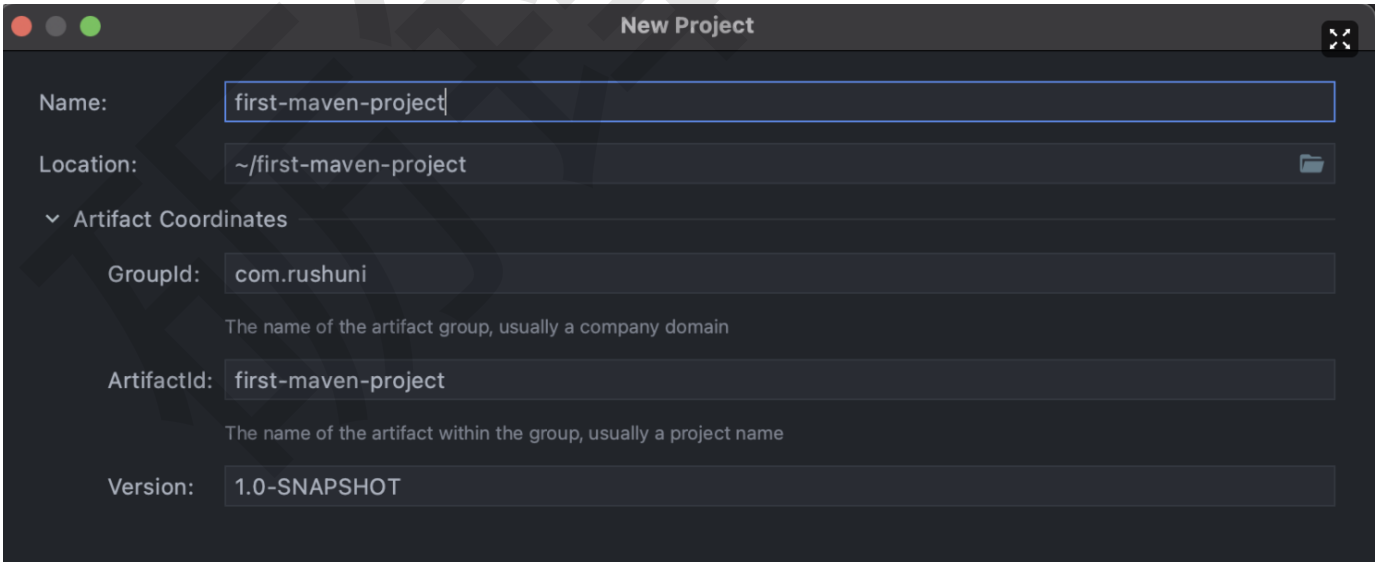
```
git revert xxxx
git reset --hard xxxx
git checkout xxxx
```

课堂练习

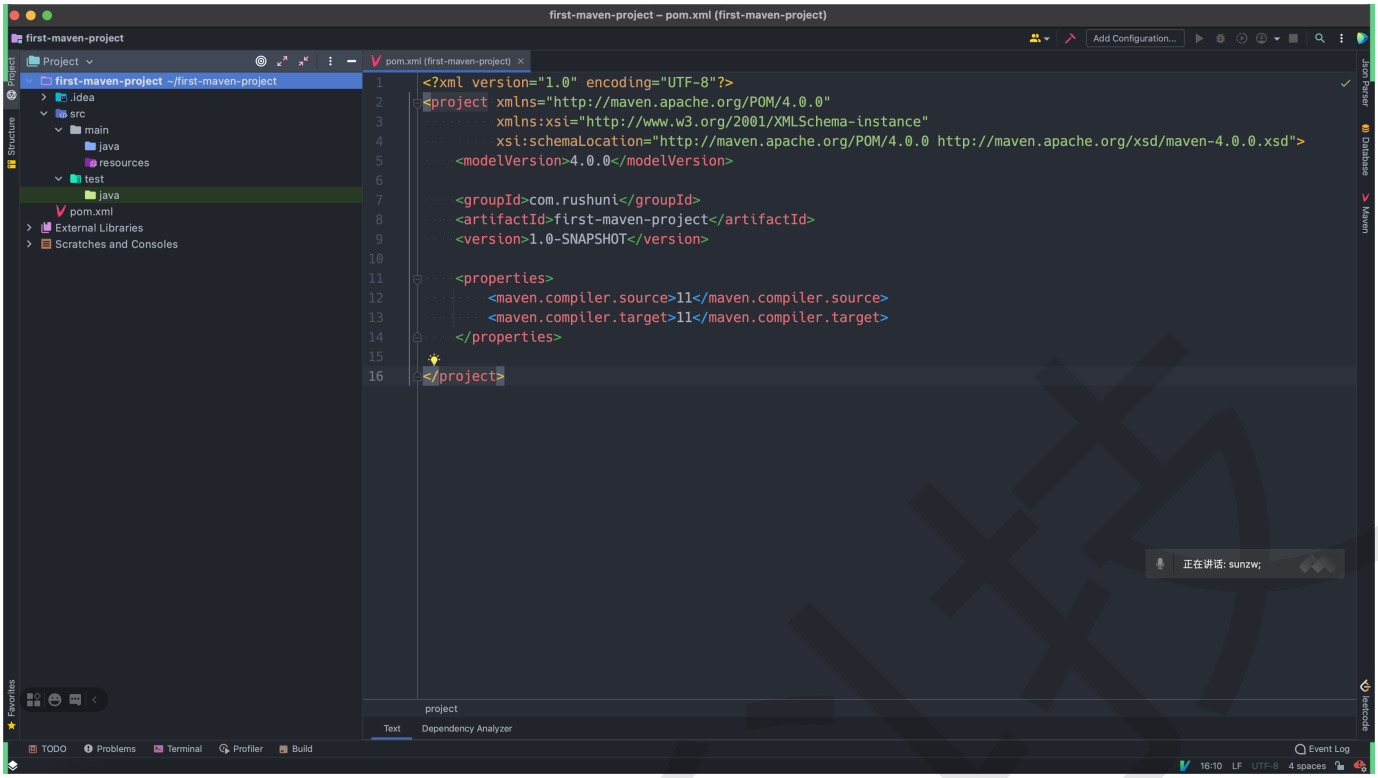
1. 新增本地git仓库
2. 新增远程仓库

Maven

新建 maven 项目：

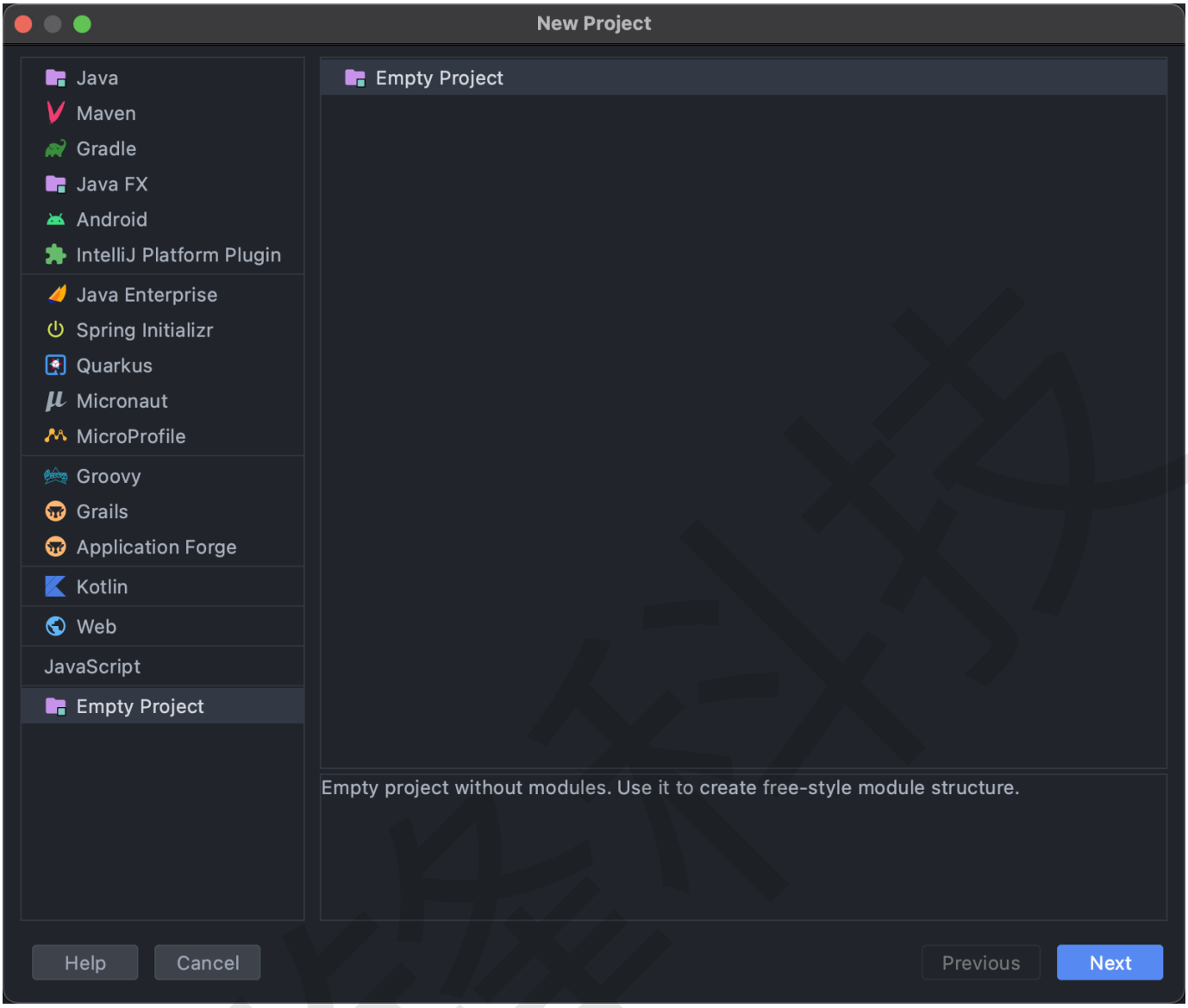


得到项目：

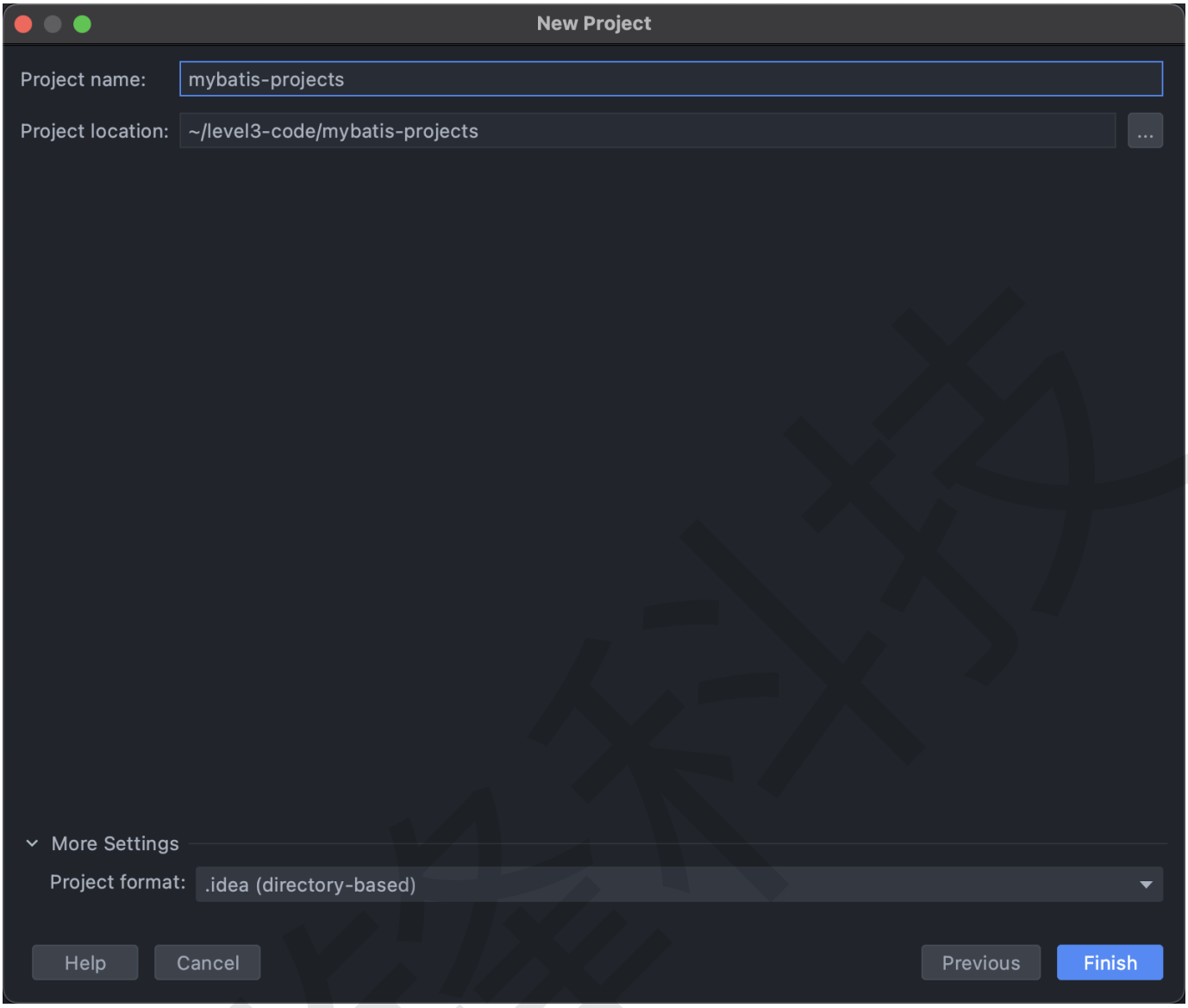


Junit 5

新建empty project 项目：

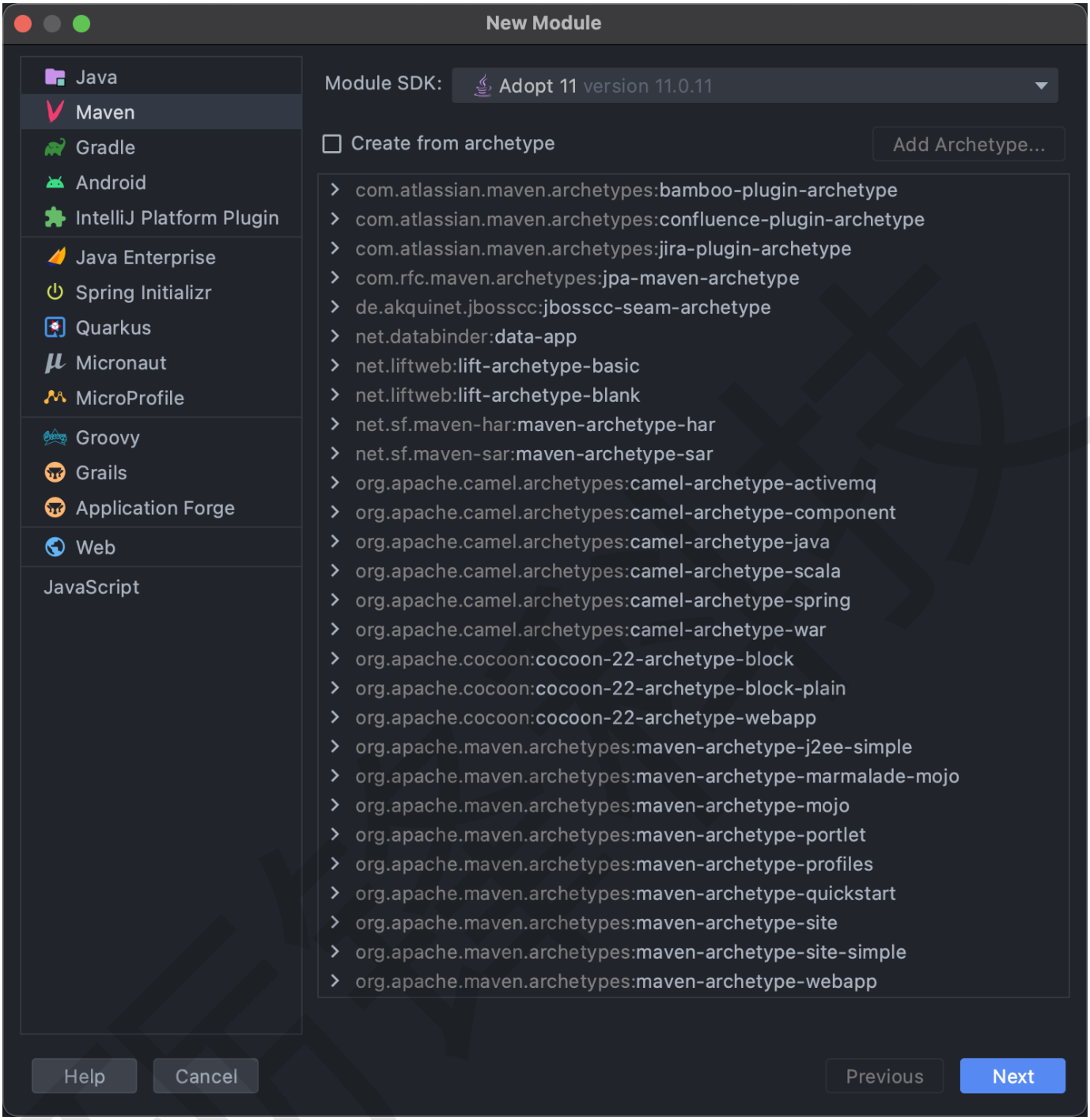

New Project
Java
Maven
Gradle
Java FX
Android
IntelliJ Platform Plugin
Java Enterprise
Spring Initializr
Quarkus
Micronaut
MicroProfile
Groovy
Grails
Application Forge
Kotlin
Web
JavaScript
Empty Project
Empty Project
Empty project without modules. Use it to create free-style module structure.
Help
Cancel
Previous
Next

输入项目名称：

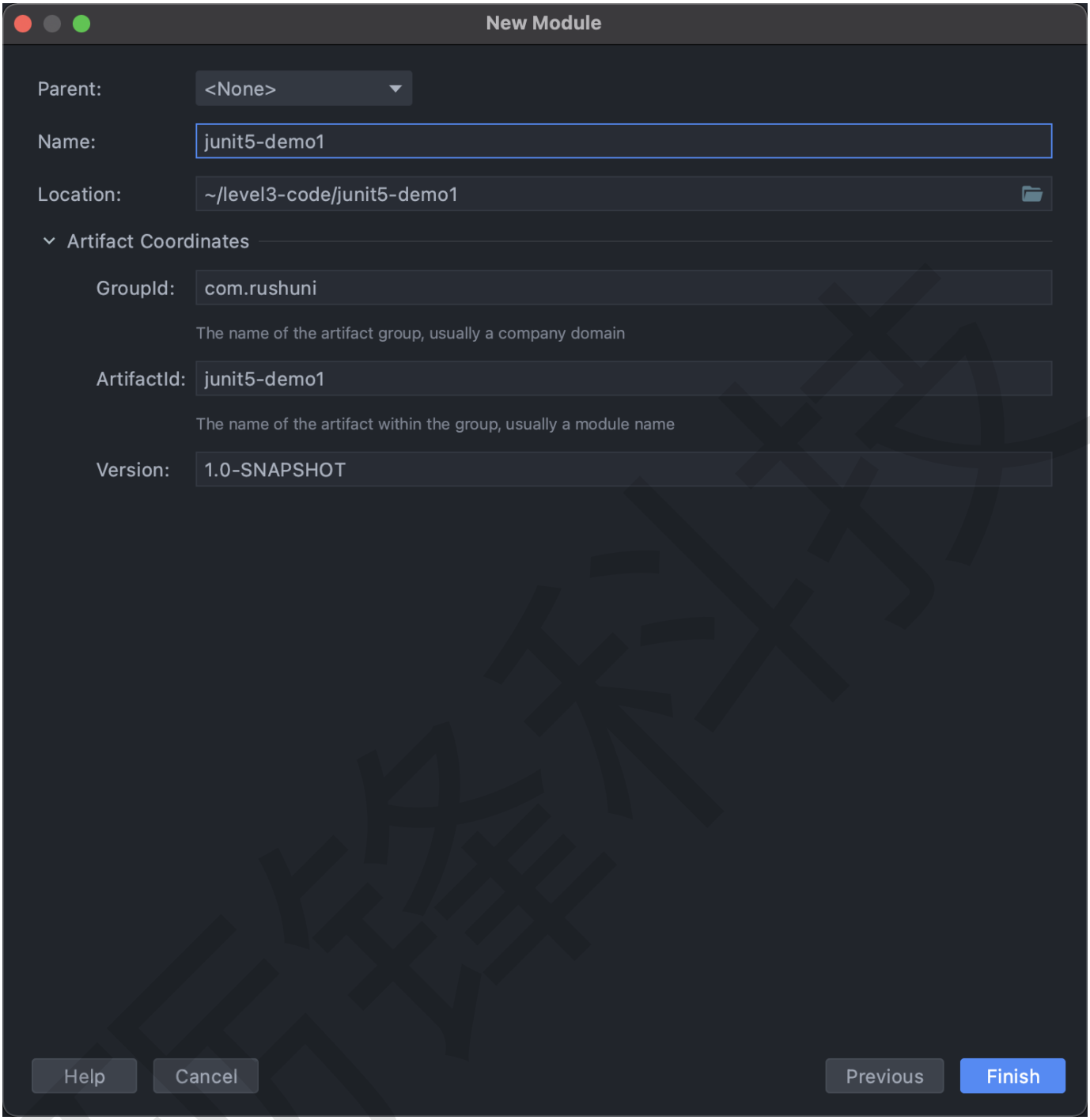


项目中新建 module：

选择 maven：


New Module
Java
Maven
Gradle
Android
IntelliJ Platform Plugin
Java Enterprise
Spring Initializr
Quarkus
Micronaut
MicroProfile
Groovy
Grails
Application Forge
Web
JavaScript
Module SDK:
Adopt 11 version 11.0.11
Create from archetype
Add 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
org.apache.maven.archetypes:maven-archetype-site
org.apache.maven.archetypes:maven-archetype-site-simple
org.apache.maven.archetypes:maven-archetype-webapp
Help
Cancel
Previous
Next

输入项目信息：



修改 pom.xml，添加 junit5 依赖：

XML 复制代码

```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.rushuni</groupId>
    <artifactId>junit5-demo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.7.2</version>
            <scope>test</scope>
        </dependency>
        <!-- Only needed to run tests in a version of IntelliJ IDEA that
bundles older versions -->
        <dependency>
            <groupId>org.junit.platform</groupId>
            <artifactId>junit-platform-launcher</artifactId>
            <version>1.7.2</version>
            <scope>test</scope>
        </dependency>

    </dependencies>


</project>
```

编写需要的测试的类和方法：

```java
package com.rushuni.controller;

/**
 * @author rushuni
 * @date 2021年07月12日 4:20 下午
 */
public class Caculator {

    public int sum(int a, int b) {
        return a + b;
    }
}
```

编写测试：

```java
package com.rushuni.controller;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

class CaculatorTest {

    @Test
    public void testSum() {
        Caculator calculator = new Caculator();
        assertEquals(3, calculator.sum(1, 2), "1 + 2 should equal 3");
    }

}
```

课堂练习

- 创建 maven 项目
- 使用 Junit 5 完成一个测试

MyBatis 3.5.x

快速入门

1. ORM 框架介绍

MyBatis 是一种 ORM （Object/Relational Mapping）框架。

> 以 MyBatis 3.5.7 版本的代码为基础：
> 中文官方地址：https://mybatis.org/mybatis-3/zh/index.html
> 源码地址：https://github.com/mybatis/mybatis-3

在 Java 领域，流行的 ORM 框架有 Hibernate，Spring Data JPA 以及 MyBatis。

所有的 ORM 框架都是为了解决对象模型与关系模型的映射问题。

说白了就是如何进行从 Java 类到数据库表，以及从 Java 数据类型到 SQL 数据类型的映射。

在学习 ORM 框架的过程中，对于映射有几个常见的概念需要提前了解：

Java 数据类型	SQL 数据类型	对应代码
类（Class）	数据库表（Table）	public class stu{} <-> create table stu(...);
对象（Object）	记录（Record，行）	new stu() <-> select * from stu;
对象的属性（Attribute）	字段（Field）	stu.getName() <-> desc stu;

> 有些人不喜欢 ORM 框架，认为 ORM 框架很复杂，性能差以及学习曲线陡峭。
> 对此，Martin Fowler 对此表示，其实大家都忽略了对象/关系映射问题的复杂度。
> 本质上，解决对象/关系映射问题是在解决两种完全不同的数据表示之间的同步，一种是关系数据库中，另一种则是在内存。所以，虽然 ORM 的意思是对象关系模型，但是，其实又和对象没有关系。

2. 为什么使用 MyBatis

实际开发 Java 程序时，我们很少回去直接使用 JDBC 去完成各种各样的数据库操作。

以执行一条 Select 查询语句作为例子，使用 JDBC 操作的核心步骤如下：

1. 注册数据库驱动以及初始化连接数据库的必要信息
    a. 驱动名，用户名、密码及其他连接信息；
2. 调用 DriverManager
    a. 需要调用 getConnection() 方法，以创建 Connection 来连接数据库；
3. 创建 Statement 对象
    a. 调用 Connection 的 createStatement() 或 prepareStatement() 方法。
    b. 此时会指定 SQL（或是 SQL 语句模板 + SQL 参数）。
4. 执行 SQL 语句，得到 ResultSet 对象，也就是查询结果集；
    a. 调用 Statement 对象的执行方法，执行结果将被保存到 ResultSet 对象汇总。
5. 把 ResultSet 中结果转换为 Java 对象
    a. 遍历 ResultSet，从结果集中读取数据，并将每一行数据库记录转换成一个对象；
6. 释放所有资源
    a. 关闭 ResultSet 结果集、Statement 对象及数据库 Connection，避免这些对象占用底层资源。

如果使用 JDBC，上述所有步骤大都会重复出现。

当然，我们可以对这些步骤进行封装，但是要做到通用化并且兼顾灵活性，则并不容易。

所以实际项目中，我们通常会选取一款适合项目业务和项目团队的 ORM 框架，因为框架本身已经实现了对象模型、关系模型之间转换。

MyBatis 和其他 ORM 框架的核心功能一样：根据配置（配置文件或是注解）实现对象模型、关系模型两者之间无感知的映射。

更重要的是我们国内的业务需求复杂多变，通常需要编写和修改复杂的 SQL，而 MyBatis 的强项正是可以让我们在Mapper 映射文件中直接编写 SQL 语句，并且提供了强大的动态 SQL 功能，避免了拼接 SQL 语句字符串时枯燥且容易出错的问题。

3. 入门 MyBatis 的正确方式

3.1 创建 Maven 项目

pom.xml

XML 复制代码

```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.rushuni</groupId>
    <artifactId>mybatis-1</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
        <mybatis.version>3.5.7</mybatis.version>
        <mysql-connector-java.version>8.0.25</mysql-connector-java.version>
        <junit5.version>5.7.2</junit5.version>
    </properties>

    <dependencies>
        <!-- MyBatis -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>${mybatis.version}</version>
        </dependency>
        <!--MySQL JDBC -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>${mysql-connector-java.version}</version>
        </dependency>

        <!-- Junit 5 Start -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>${junit5.version}</version>
            <scope>test</scope>
        </dependency>
        <!-- Only needed to run tests in a version of IntelliJ IDEA that
bundles older versions -->
        <dependency>
            <groupId>org.junit.platform</groupId>
            <artifactId>junit-platform-launcher</artifactId>
            <version>1.7.2</version>
            <scope>test</scope>
```

```
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit5.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.vintage</groupId>
            <artifactId>junit-vintage-engine</artifactId>
            <version>${junit5.version}</version>
            <scope>test</scope>
        </dependency>
        <!-- Junit 5 End -->
    </dependencies>

</project>
```

3.2 创建测试数据表

SQL 复制代码

```sql
create table student (
  id int unsigned primary key auto_increment,
  name varchar(20),
  phone varchar(20),
  age int(3)
);
```

3.3 创建实体类

Java 复制代码

```java
package com.rushuni.mybatis_demo.dao;

/**
 * @author rushuni
 * @date 2021年07月07日 4:10 下午
 */
public class Student {
    private Integer id;
    private String name;
    private String phone;
    private Integer age;
  // 下面省略了 getter/setter 方法。。。
}
```

3.4 编写 DAO 层接口

Java 复制代码

```java
/**
 * @author rushuni
 * @date 2021年07月07日 4:11 下午
 */
public interface StudentMapper {

    /**
     * 查询所有学生信息
     * @return
     */
    List<Student> listStudents();
}
```

3.5 编写 DAO 接口对应的映射文件

上一节定义了 StudentMapper 接口，接着是对该接口的实现。

在实际使用 MyBatis 的过程中，我们是不需要去写实现类的，而是在 /resources/mapper 目录下配置相应的映射文件 – StudentMapper.xml。

在该文件中定义需要执行的 SQL 语句以及查询结果集的映射规则。

StudentMapper.xml 明显不是一个 Java 文件，而是一个 XML 文件。

MyBatis 底层会生成一个实现了 StudentMapper 接口的代理对象来执行 StudentMapper.xml 文件中的 SQL 语句。

为了做到这一点，MyBatis 对于这个实现 DAO 层接口的 XML映射文件有两点必须要遵循的规范：

1. 映射文件中的 namespace 属性必须和接口的全限定路径一致。
2. 映射文件中的 id 属性必须和接口中对应方法的名称一致。

另外，还有两个可选的规范：

1. 映射文件和接口使用相同的命名。
2. 映射文件和接口放在同一个目录。

如果遵循了可选规范，那么我们可以在下一步创建的 MyBatis核心配置文件中直接使用 package 标签进行映射文件扫描。

这里两个可选规范，一般我们会遵循第一个。

对于第二个，实际项目中，MyBatis 通常和 Spring 框架一起使用，自动扫描的工作届时会交给 Spring，所以我们会把 xml 映射文件放在一个统一的地方，也就是 /resources/mapper 目录。

StudentMapper.xml

XML 复制代码

```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">
<!-- namespace：接口名 -->
<mapper namespace="com.rushuni.mybatis_demo.dao.StudentMapper">
    <!--
    <select>: 查询数据， 标签内只能写 select 语句
    id: 接口中的方法名
    resultType: 查询语句的返回结果数据类型，也是方法返回值的类型，这里使用完整类名。
    -->
    <select id="listStudents"
resultType="com.rushuni.mybatis_demo.entity.Student">
        <!--要执行的 sql 语句-->
        select id, name, phone, age from student
    </select>
</mapper>
```

3.6 编写 MyBatis 核心配置文件

参考官方文档，MyBatis 核心配置文件通常名称为 mybatis-config.xml。

Mybatis 核心配置文件是 MyBatis 框架配置的入口，其中配置了数据库地址、映射文件 xxxMapper.xml 的位置以及一些自定义变量和别名等等。

通常，我们会把它放在 /resource 目录下。

mybatis-config.xml

```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>
  <settings>
    <setting name="logImpl" value="STDOUT_LOGGING" />
  </settings>

  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
        <property name="driver" value="com.mysql.cj.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/mybatis_demo?useUnicode=true&amp;characterEncoding=UTF-8"/>
        <property name="username" value="${username}"/>
        <property name="password" value="${password}"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="mapper/StudentMapper.xml"/>
  </mappers>
</configuration>
```

Mybatis 的默认不会打印出 SQL 执行日志，所以，不要忘了开启 MyBatis 的日志，以方便我们查看 SQL 语句执行情况：

```xml
<!-- 开启默认日志功能 -->
<settings>
  <setting name="logImpl" value="STDOUT_LOGGING" />
</settings>
```

日志开启：https://mybatis.org/mybatis-3/logging.html。

告诉 MyBatis 实现了接口的 Mapper 映射文件在哪里有几种方法：

```xml
<!-- Using classpath relative resources -->
<mappers>
  <mapper resource="org/mybatis/builder/AuthorMapper.xml"/>
  <mapper resource="org/mybatis/builder/BlogMapper.xml"/>
  <mapper resource="org/mybatis/builder/PostMapper.xml"/>
</mappers>
```

```xml
<!-- Using url fully qualified paths -->
<mappers>
  <mapper url="file:///var/mappers/AuthorMapper.xml"/>
  <mapper url="file:///var/mappers/BlogMapper.xml"/>
  <mapper url="file:///var/mappers/PostMapper.xml"/>
</mappers>
```

```xml
<!-- Using mapper interface classes -->
<mappers>
  <mapper class="org.mybatis.builder.AuthorMapper"/>
  <mapper class="org.mybatis.builder.BlogMapper"/>
  <mapper class="org.mybatis.builder.PostMapper"/>
</mappers>
```

```xml
<!-- Register all interfaces in a package as mappers -->
<mappers>
  <package name="org.mybatis.builder"/>
</mappers>
```

上图内容来自官方文档：https://mybatis.org/mybatis-3/configuration.html#mappers

3.7 编写 Junit 5 测试类

Java 复制代码

```java
package com.rushuni.mybatis_demo.dao;

import com.rushuni.mybatis_demo.entity.Student;
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 org.junit.jupiter.api.Test;

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

class StudentMapperTest {

    @Test
    public void testListStudent() throws IOException {
        String resource = "mybatis-config.xml";
        InputStream inputStream = Resources.getResourceAsStream(resource);
        SqlSessionFactory sqlSessionFactory =
                    new SqlSessionFactoryBuilder().build(inputStream);
        try (SqlSession session = sqlSessionFactory.openSession()) {
            StudentMapper mapper = session.getMapper(StudentMapper.class);
            List<Student> studentList = mapper.listStudents();
            studentList.forEach(System.out::println);
        }
    }
}
```

作业

1. 复习并总结今天内容：git，maven，junit5，mybatis。
2. 预习 MyBatis 的内容，参考第二阶段课件 Day25 半天。