

IDE 中常用的插件

第 1 章 在 Eclipse 中使用 Maven

1.1 安装 Maven 核心程序

- 1) 下载地址: <http://maven.apache.org/>
- 2) 检查 JAVA_HOME 环境变量。Maven 是使用 Java 开发的，所以必须知道当前系统环境中 JDK 的安装目录。

```
C:\Users\韩总>echo %JAVA_HOME%  
D:\Java\jdk1.8.0_111
```

- 3) 解压 Maven 的核心程序。

将 apache-maven-3.6.3-bin.zip 解压到一个**非中文无空格**的目录下。例如：

```
D:\apache-maven-3.6.3
```

- 4) 配置环境变量。

MAVEN_HOME
D:\apache-maven-3.6.3

```
path
```

```
%MAVEN_HOME%\bin
```

- 5) ④查看 Maven 版本信息验证安装是否正确

C:\Users\韩总>mvn -v Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)

```
Maven home: D:\apache-maven-3.6.3\bin\..  
Java version: 1.8.0_111, vendor: Oracle Corporation, runtime:  
D:\Java\jdk1.8.0_111\jre  
Default locale: zh_CN, platform encoding: GBK  
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

1.2 配置本地仓库和阿里云镜像

1.2.1 配置本地仓库

- 1) Maven 的核心程序并不包含具体功能，仅负责宏观调度。具体功能由插件来完成。Maven 核心程序会到本地仓库中查找插件。如果本地仓库中没有就会从远程中央仓库下载。此时如果不能上网则无法执行 Maven 的具体功能。为了解决这个问题，我们可以将 Maven 的本地仓库指向一个在联网情况下下载好的目录。
- 2) Maven 默认的本地仓库：~\.m2\repository 目录。
Tips: ~表示当前用户的家目录。
- 3) 找到 Maven 的核心配置文件 settings.xml 文件：

```
解压目录 D:\apache-maven-3.6.3\conf\settings.xml
```

- 4) 设置方式

```
<localRepository>本地仓库的路径</localRepository>  
<localRepository>E:\LocalRepository</localRepository>
```

1.2.2 配置阿里云镜像

为了下载 jar 包方便，在 Maven 的核心配置文件 settings.xml 文件的 <mirrors></mirrors> 标签里面配置以下标签：

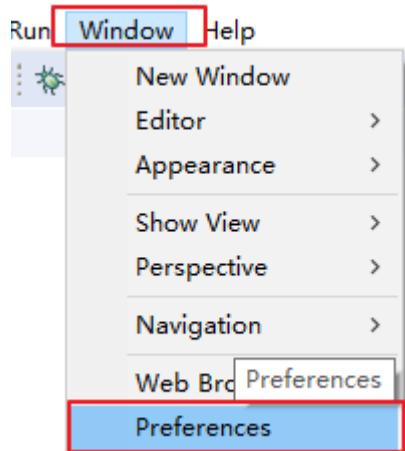
```
<mirror>  
  <id>nexus-aliyun</id>  
  <mirrorOf>central</mirrorOf>
```

```
<name>Nexus aliyun</name>
<url>http://maven.aliyun.com/nexus/content/groups/public</url>
</mirror>
```

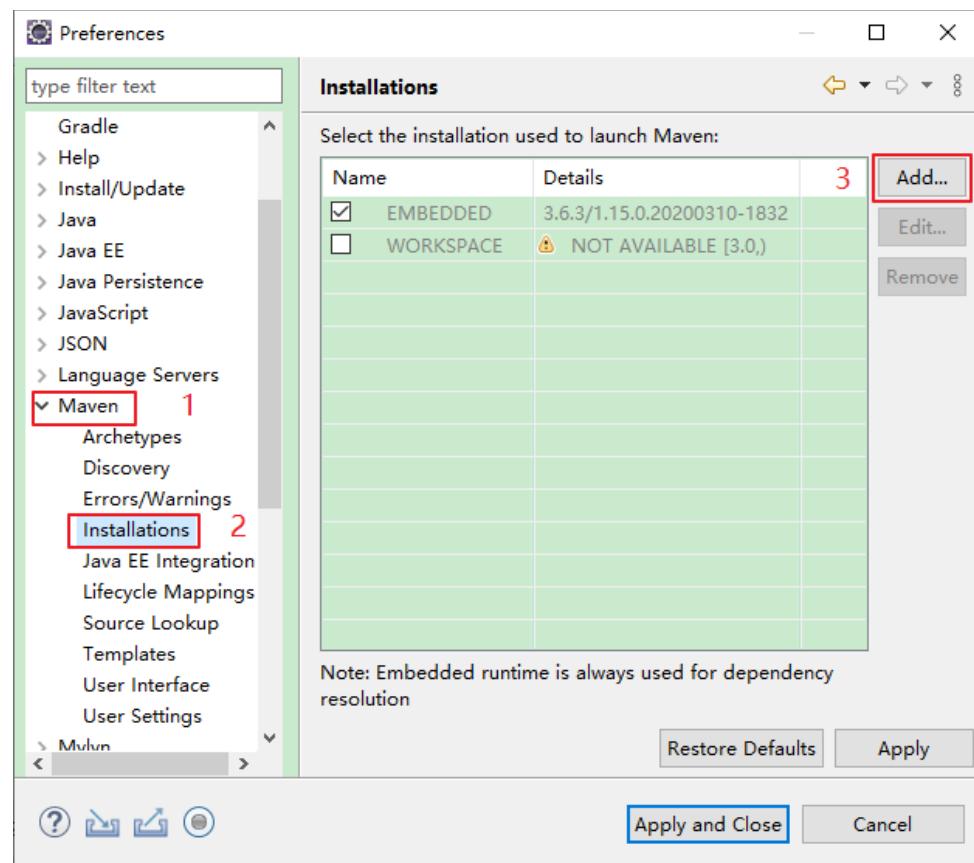
1.3 在 Eclipse 中配置 Maven

Eclipse 中默认自带 Maven 插件，但是自带的 Maven 插件不能修改本地仓库，所以通常我们不使用自带的 Maven，而是使用自己安装的，在 Eclipse 中配置 Maven 的步骤如下：

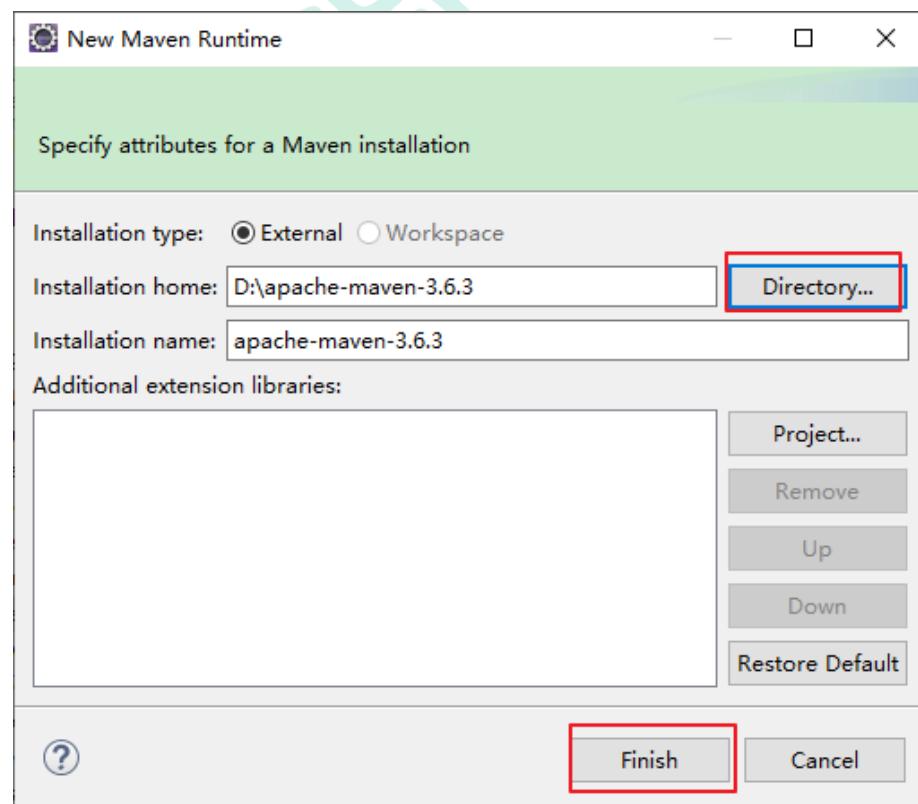
- 1) 点击 Eclipse 中的 Window→Preferences



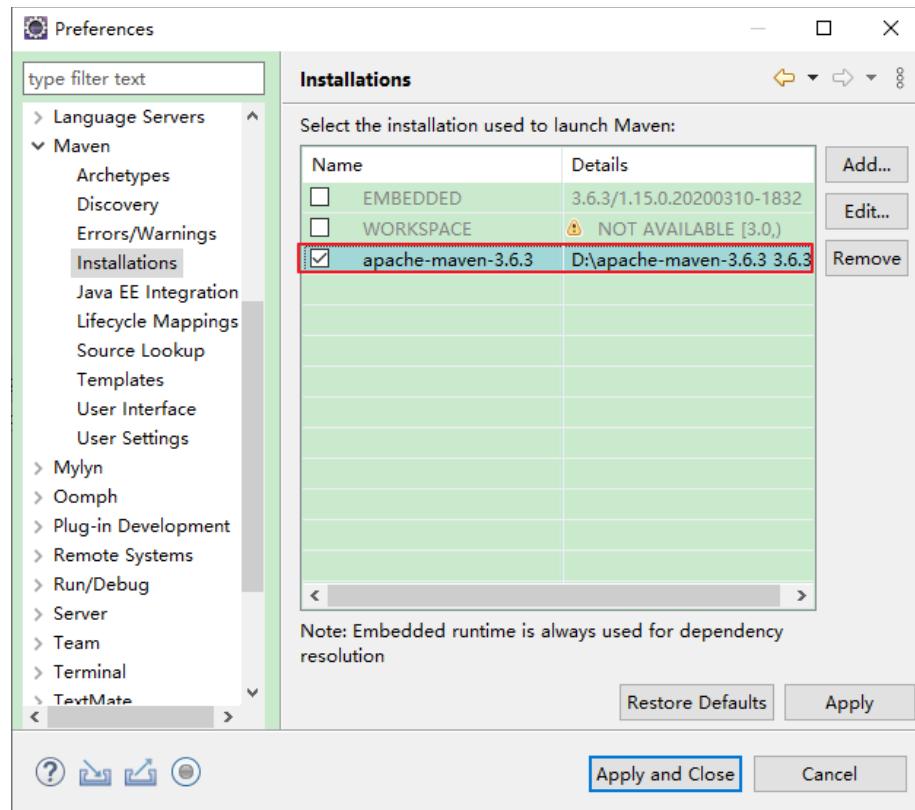
- 2) 点开 Maven 前面的箭头，选择 Installations，点击 Add...



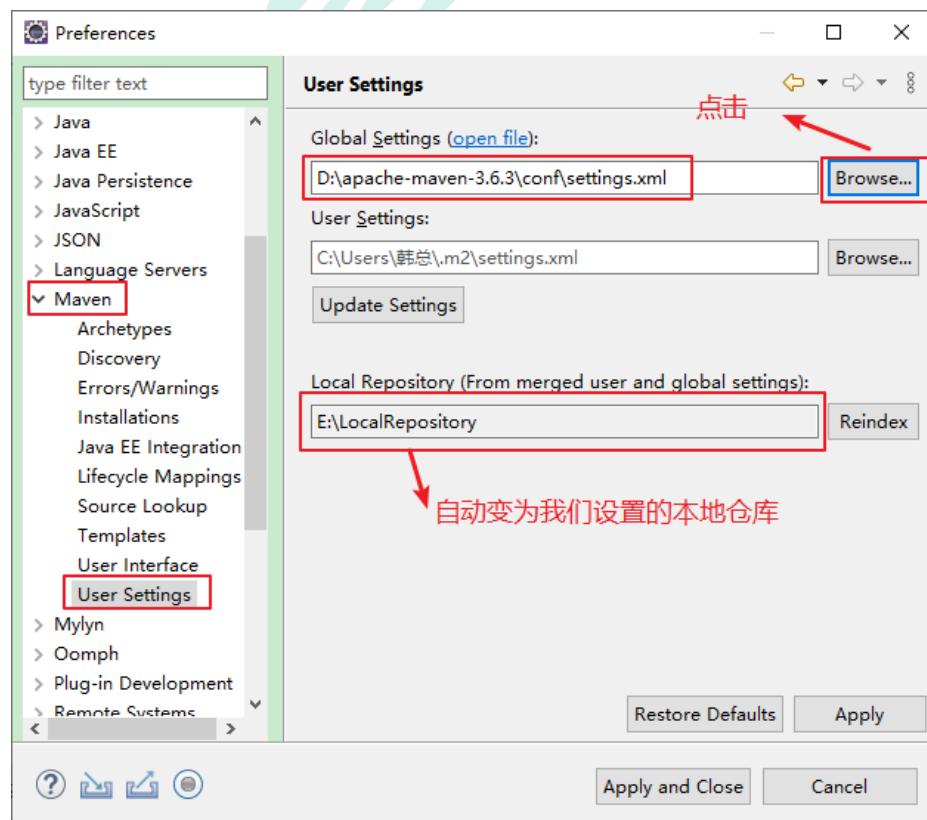
3) 点击 Directory...选择我们安装的 Maven 核心程序的根目录，然后点击 Finish



4) 勾上添加的 Maven 核心程序



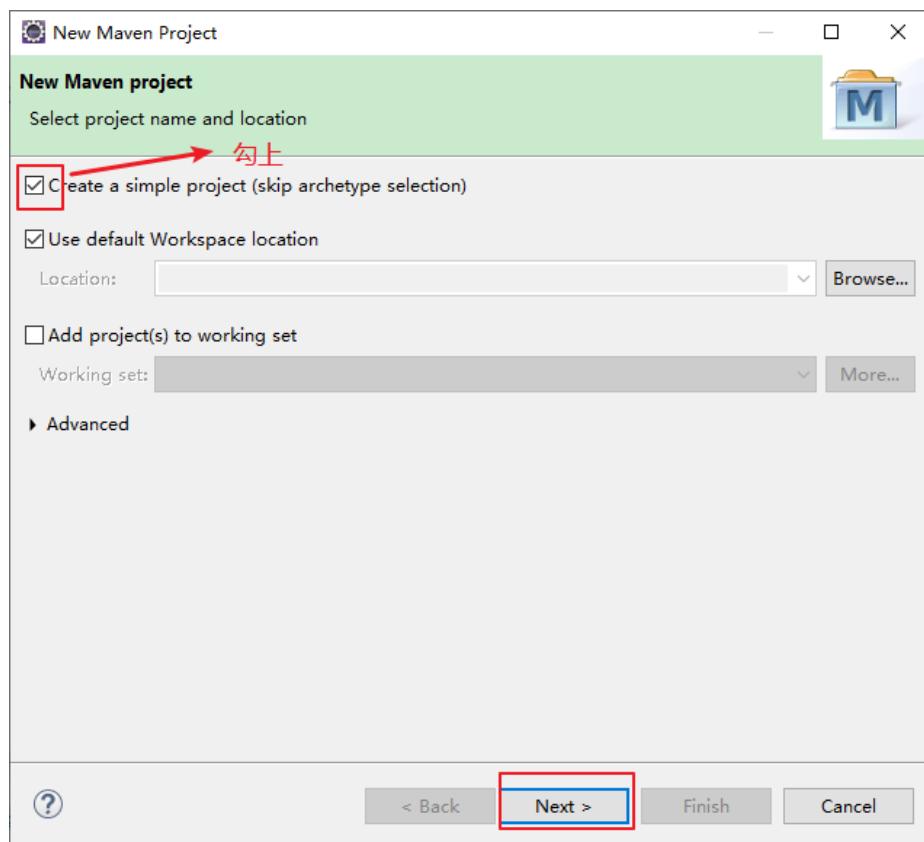
- 5) 选择 Maven 下的 User Settings，在全局设置哪儿点击 Browse... 选择 Maven 核心程序的配置文件 settings.xml，本地仓库会自动变为我们在 settings.xml 文件中设置的路径



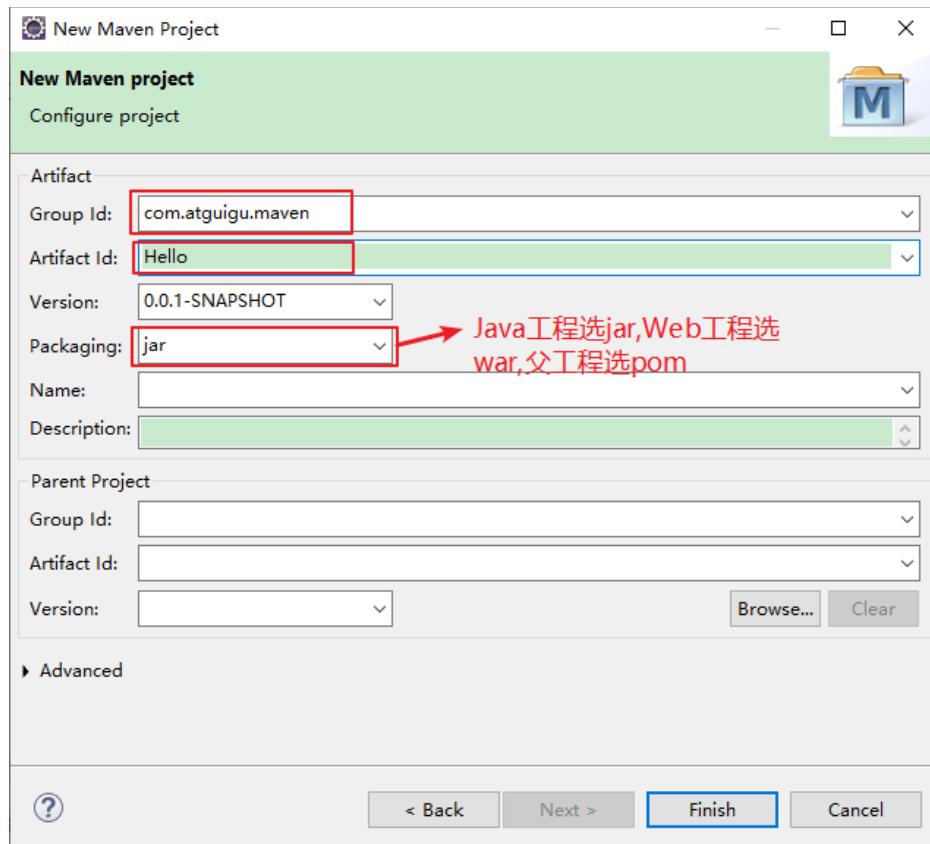
1.4 在 Eclipse 中创建 Maven 项目

1.4.1 创建 Java 工程

1) 点击 File→New→Maven Project, 弹出如下窗口



2) 点击 Next, 配置坐标 (GAV) 及打包方式, 然后点击 Finish



- 3) 创建完工程之后发现默认的 JDK 的编译版本是 1.5，在 Maven 的核心配置文件 settings.xml 文件中添加以下配置将编译版本改为 1.8，重启 Eclipse 即可

```
<profile>
<id>jdk-1.8</id>
<activation>
<activeByDefault>true</activeByDefault>
<jdk>1.8</jdk>
</activation>
<properties>
<maven.compiler.source>1.8</maven.compiler.source>
<maven.compiler.target>1.8</maven.compiler.target>

<maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>
</properties>
</profile>
```

- 4) 配置 Maven 的核心配置文件 pom.xml 文件

```
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.atguigu.maven</groupId>
<artifactId>Hello</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
</dependencies>
</project>
```

5) 编写主代码

在 src/main/java 目录下创建包并创建 Hello.java 文件

```
package com.atguigu.maven;

public class Hello {
    public String sayHello(String name){
        return "Hello "+name+"!";
    }
}
```

6) 编写测试代码

在 src/test/java 目录下创建包并创建 HelloTest.java 文件

```
package com.atguigu.maven;

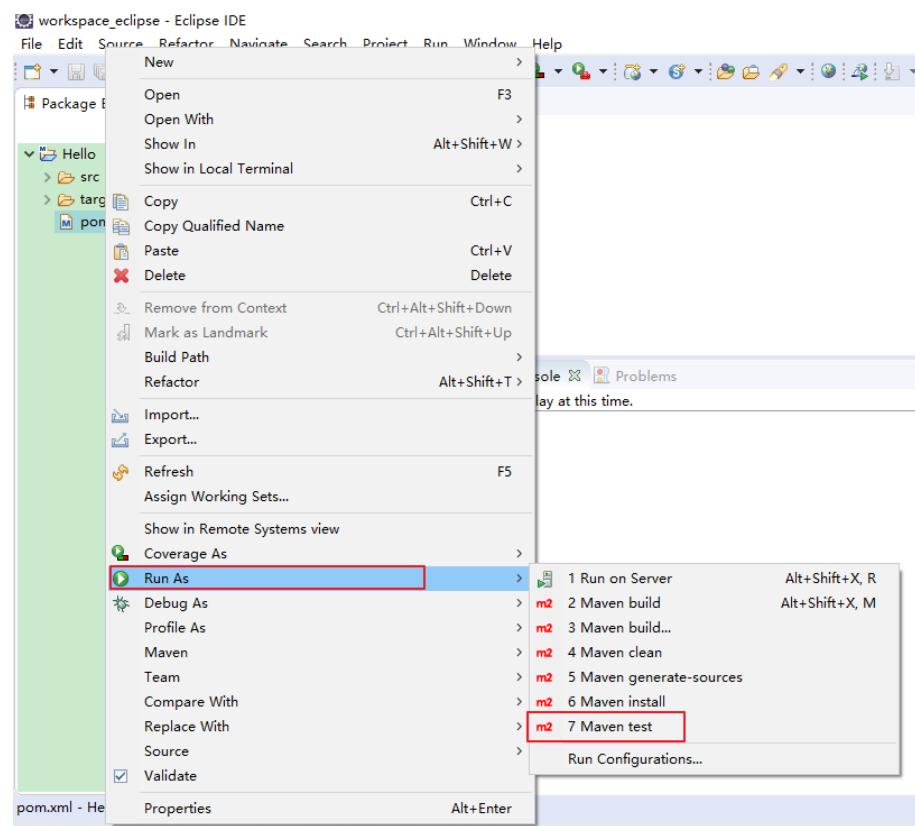
import org.junit.Test;

public class HelloTest {

    @Test
    public void testHello() {
        Hello hello = new Hello();
        String maven = hello.sayHello("Maven");
        System.out.println(maven);
    }
}
```

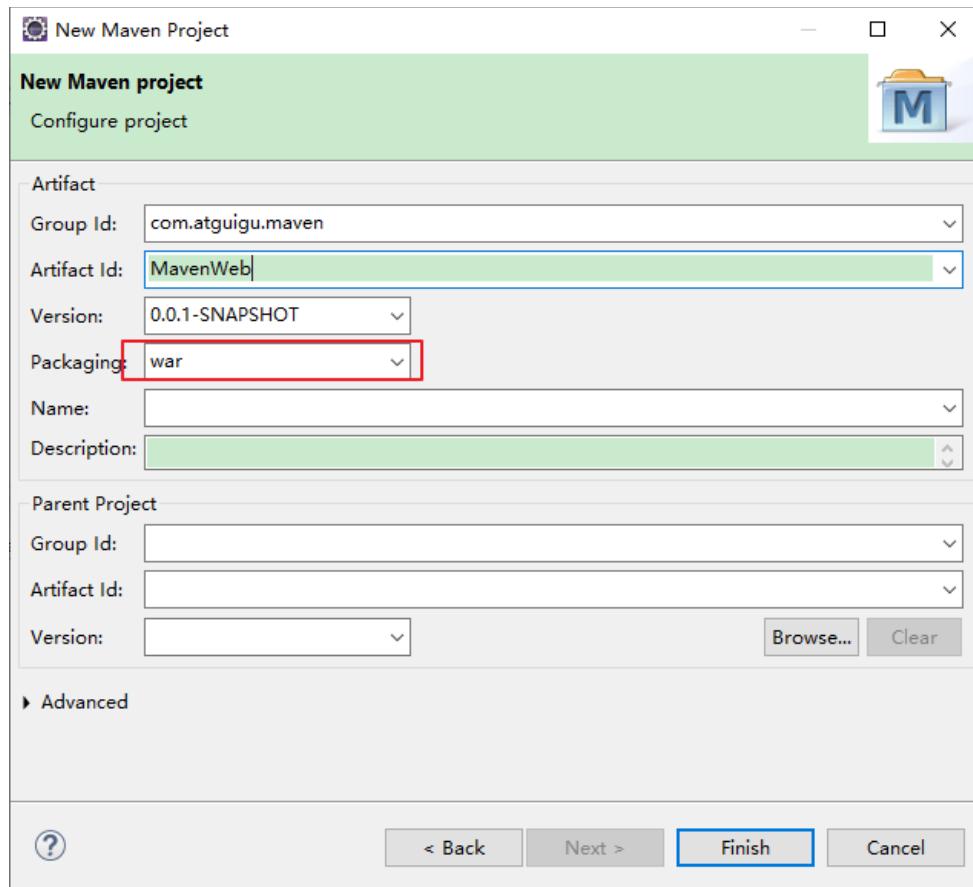
7) 使用 Maven 的方式运行 Maven 工程

在工程名 Hello 或 pom.xml 上右键→Run As 运行 Maven 项目

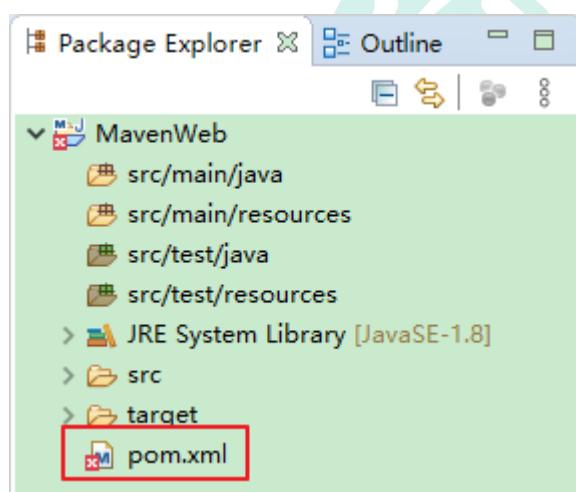


1.4.2 创建 Web 工程（了解）

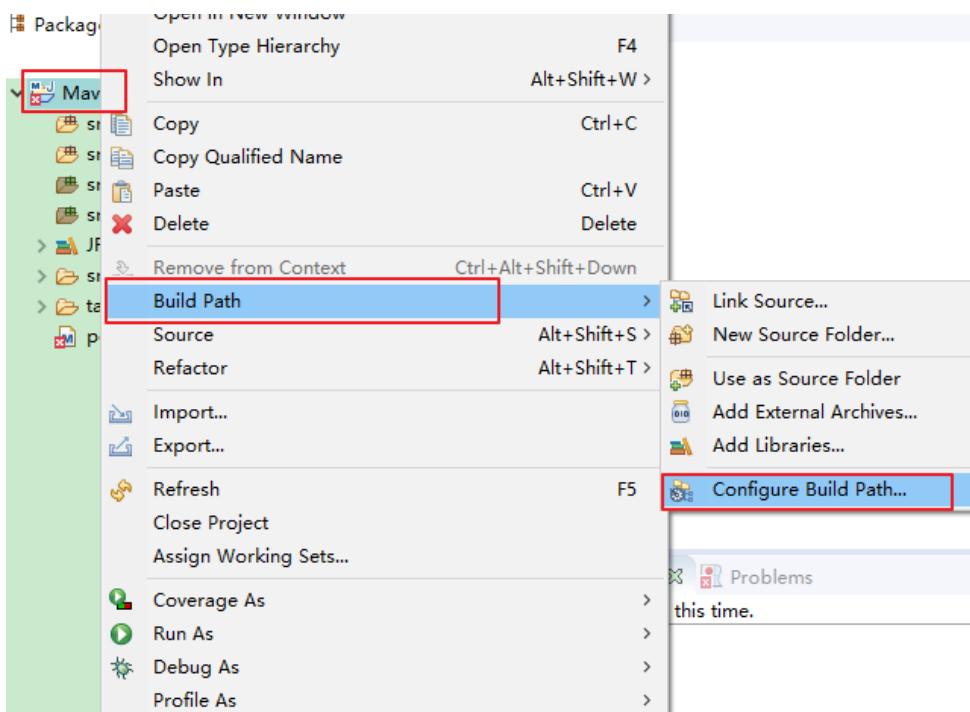
- 1) 创建简单的 Maven 工程，打包方式为 war 包



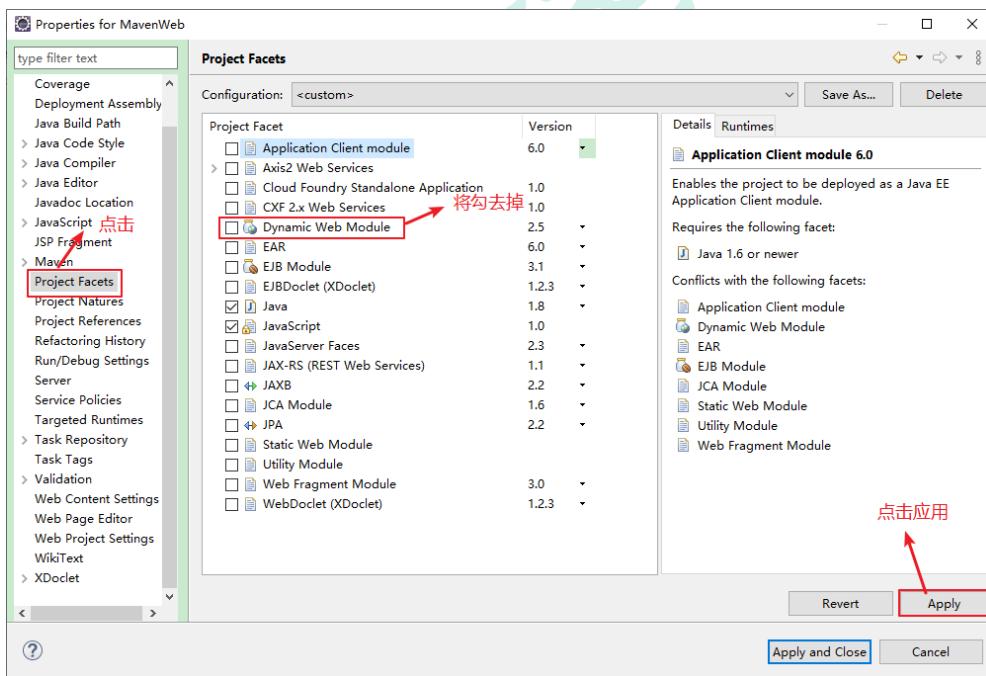
2) 创建完成之后因缺少 web.xml 文件工程出现小红叉



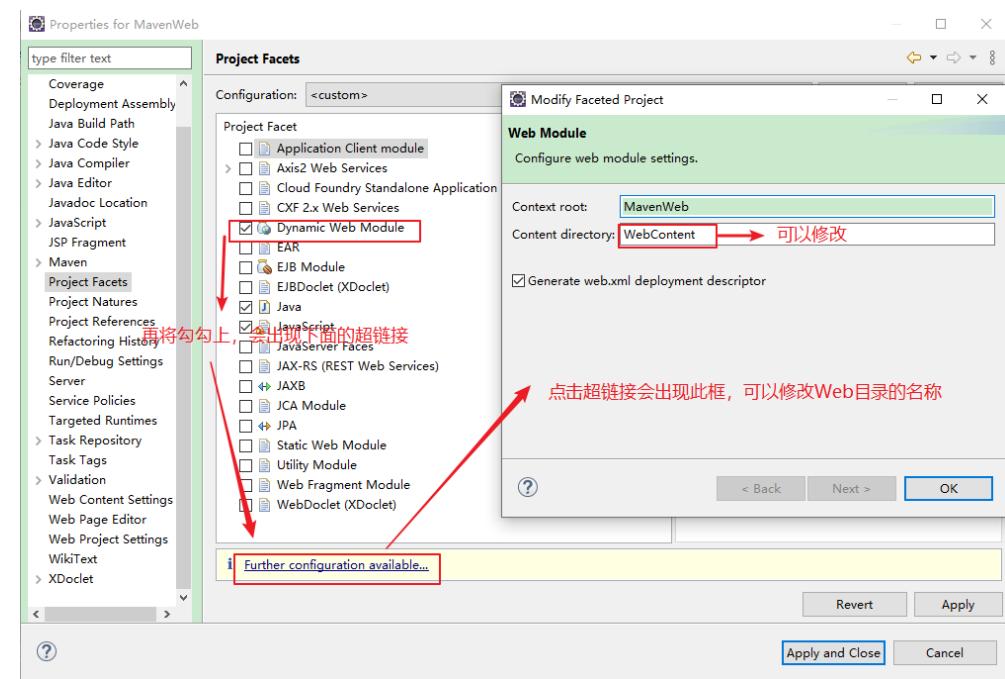
3) 在工程上右键→Build Path→Configure Build Path...



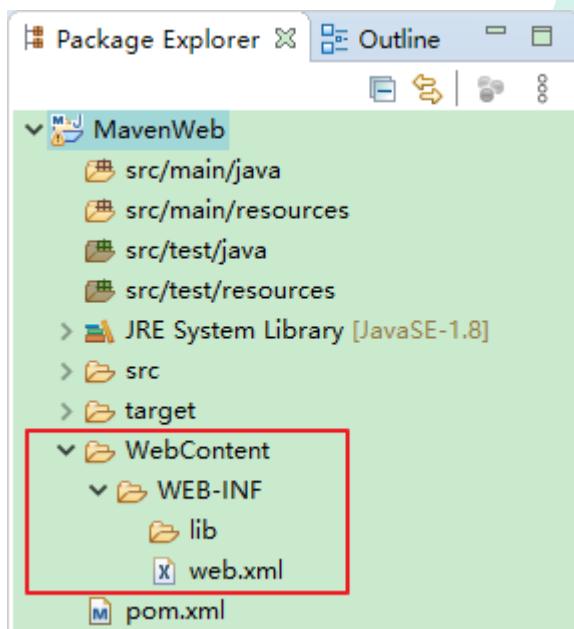
4) 点击 Project Facets 欺骗 Eclipse 当前工程不是 Web 工程，点击应用



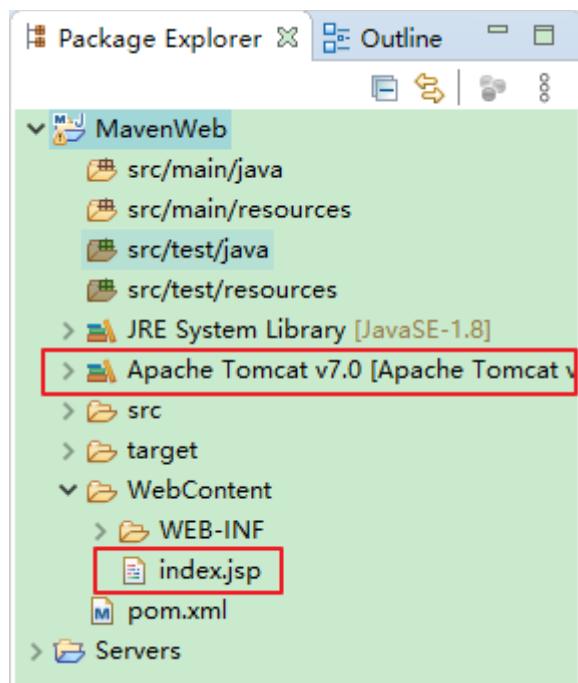
5) 再告诉 Eclipse 当前工程是一个 Web 工程，点击应用并关闭



6) 发现 MavenWeb 工程小红叉消失，并出现了 WebContent 目录



7) 在 WebContent 下创建 index.jsp 页面并添加 Tomcat 库

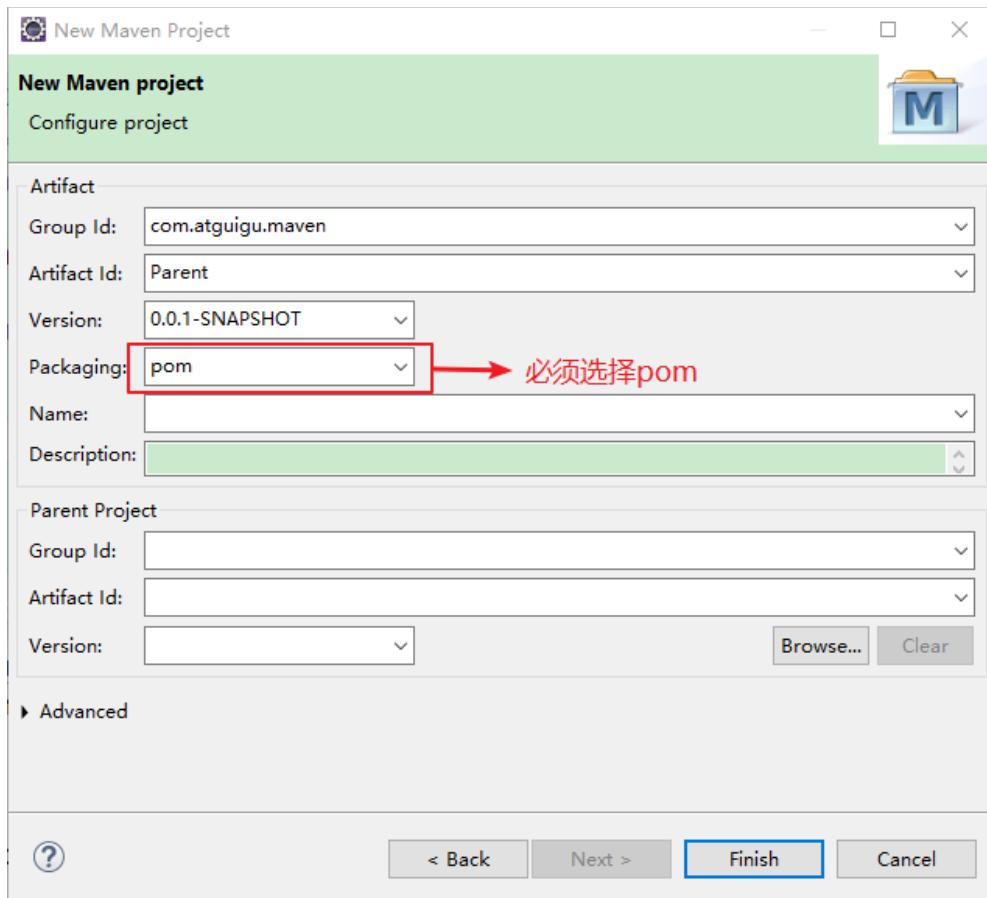


- 8) 在 MavenWeb 上右键→Run As→Run on Server 部署到 Tomcat 上运行

1.4.3 创建父工程

父工程的打包方式为 pom，父工程只需要保留 pom.xml 文件即可

- 1) 创建简单的 Maven 工程，打包方式选择 pom



- 2) 在 pom.xml 文件中通过<dependencyManagement></dependencyManagement>标签进行依赖管理

```
<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
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.atguigu.maven</groupId>
  <artifactId>Parent</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>pom</packaging>

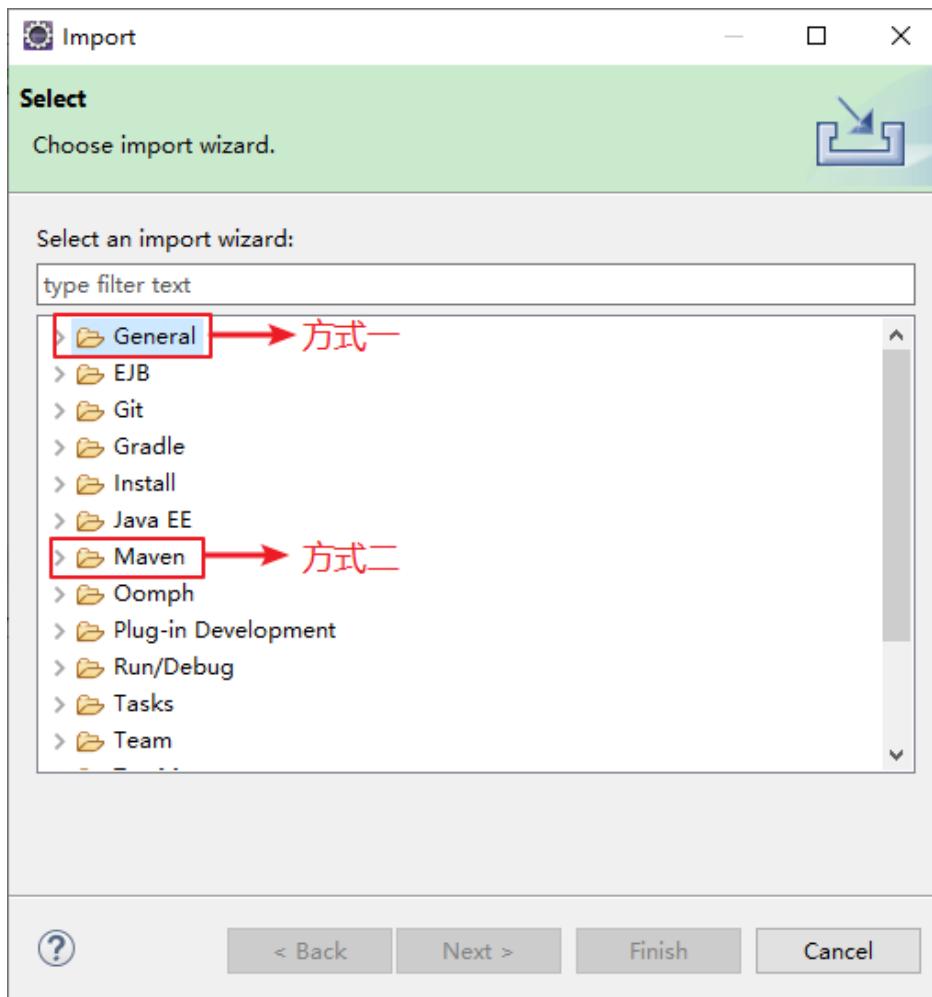
  <!-- 依赖管理 -->
  <dependencyManagement>
    <dependencies>
      <!-- 在此配置要管理的依赖 -->
    </dependencies>
  </dependencyManagement>
</project>
```

- 3) 在子工程中继承父工程

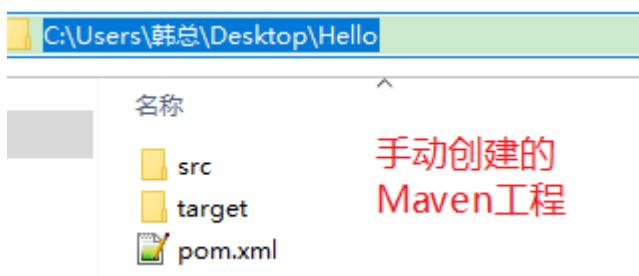
```
<!-- 继承 -->
<parent>
  <!-- 在此配置父工程的坐标 -->
</parent>
```

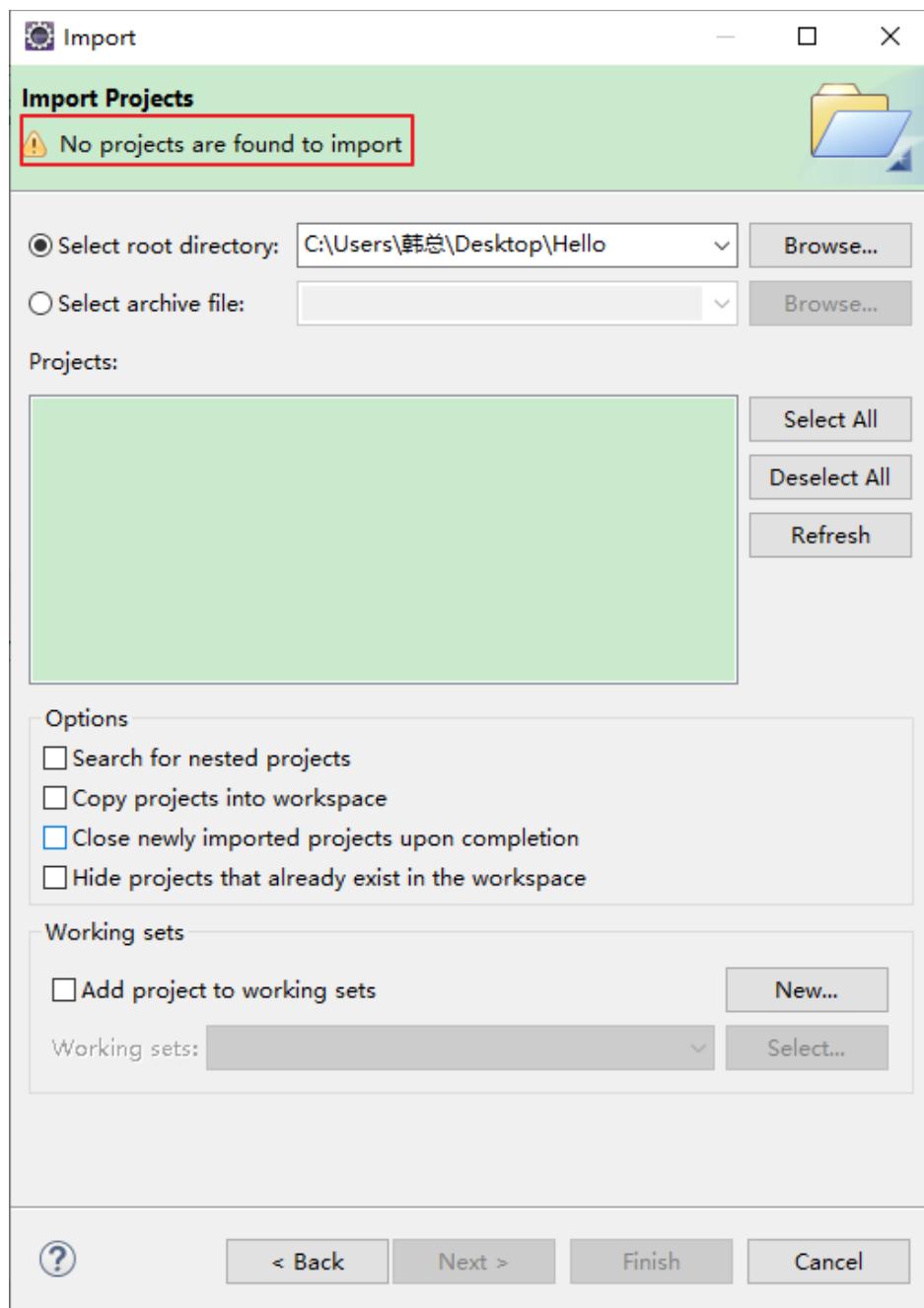
1.5 在 Eclipse 中导入 Maven 项目

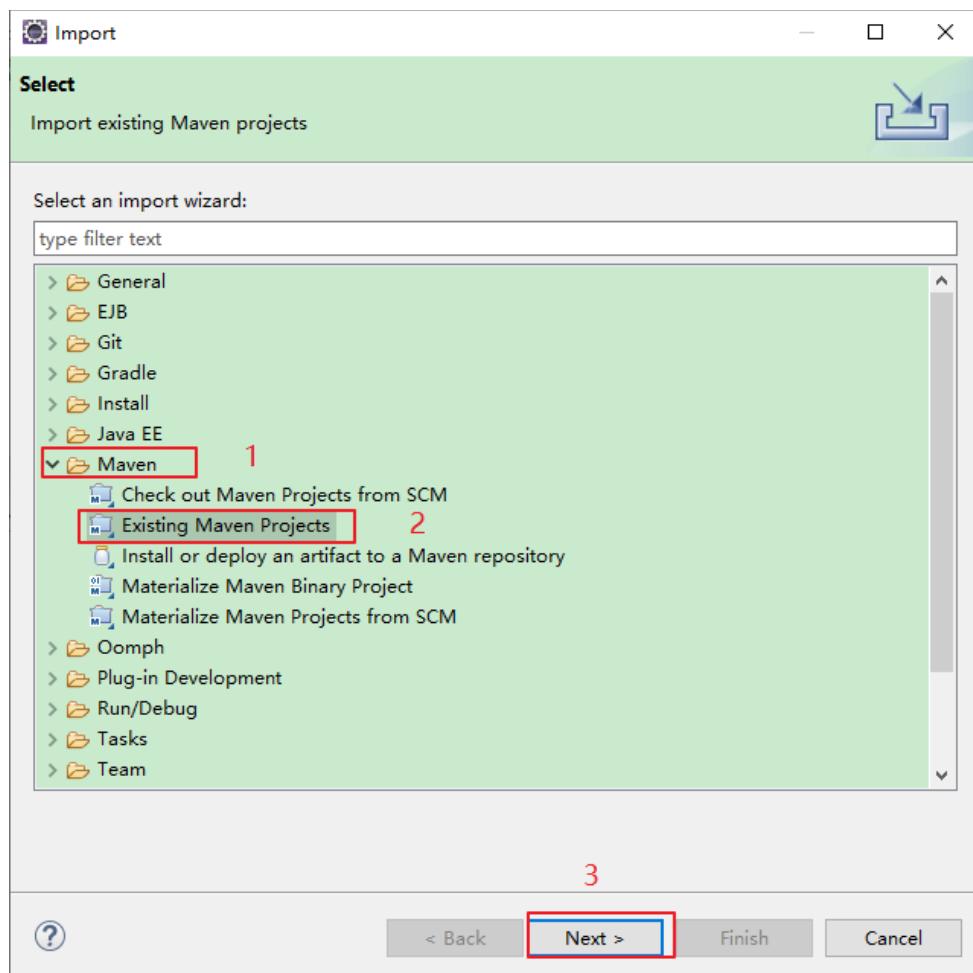
- 1) 点击 File→Import...

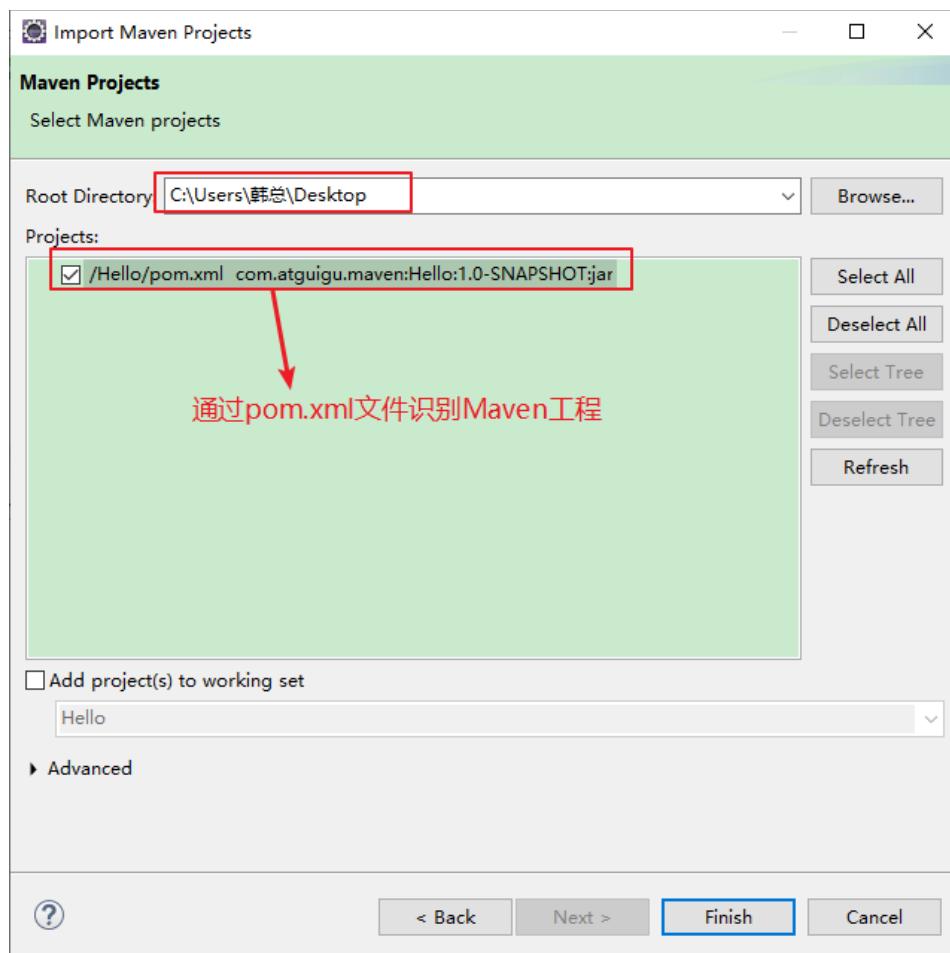


- 2) 第一次导入手动创建的 Maven 项目时，由于项目中没有 Eclipse 生成的一些文件，使用**方式一**导入时 Eclipse 认为它不是一个工程

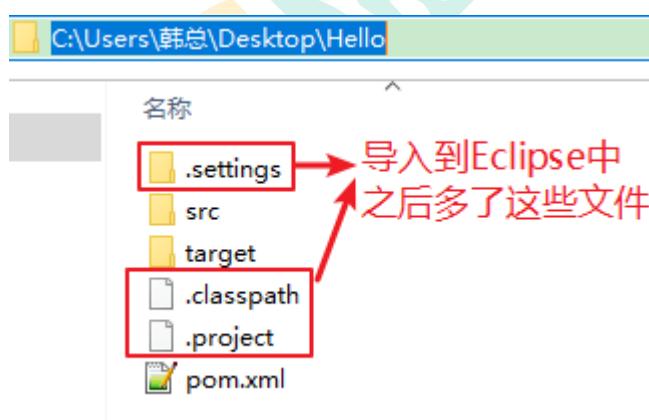








4) 导入到 Eclipse 中之后就会生成一些 Eclipse 能识别的文件



5) 有了这些 Eclipse 能识别的文件之后以后再往 Eclipse 中导入的时候选择方式一和方式二都可以

第 2 章 在 Idea 中使用 Maven

2.1 在 Idea 中配置 Maven

Idea 中也自带 Maven 插件，而且我们也可以给自带的 Maven 插件进行配置，所以我们可以使用自带的 Maven，也可以使用我们安装的 Maven 核心程序

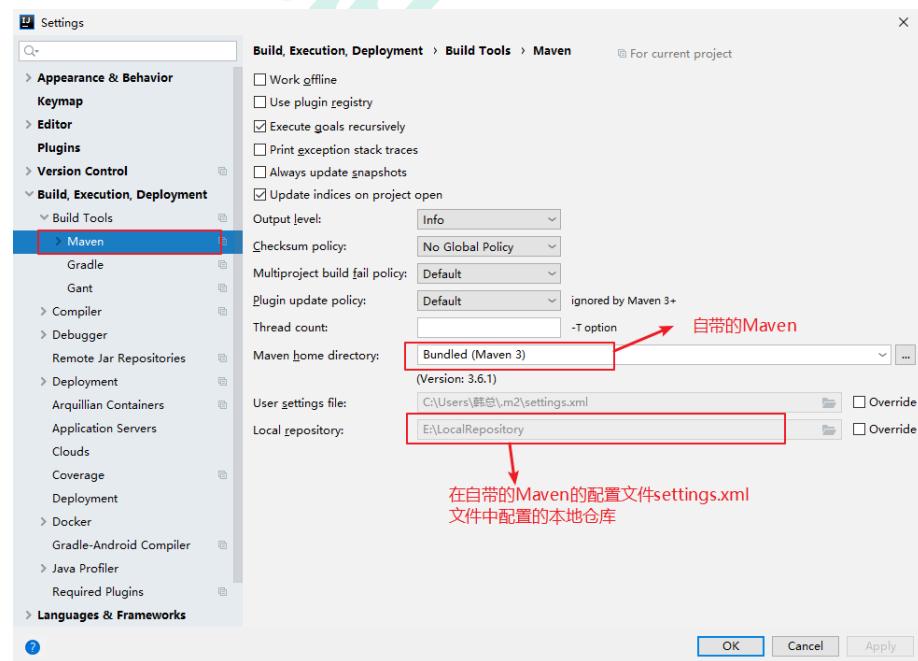
2.1.1 配置自带的 Maven 插件

1) Idea 自带的 Maven 在 Idea 的安装目录的 plugins 目录中

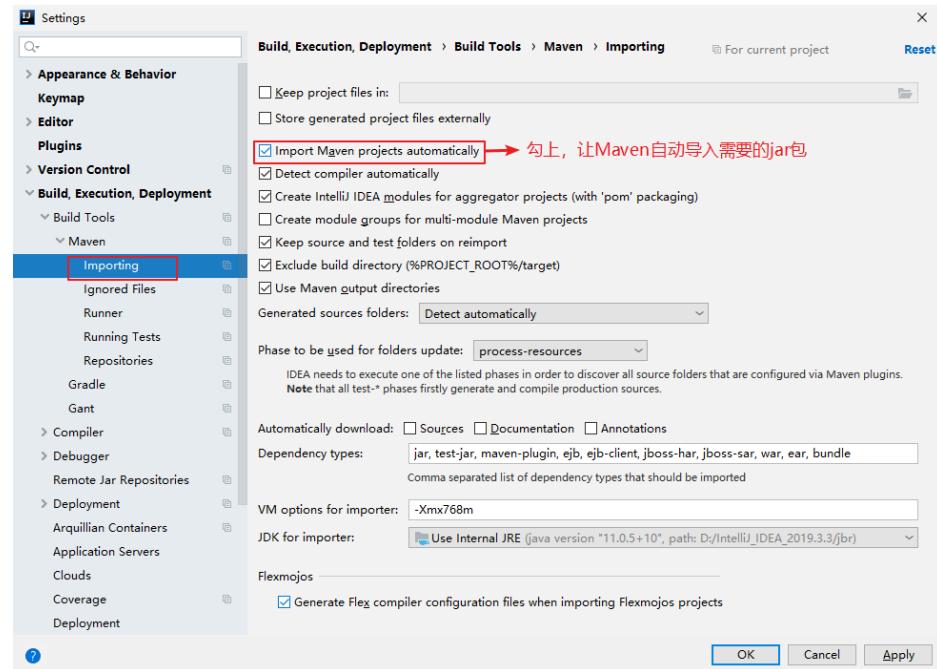


名称	修改日期	类型	大小
bin	2020/4/22 11:11	文件夹	
boot	2020/4/22 11:11	文件夹	
conf	2020/4/22 11:11	文件夹	
lib	2020/4/22 11:11	文件夹	
LICENSE	2020/2/11 12:37	文件	14 KB
NOTICE	2020/2/11 12:37	文件	1 KB
README.txt	2020/2/11 12:37	文本文档	3 KB

2) 在自带的 Maven 里配置了本地仓库之后打开 Idea 之后会发现本地仓库自动变成了我们设置的仓库

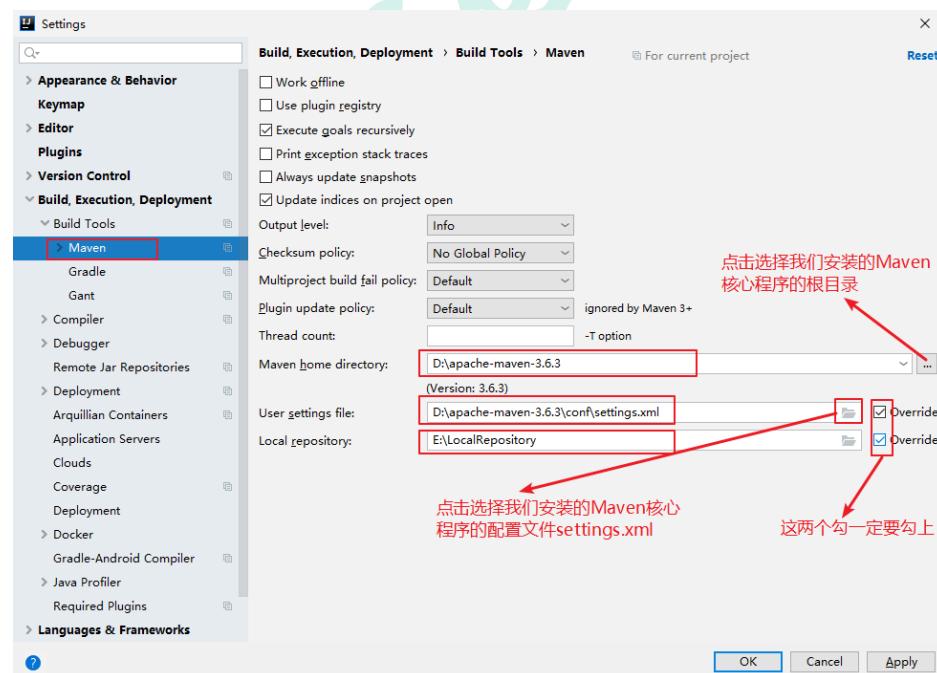


3) 设置 Maven 自动导包



2.1.2 配置我们自己安装的 Maven

1) 点击工具栏中的 Settings

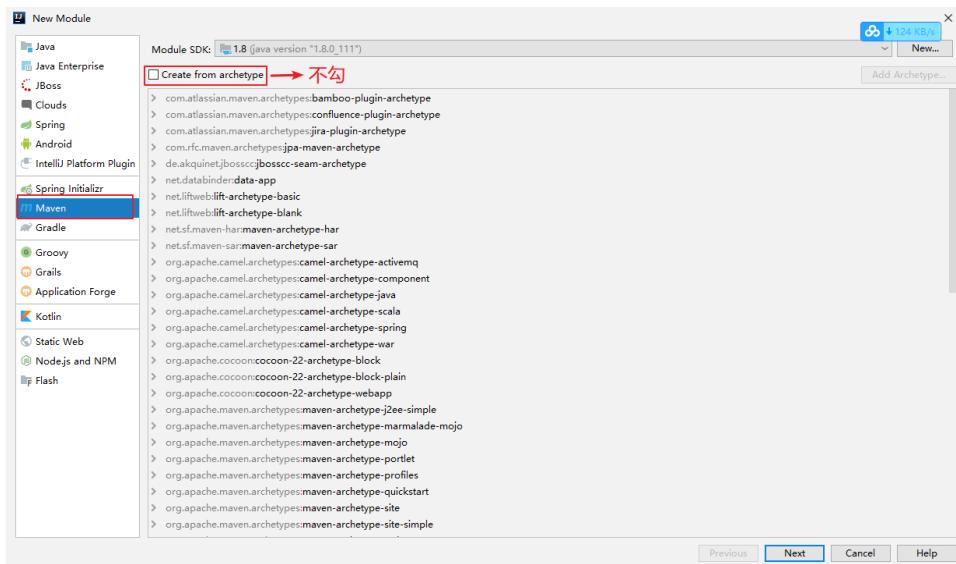


2) 点击 OK 保存即可

2.2 在 Idea 中创建 Maven 项目

2.2.1 创建 Java 工程

- 1) 点击 File→New→Module... (如果之前没有 Project 选 Project) →Maven



- 2) 点击 Next, 配置要继承的模块 (如果直接创建的是 Project 不存在这一项)、坐标 (GAV)、路径。不同的 Idea 版本可能有所差别, 我使用的是 2019.3.3 的版本



- 3) 点击 Finish 即可创建成功

- 4) 配置 Maven 的核心配置文件 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.atguigu.maven</groupId>
<artifactId>Hello</artifactId>
<version>1.0-SNAPSHOT</version>

<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
</dependencies>

</project>
```

5) 编写主代码

在 src/main/java 目录下创建包并创建 Hello.java 文件

```
package com.atguigu.maven;

public class Hello {
    public String sayHello(String name){
        return "Hello "+name+"!";
    }
}
```

6) 编写测试代码

在 src/test/java 目录下创建包并创建 HelloTest.java 文件

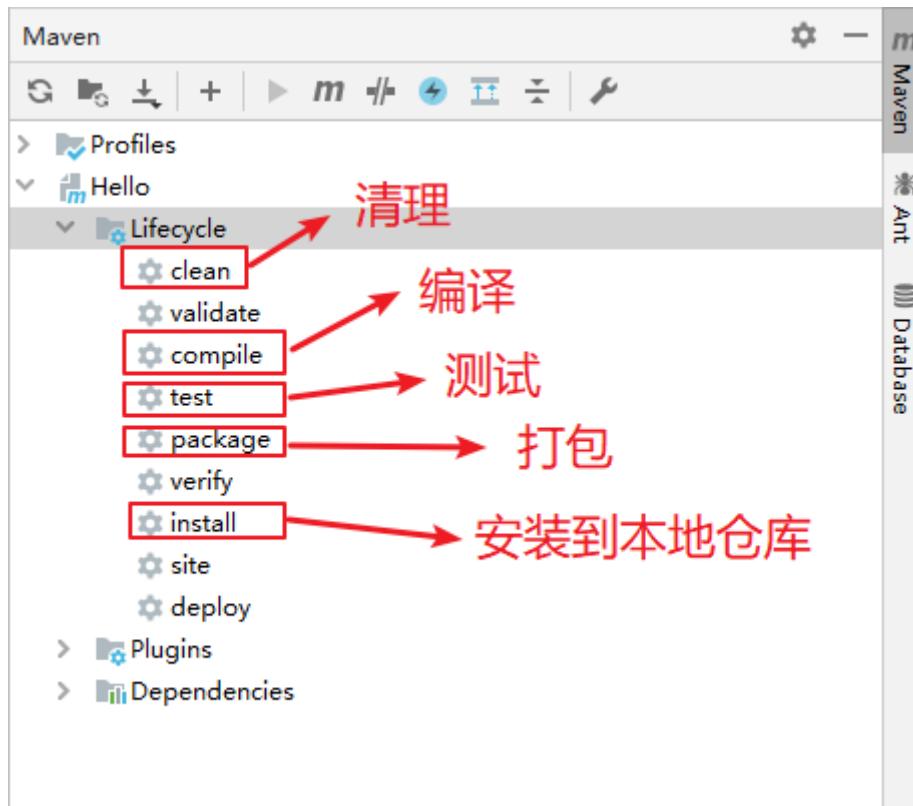
```
package com.atguigu.maven;

import org.junit.Test;

public class HelloTest {

    @Test
    public void testHello(){
        Hello hello = new Hello();
        String maven = hello.sayHello("Maven");
        System.out.println(maven);
    }
}
```

7) 使用 Maven 的方式运行 Maven 工程

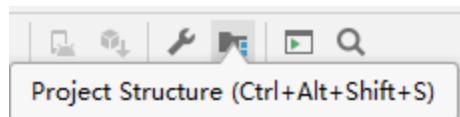


2.2.2 创建 Web 工程（了解）

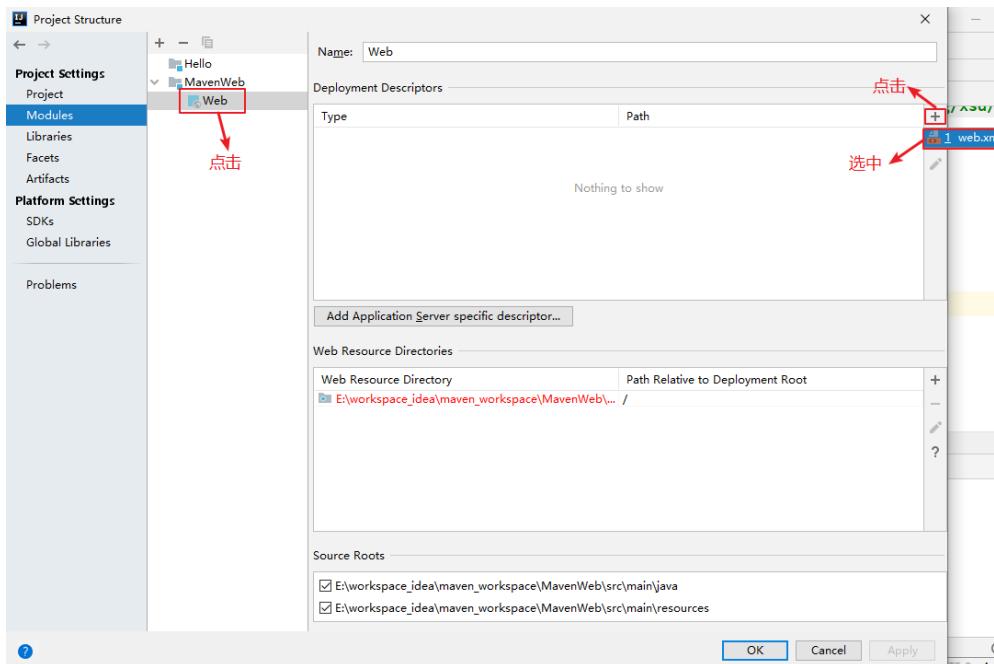
1) 创建简单的 Maven 工程，打包方式为 war 包

```
<groupId>com.atguigu.maven</groupId>
<artifactId>MavenWeb</artifactId>
<packaging>war</packaging>
<version>1.0-SNAPSHOT</version>
```

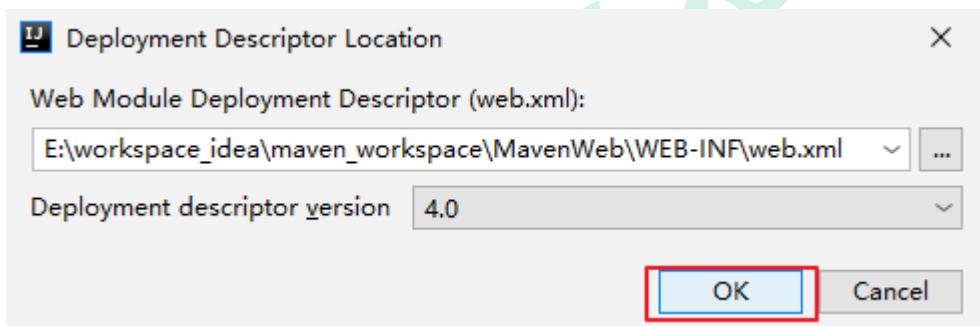
2) 点击 Project Structure



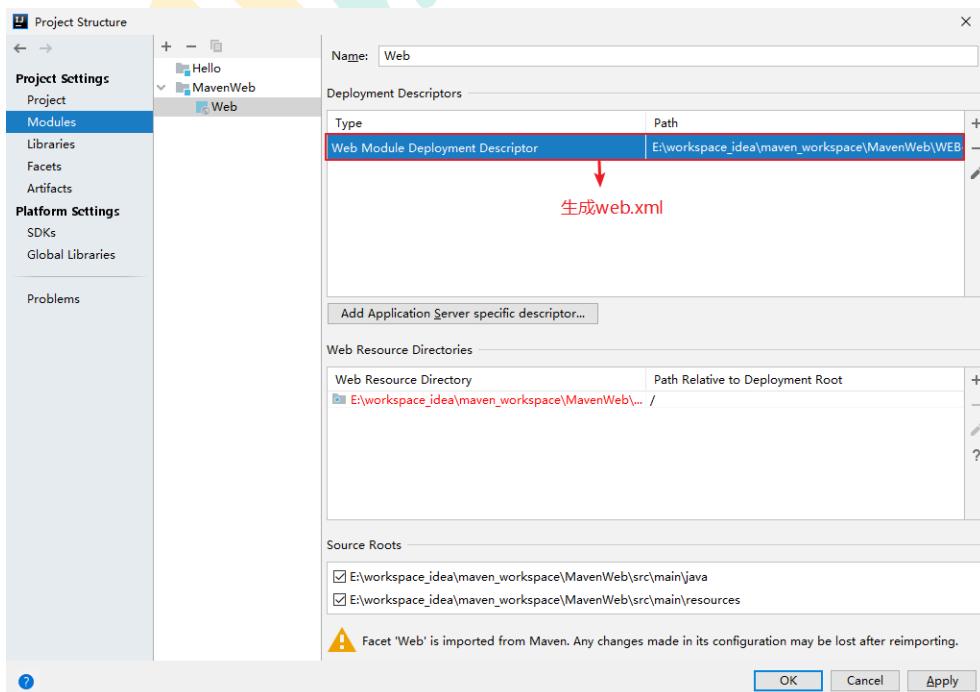
3) 选择对应的 Module，设置 Web 目录



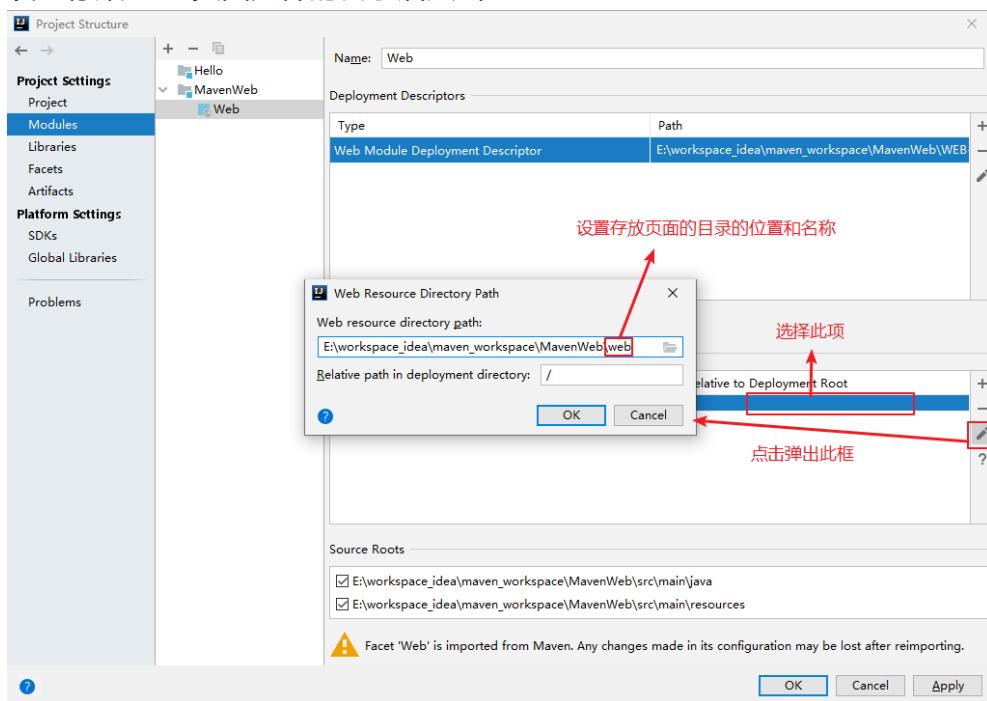
4) 弹出提示框，选择版本后点击 OK



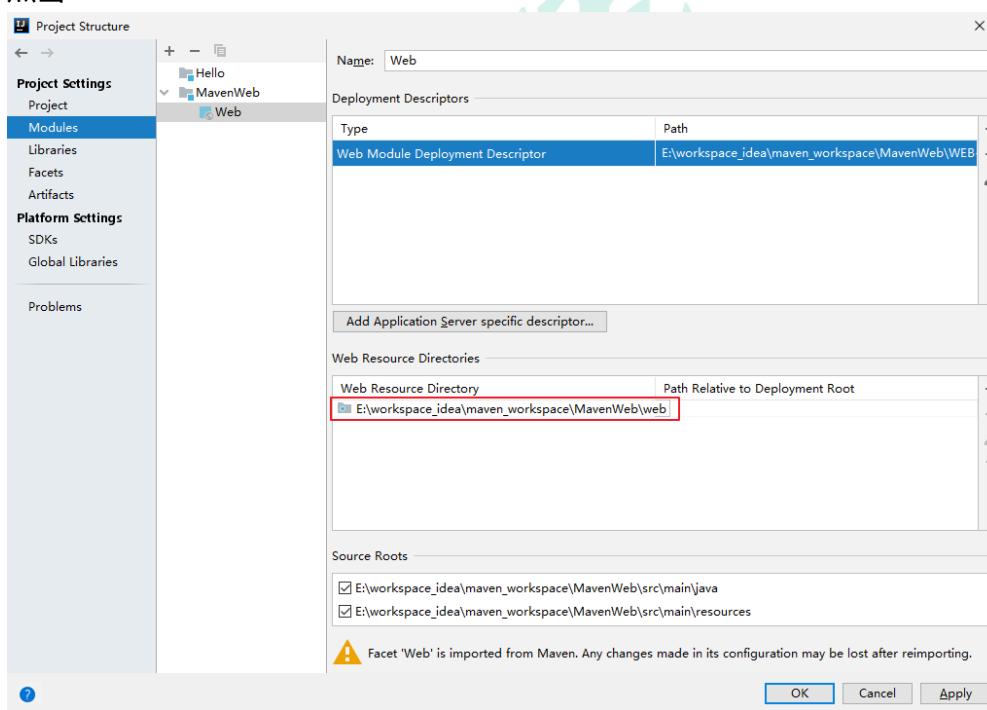
5) 生成 web.xml 文件



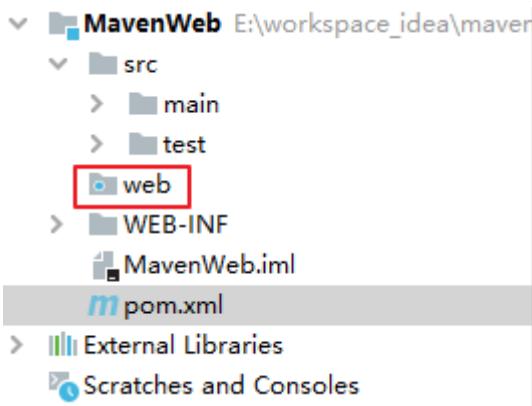
6) 设置存放 web 页面文件的目录后点击 OK



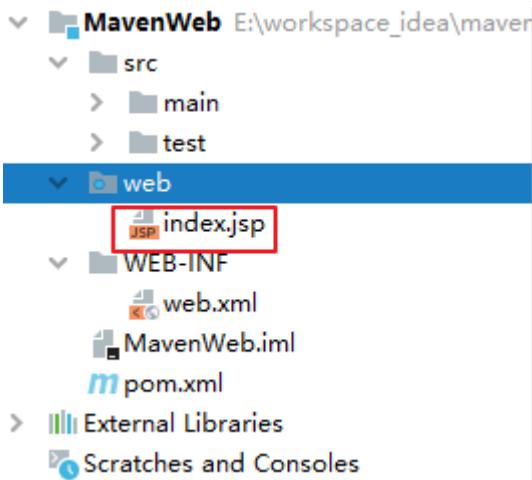
7) 点击 OK



8) 发现项目中多了一个 web 目录，而且目录上有一个蓝点



9) 在 web 目录下创建 index.jsp 页面



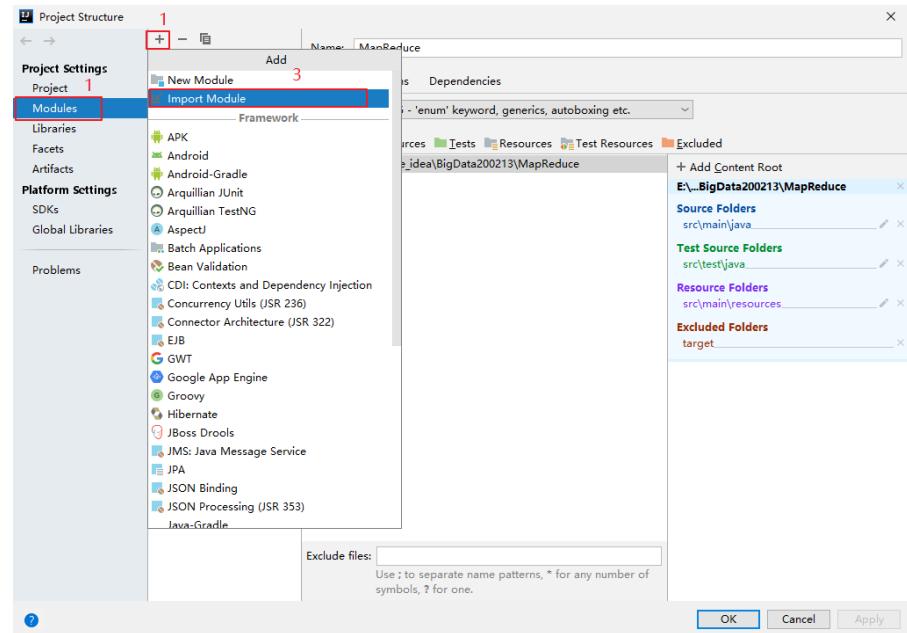
10) 部署到 Tomcat 上运行

2.3 在 Idea 中导入 Maven 项目

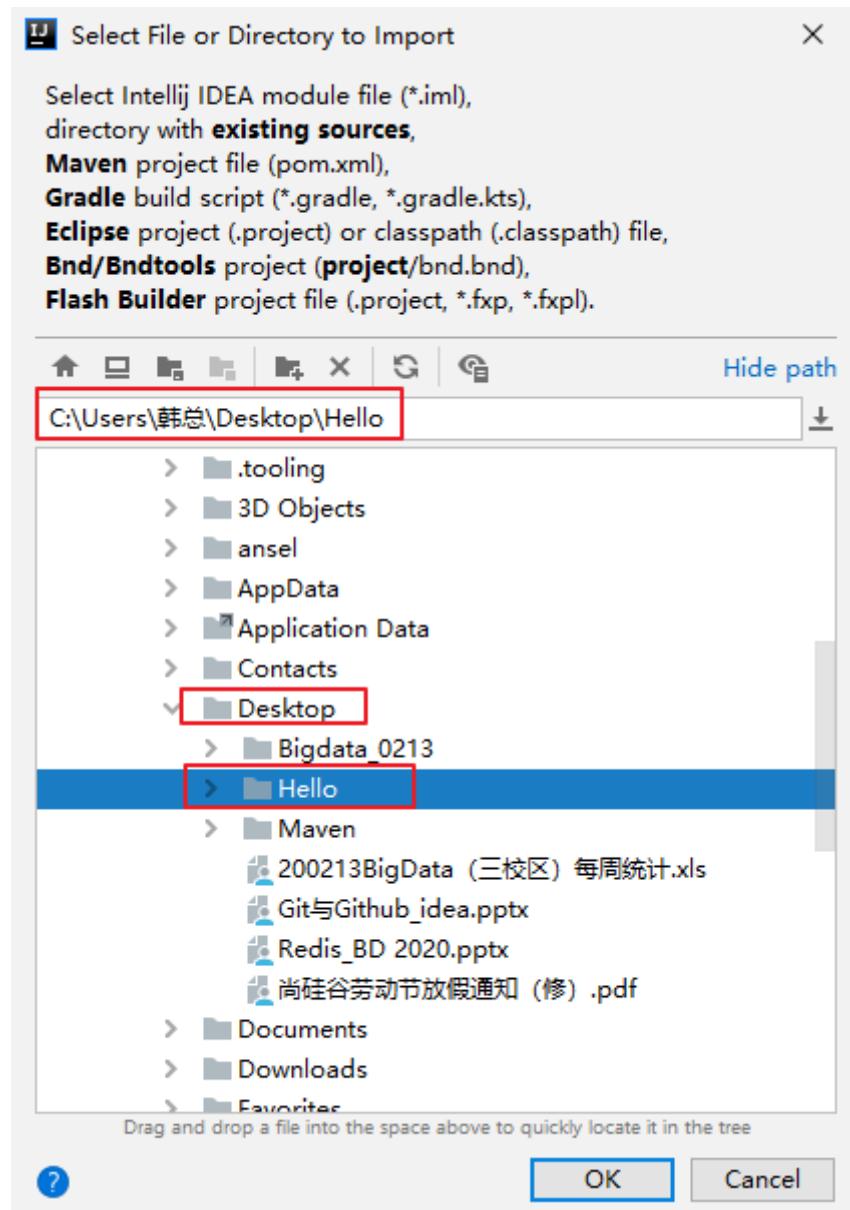
1) 点击 Project Structure



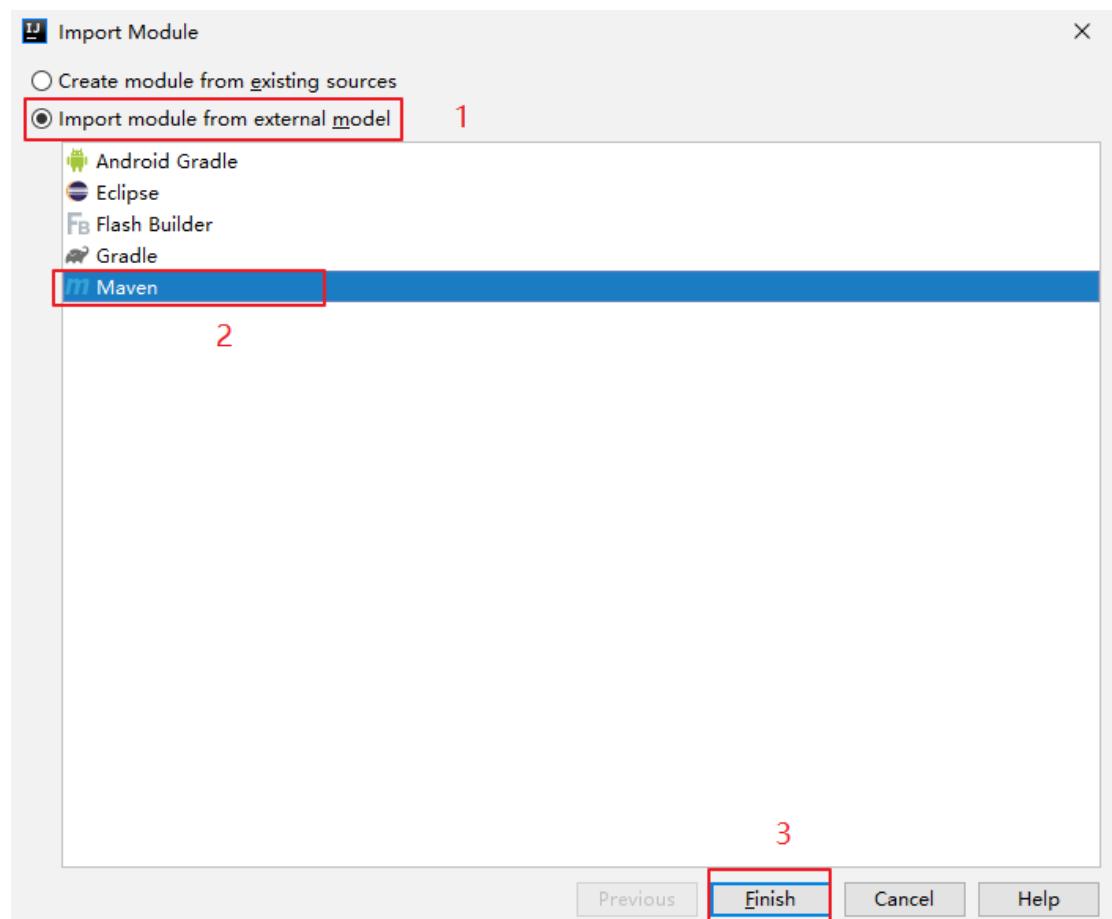
2) 点击 Modules → + → Import Module



3) 找到项目所在的位置



- 4) 选择 Import module from external model (从外部模型导入模块) → Maven
→ Finish



第3章 在 Eclipse 中使用 Git

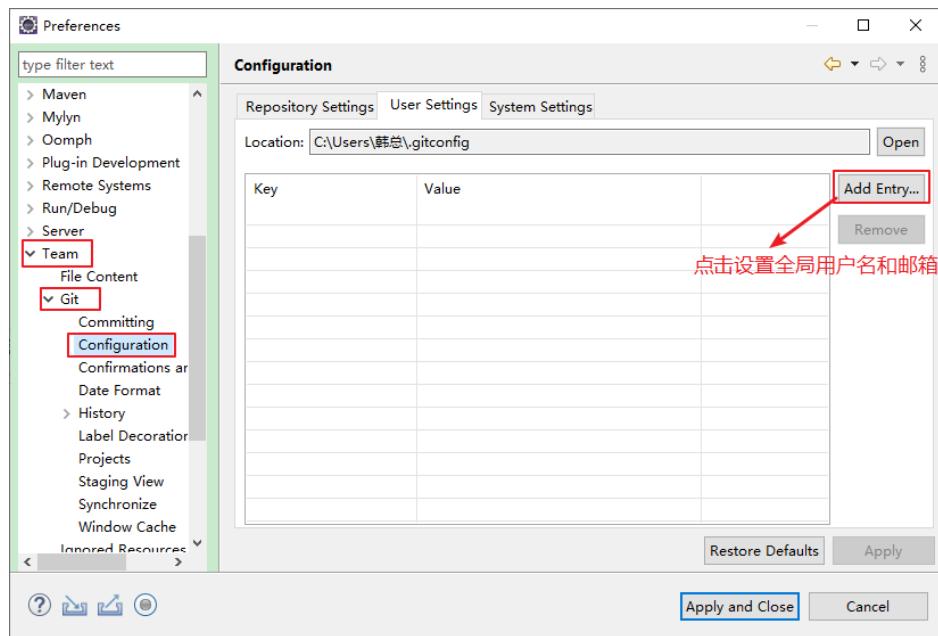
Eclipse 中默认自带了 Git 插件，通过点击 Help→About Eclipse IDE 可以查看



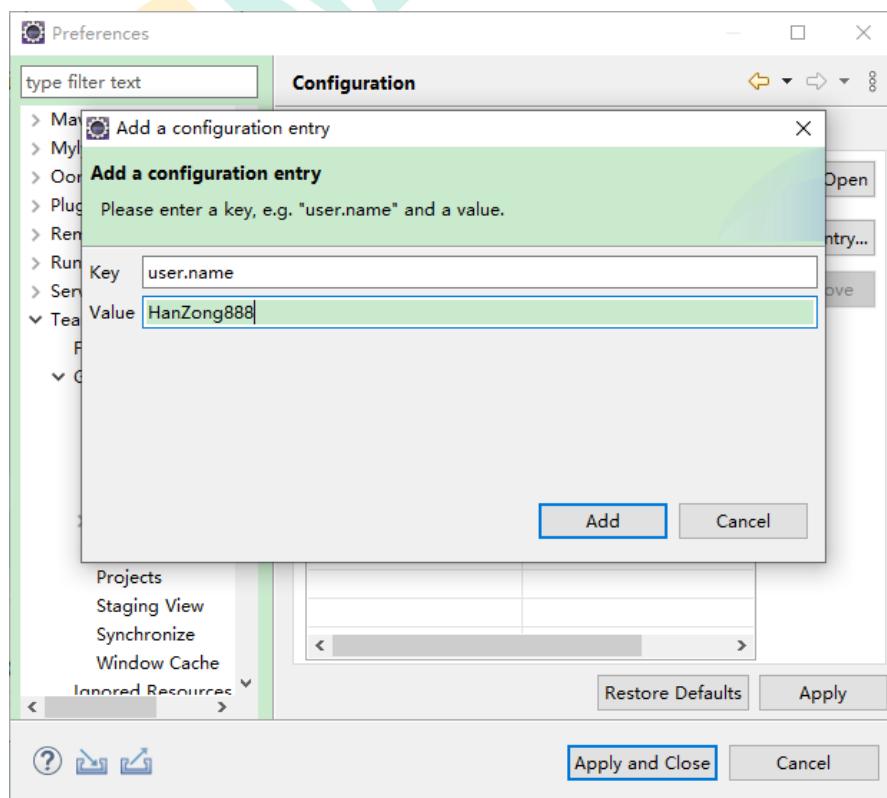
3.1 全局配置

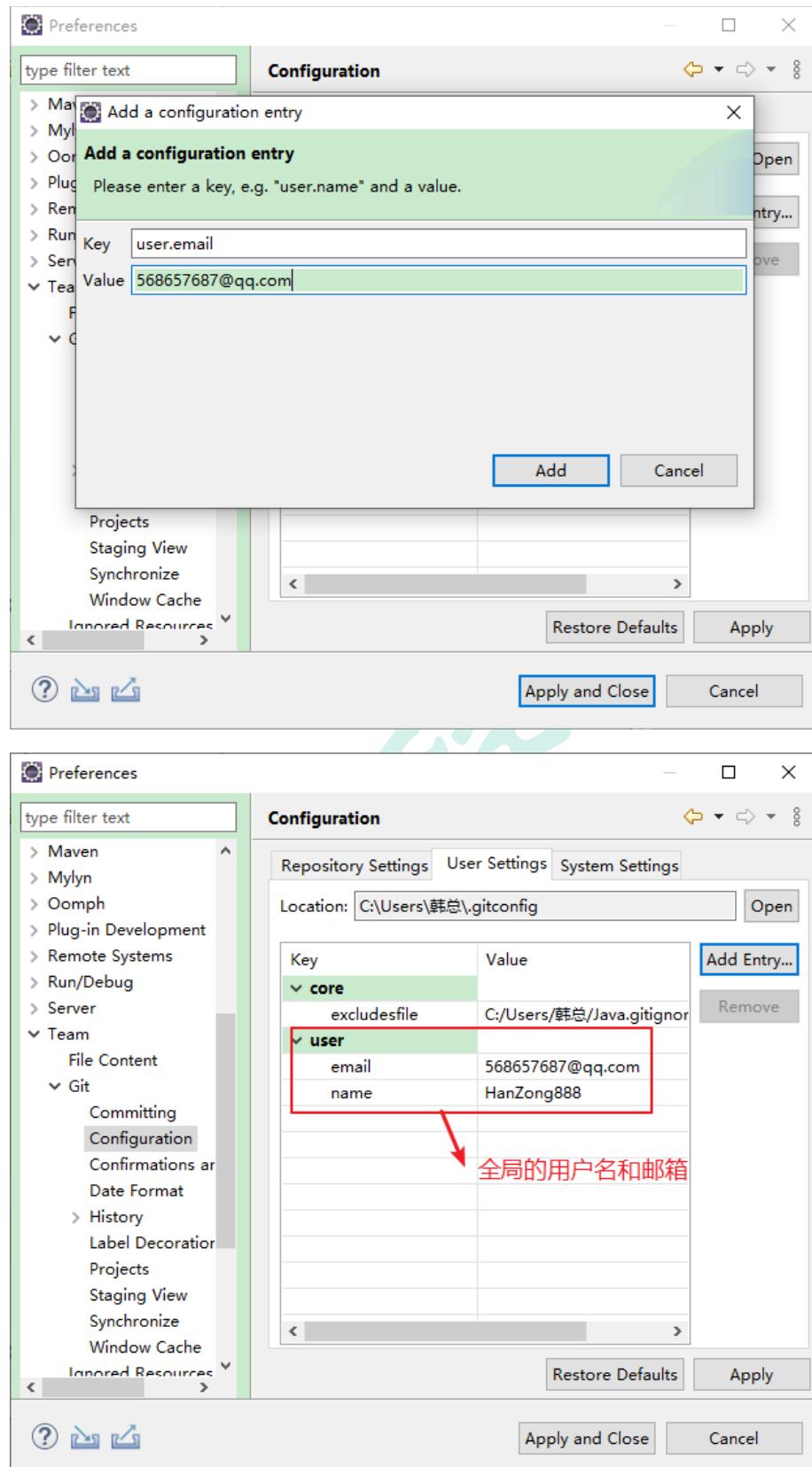
3.1.1 配置用户名和邮箱

- 1) 点击 Window→Preferences→Team→Git→Configuration



- 2) 点击 Add Entry...设置全局用户名和邮箱





- 3) 点击 Apply and Close 之后在 Windows 的用户目录下会生成.gitconfig 配置文件

此电脑 > Windows-SSD (C:) > 用户 > 韩总 >				
名称	修改日期	类型	大小	
.android	2020/4/22 11:12	文件夹		
.config	2020/5/4 13:18	文件夹		
.eclipse	2020/4/23 16:45	文件夹		
.IntelliJIdea2019.3	2020/4/22 11:11	文件夹		
.lsp4xml	2020/4/24 15:19	文件夹		
.m2	2020/4/24 10:41	文件夹		
.p2	2020/5/4 13:26	文件夹		
.tooling	2020/4/23 16:44	文件夹		
3D 对象	2019/12/18 10:38	文件夹		
ansel	2020/2/6 18:52	文件夹		
AppData	2019/10/25 18:59	文件夹		
Desktop	2020/5/4 11:07	文件夹		
MicrosoftEdgeBackups	2019/10/25 19:11	文件夹		
OneDrive	2020/5/4 13:05	文件夹		
Yinxing Biji	2020/4/22 11:04	文件夹		
保存的游戏	2019/12/18 10:38	文件夹		
联系人	2019/12/18 10:38	文件夹		
链接	2019/12/18 10:38	文件夹		
视频	2020/5/4 13:03	文件夹		
收藏夹	2019/12/18 10:38	文件夹		
搜索	2019/12/18 10:38	文件夹		
图片	2019/12/18 10:38	文件夹		
文档	2020/3/8 15:35	文件夹		
下载	2020/5/4 11:20	文件夹		
音乐	2019/12/18 10:38	文件夹		
.gitconfig	2020/5/4 13:34	GITCONFIG 文件	1 KB	
.minttyrc	2020/4/29 14:55	MINTTYRC 文件	1 KB	
.viminfo	2020/4/29 15:16	VIMINFO 文件	11 KB	

3.1.2 配置忽略的文件

- 在用户目录（其他目录也可以）创建 .gitignore 文件，添加以下内容

```
# Compiled class file
*.class

# Log file
*.log

# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

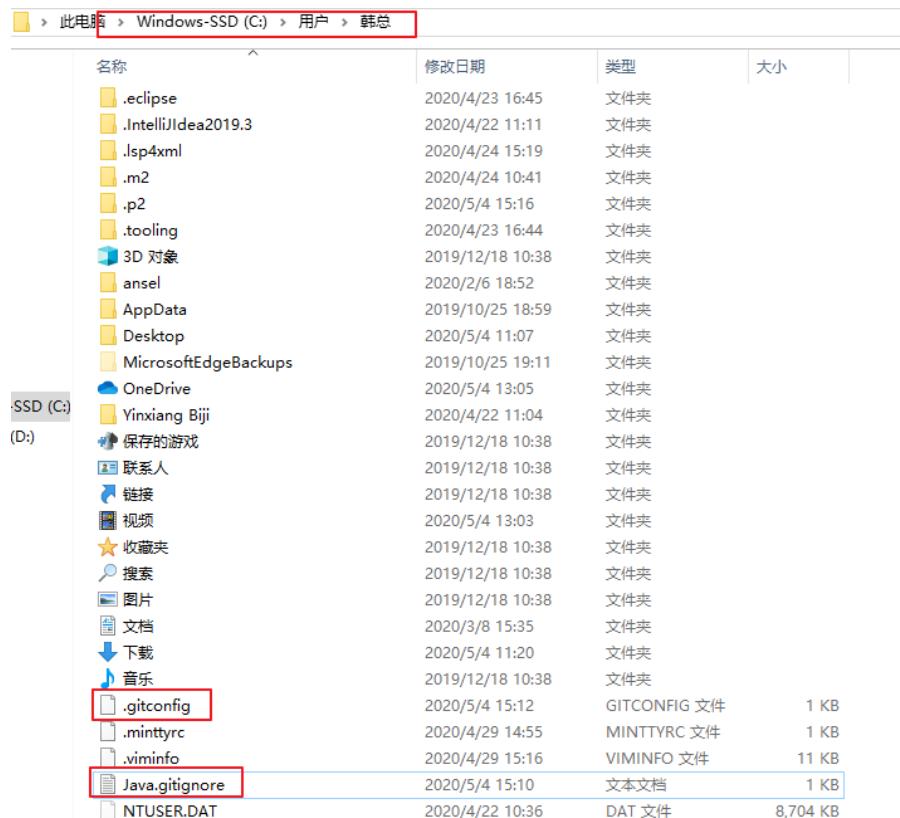
# virtual machine crash logs, see
http://www.java.com/en/download/help/error\_hotspot.xml
hs_err_pid*
```

```
.classpath  
.project  
.settings  
target
```

2) 在全局的配置文件.gitconfig 文件中添加如下内容

```
[core]  
excludesfile = C:/Users/韩总/Java.gitignore
```

3) 文件所在位置图

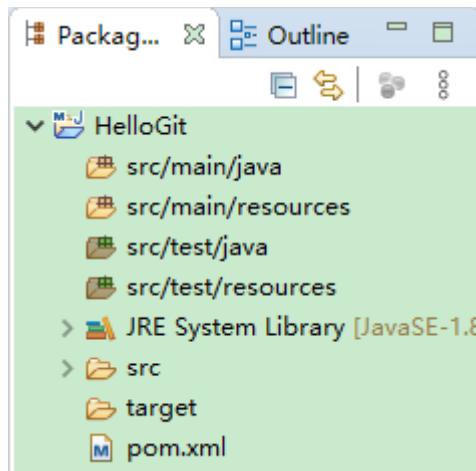


4) 重启 Eclipse 忽略文件 Java.gitignore 即生效

3.2 创建本地库

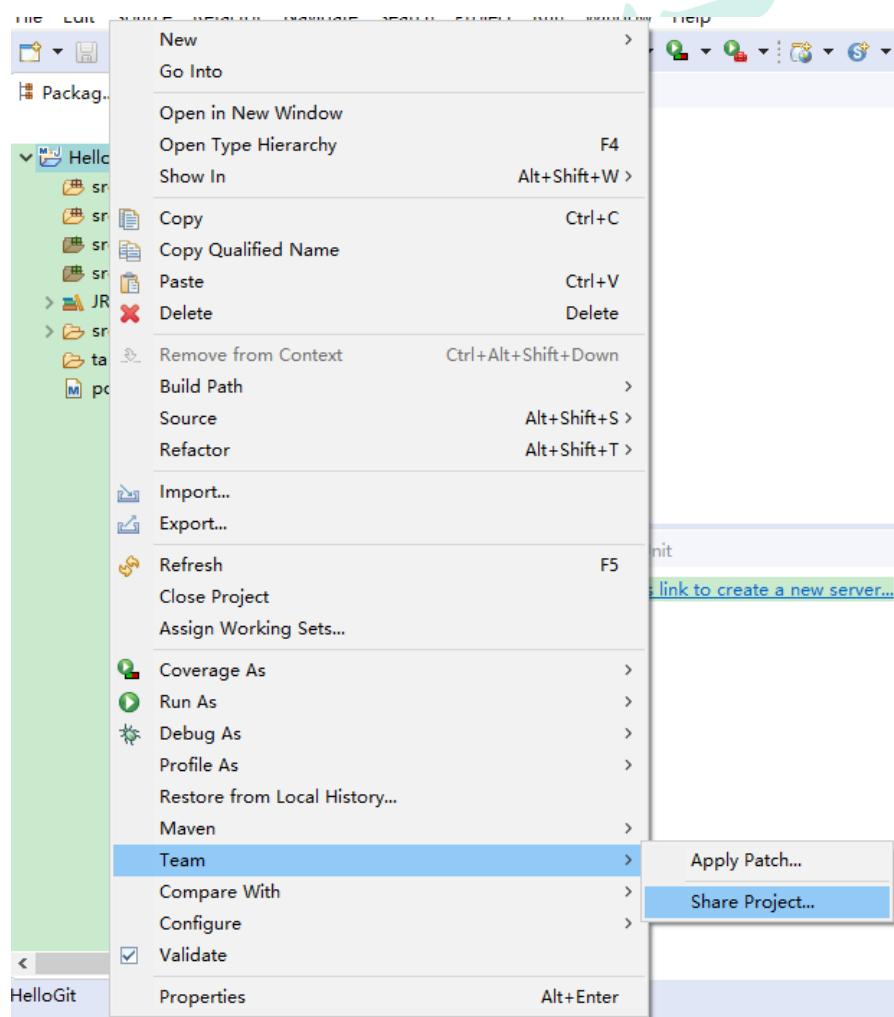
3.2.1 新建本地库

1) 创建一个普通的 Maven 工程

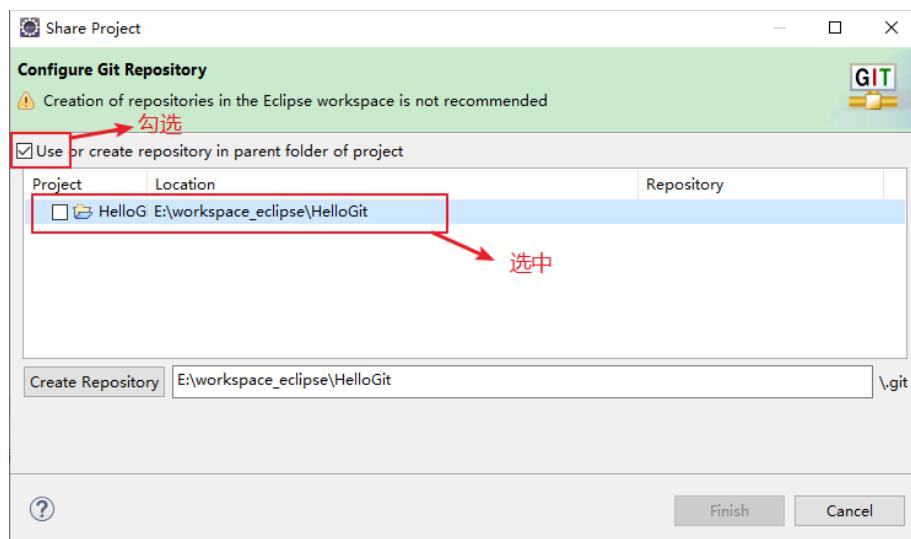


2) 将 Maven 工程交给 Git 管理，即生成.git 目录

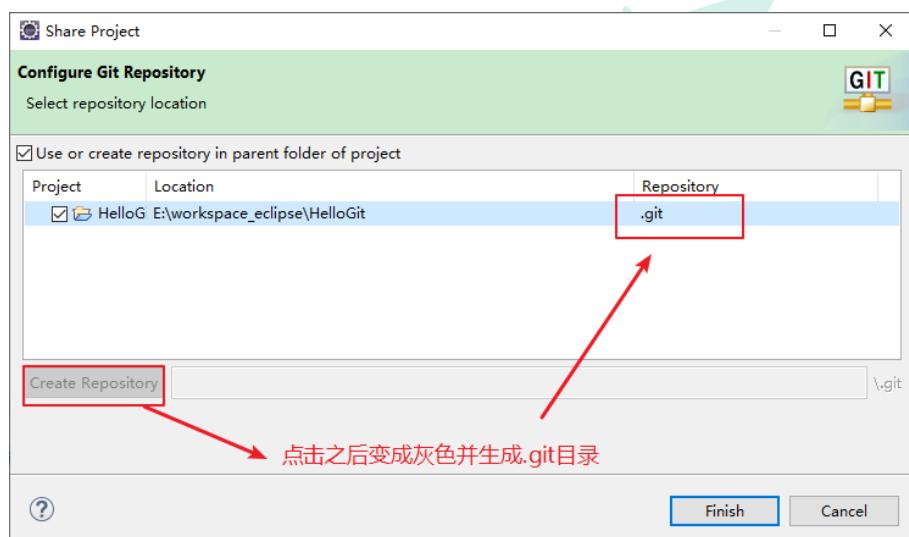
在工程上右键→Team→Share Project...



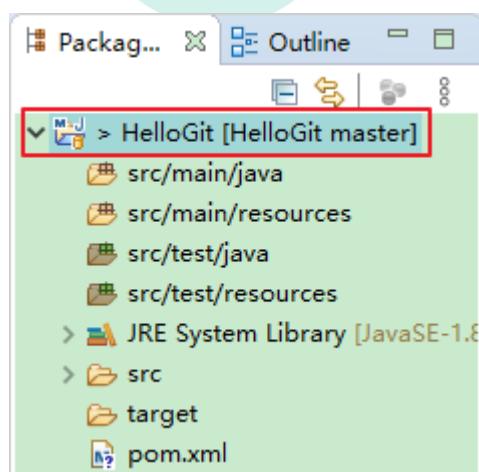
3) 勾选 Use or create repository in parent folder of project 并选中工程



4) 点击 Create Repository 按钮生成.git 目录

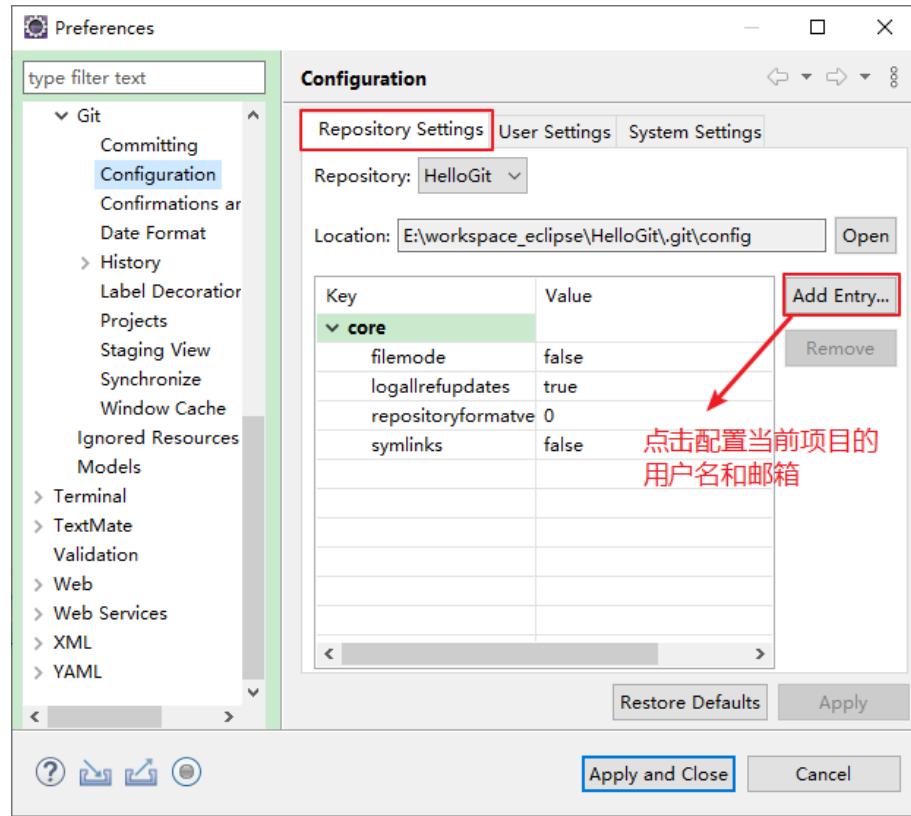


5) 点击 Finish 之后发现工程已被 Git 管理

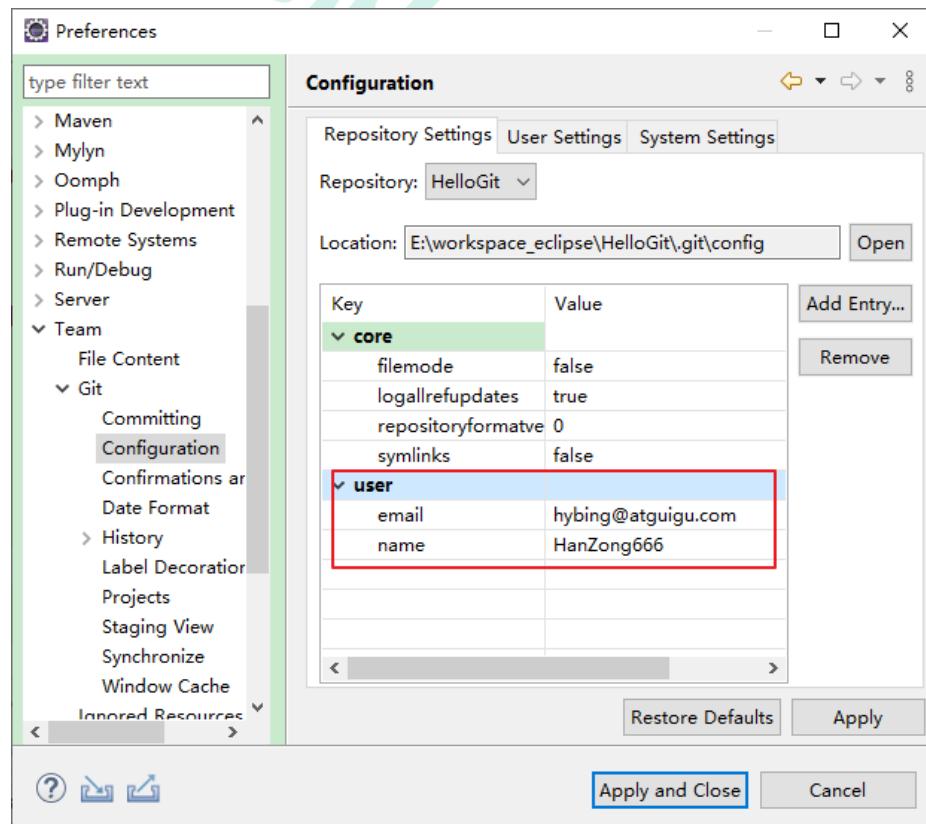


6) 可以配置当前工程的用户名和邮箱

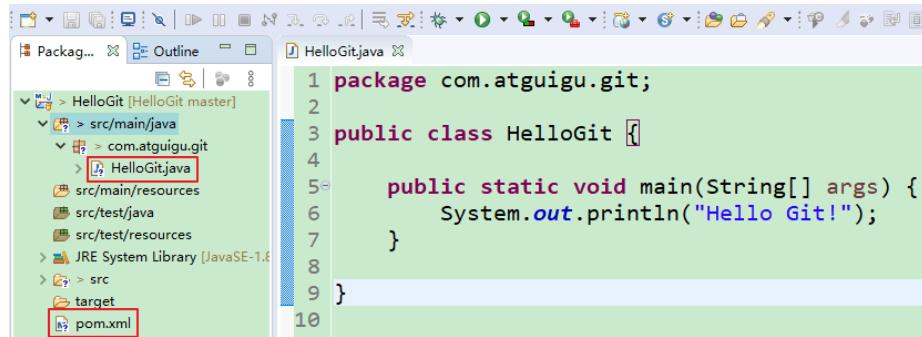
点击 Window→Preferences→Team→Git→Configuration→Repository Settings



7) 点击 Add Entry...配置当前工程的用户名和邮箱



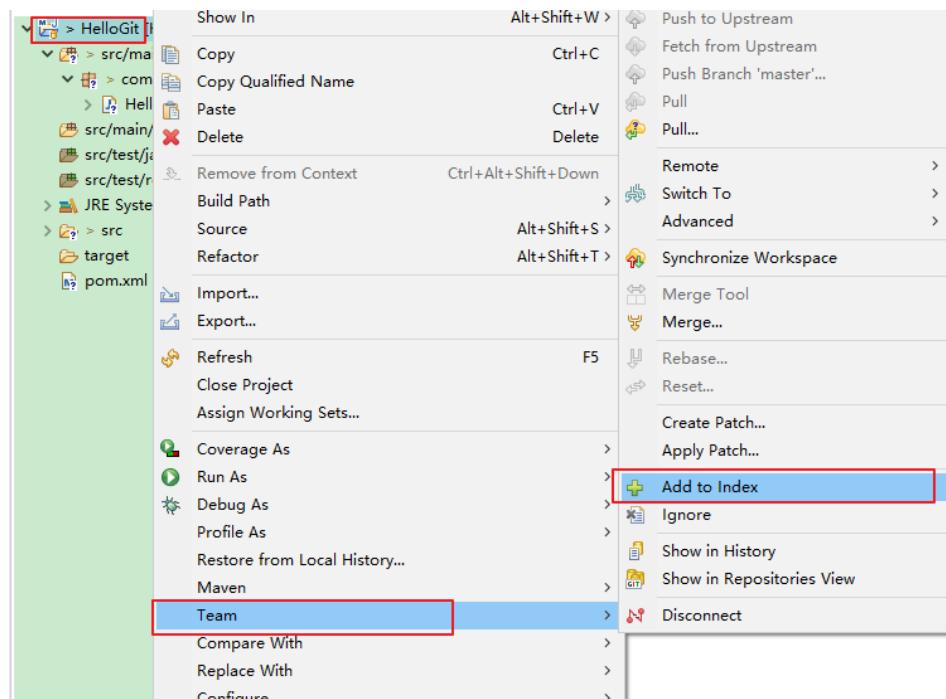
- 8) 在 src/main/java 目录下创建包并创建 HelloGit.java 文件，此时文件只存在于工作区，文件的状态如下图：



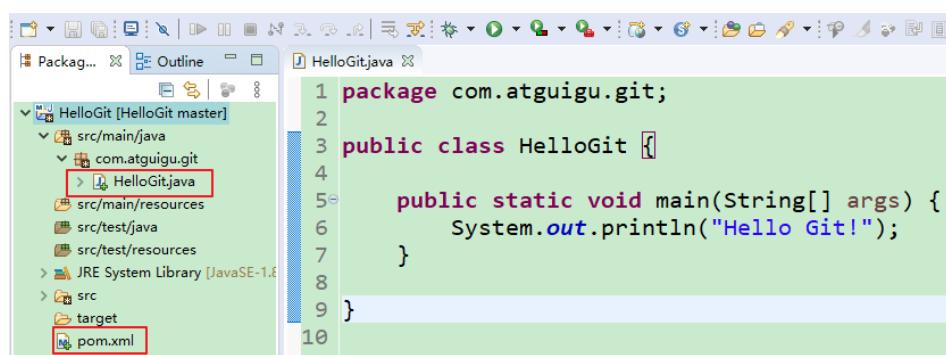
The screenshot shows the Eclipse IDE interface. On the left is the Package Explorer view, which displays the project structure: HelloGit [HelloGit master] > src/main/java > com.atguigu.git > HelloGit.java. The file HelloGit.java is highlighted with a red box. On the right is the Outline view, showing the code of the HelloGit.java file:

```
1 package com.atguigu.git;
2
3 public class HelloGit {
4
5     public static void main(String[] args) {
6         System.out.println("Hello Git!");
7     }
8 }
9
10 }
```

- 9) 在工程上右键→Team→Add to Index 将工程添加到暂存区

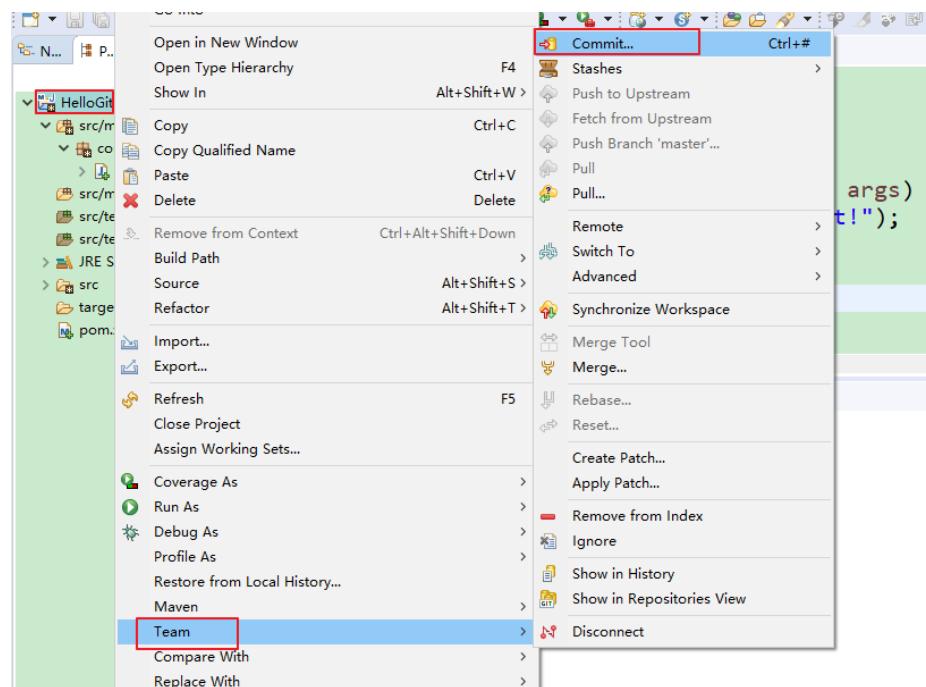


- 10) 添加到暂存区之后文件的状态如下图：

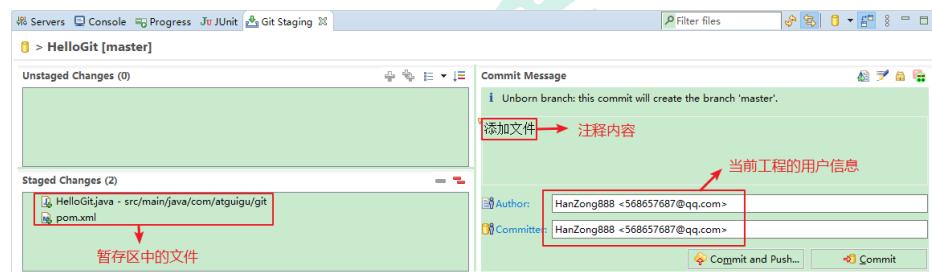


The screenshot shows the Eclipse IDE interface after adding the file to the index. The Package Explorer view shows the same project structure as before, but the file status for HelloGit.java is now indicated by a small green icon in the package tree, signifying it is tracked by version control.

- 11) 在工程上右键→Team→Commit...将工程添加到本地库

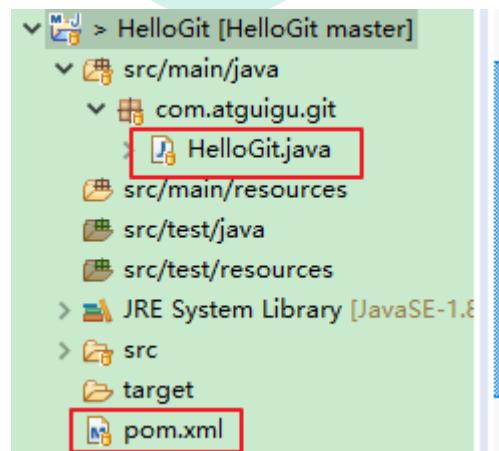


12) 添加注释后点击 Commit 将工程添加到本地库



13) 也可以直接点击 Commit and Push...添加到本地库后开始上传到项目托管的网站

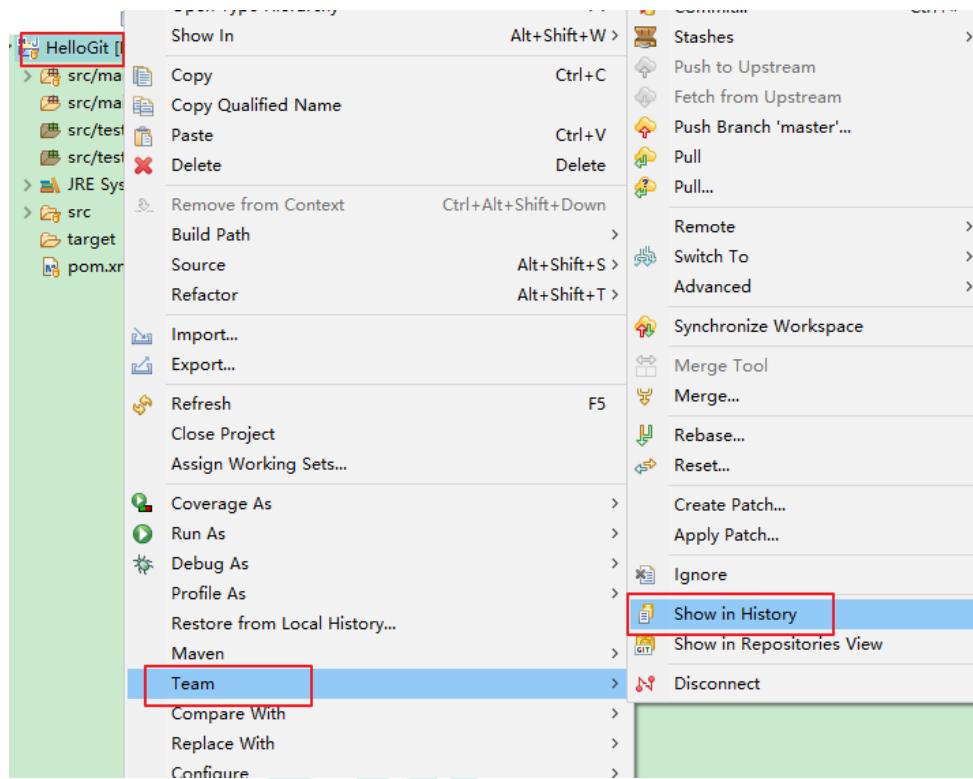
14) 工程添加到本地库之后文件的状态如下图：



3.2.2 版本间切换

1) 查看历史版本

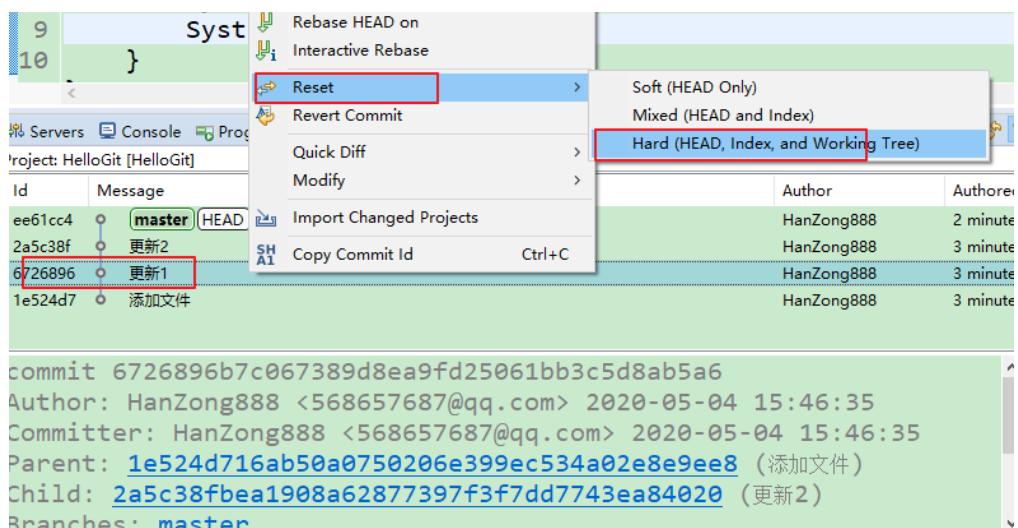
在工程上右键→Team→Show in History



2) 当前版本



3) 在要切换的版本上右键→Reset→Hard

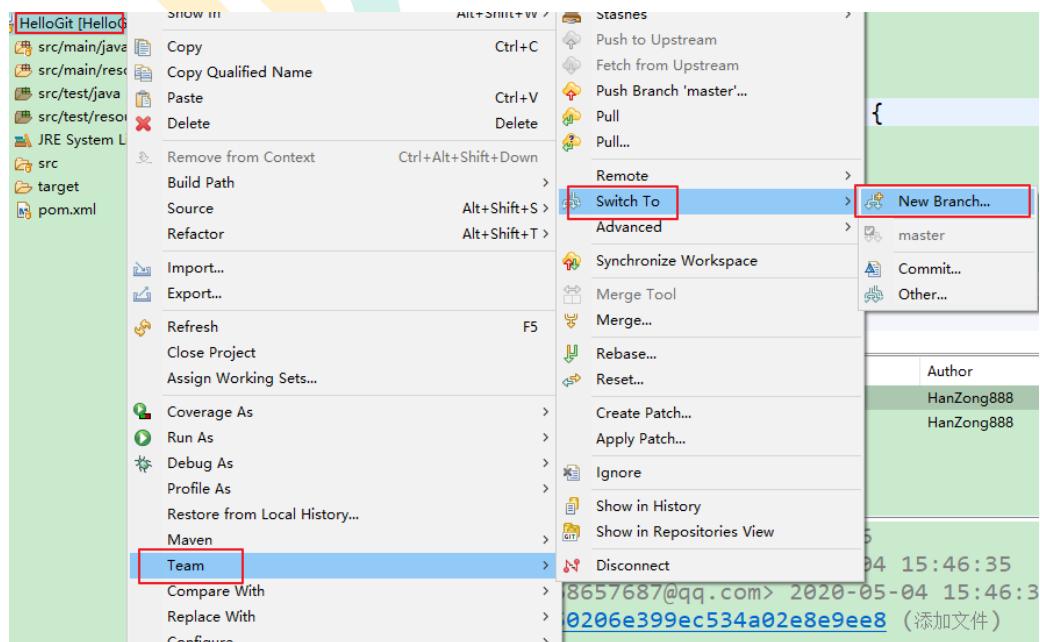


4) 切换成功

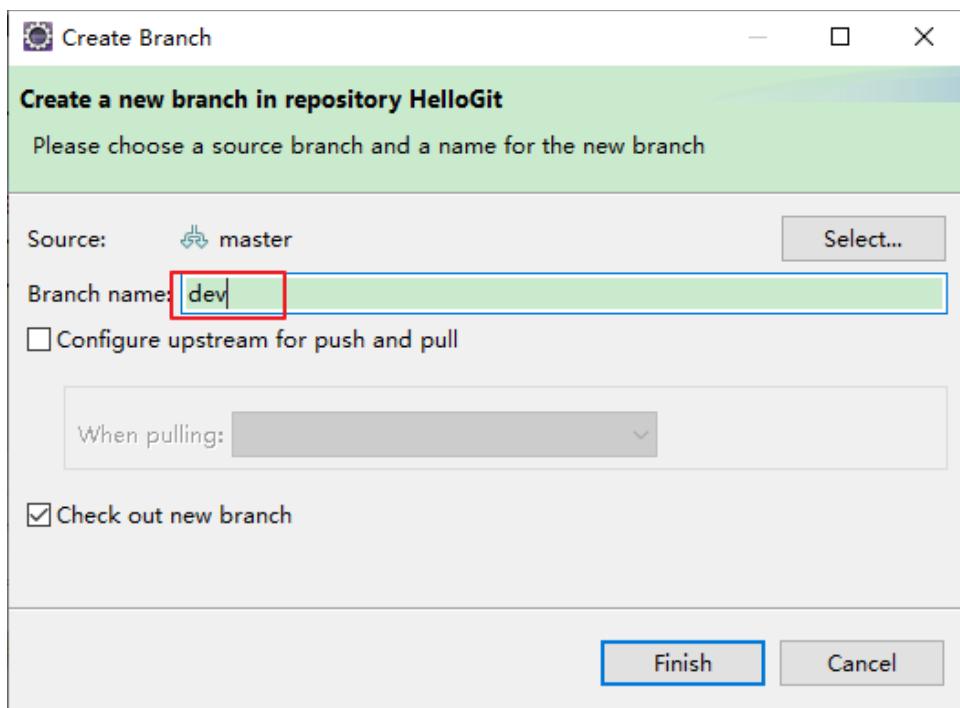


3.2.3 创建分支

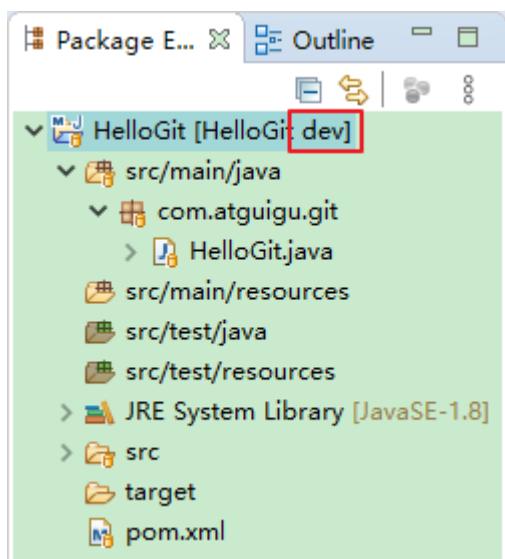
1) 在工程上右键→Team→Switch To→New Branch...



2) 给分支命名



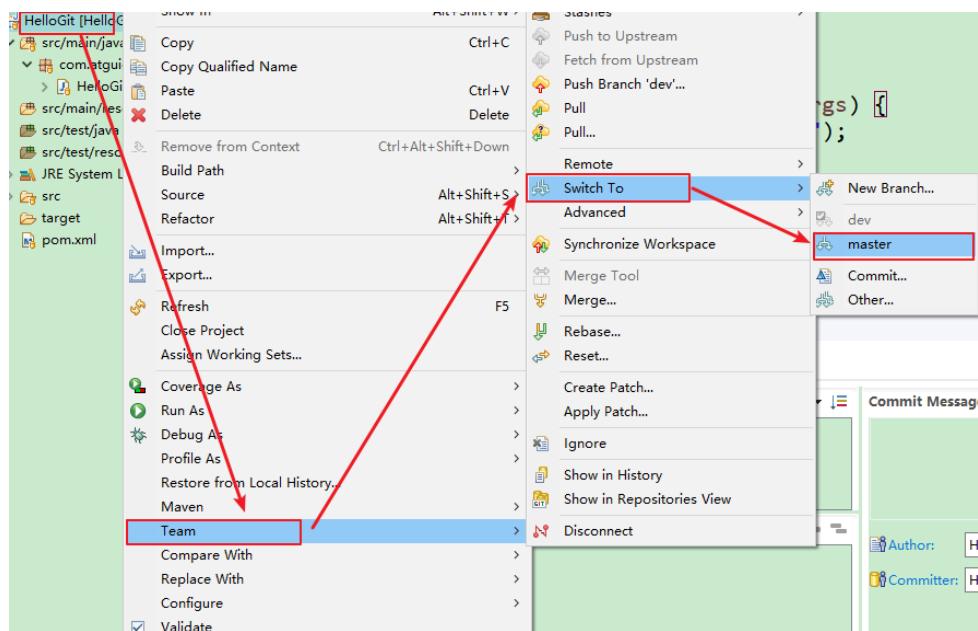
3) 点击 Finish 之后自动切换到新的分支



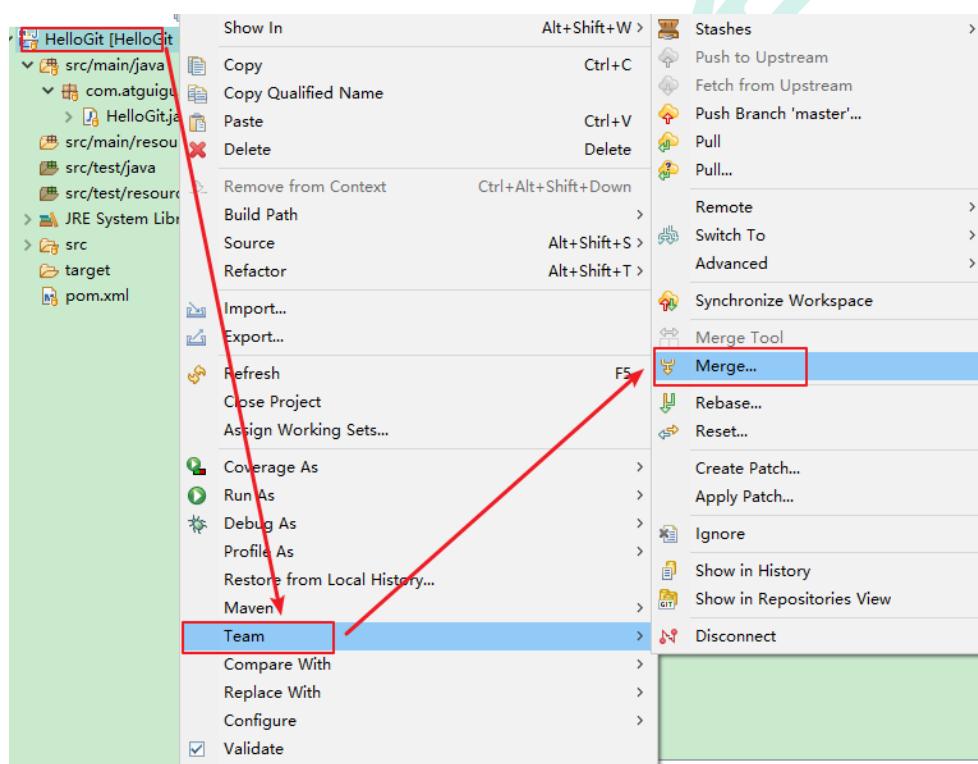
4) 在新的分支上添加新的内容，添加到暂存区，添加到本地库

3.2.4 合并分支

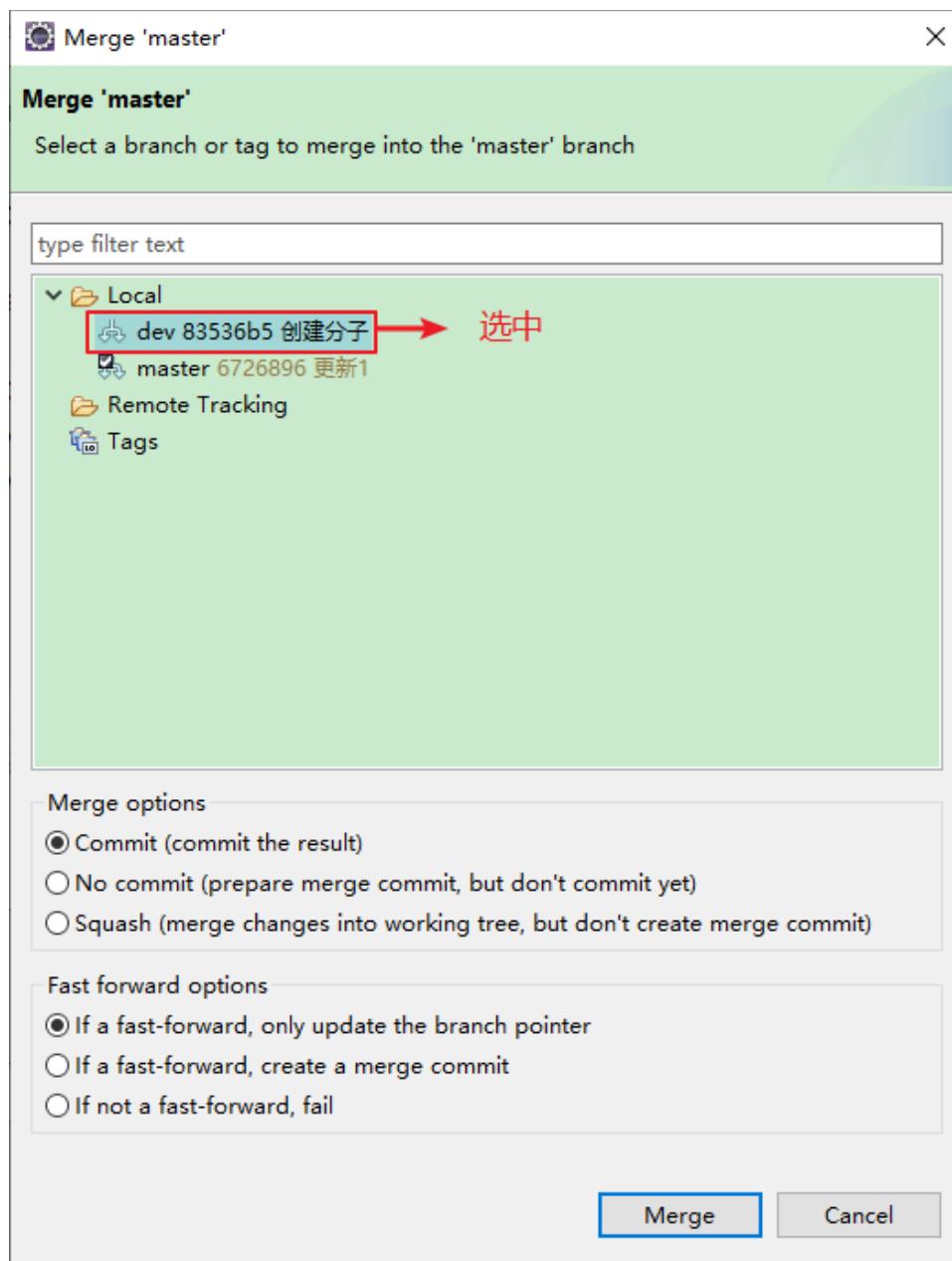
1) 切换到主干



2) 将分支中的内容合并到主干



3) 选中分支开始合并



4) 合并成功

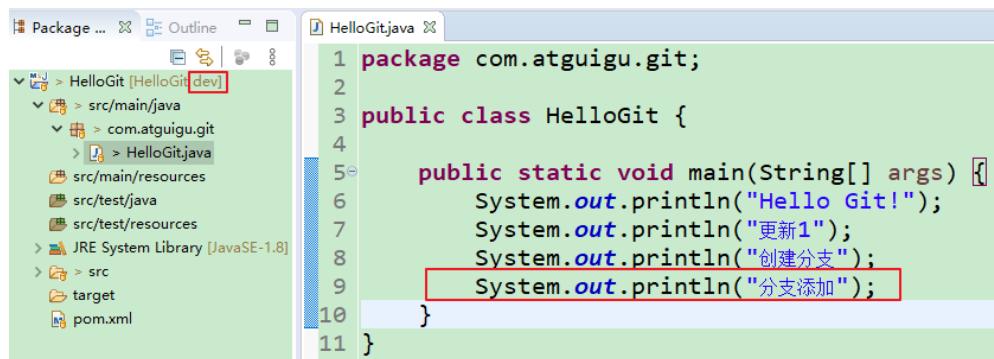
```
1 package com.atguigu.git;
2
3 public class HelloGit {
4
5     public static void main(String[] args) {
6         System.out.println("Hello Git!");
7         System.out.println("更新1");
8         System.out.println("创建分支");
9     }
10 }
```

分支中添加的内容

3.2.5 解决冲突

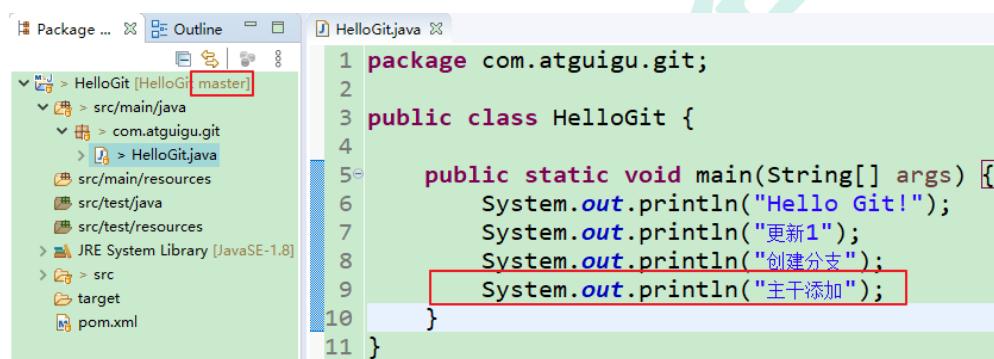
让主干和分支在同一个位置添加一行代码

- 1) 分支添加内容，并添加到暂存区和本地库



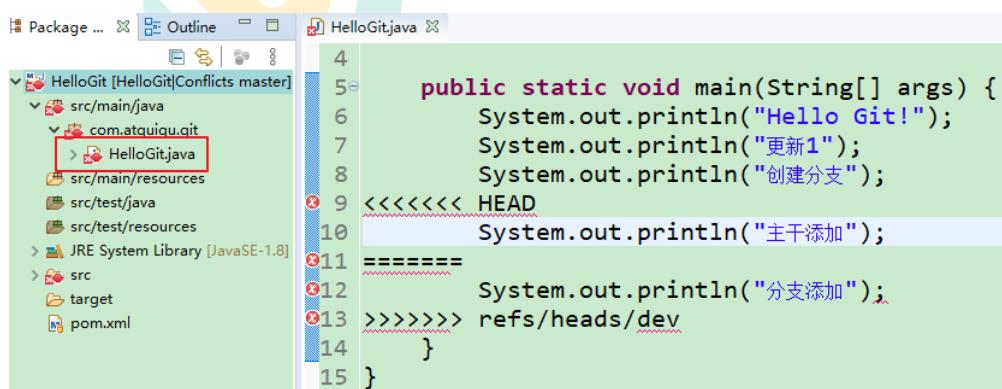
```
1 package com.atguigu.git;
2
3 public class HelloGit {
4
5     public static void main(String[] args) {
6         System.out.println("Hello Git!");
7         System.out.println("更新1");
8         System.out.println("创建分支");
9         System.out.println("分支添加");
10    }
11 }
```

- 2) 主干添加内容，并添加到暂存区和本地库



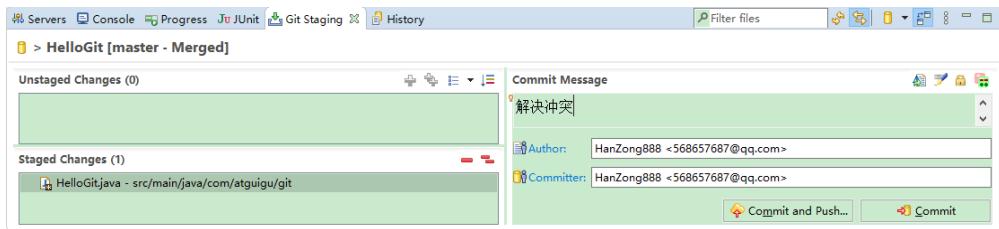
```
1 package com.atguigu.git;
2
3 public class HelloGit {
4
5     public static void main(String[] args) {
6         System.out.println("Hello Git!");
7         System.out.println("更新1");
8         System.out.println("创建分支");
9         System.out.println("主干添加");
10    }
11 }
```

- 3) 在主干上合并分支，出现冲突



```
4
5     public static void main(String[] args) {
6         System.out.println("Hello Git!");
7         System.out.println("更新1");
8         System.out.println("创建分支");
9         <<<<< HEAD
10        =====
11        System.out.println("主干添加");
12        =====
13        System.out.println("分支添加");
14    }
15 }
```

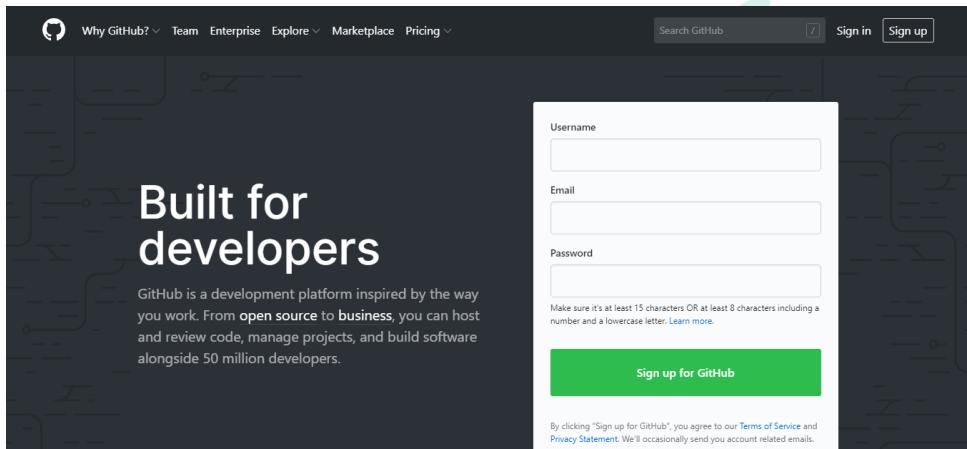
- 4) 有效沟通后选择保留的代码，重写添加到暂存区、本地库冲突即可解决



3.3 将本地库上传到 GitHub

3.3.1 注册 GitHub 账号

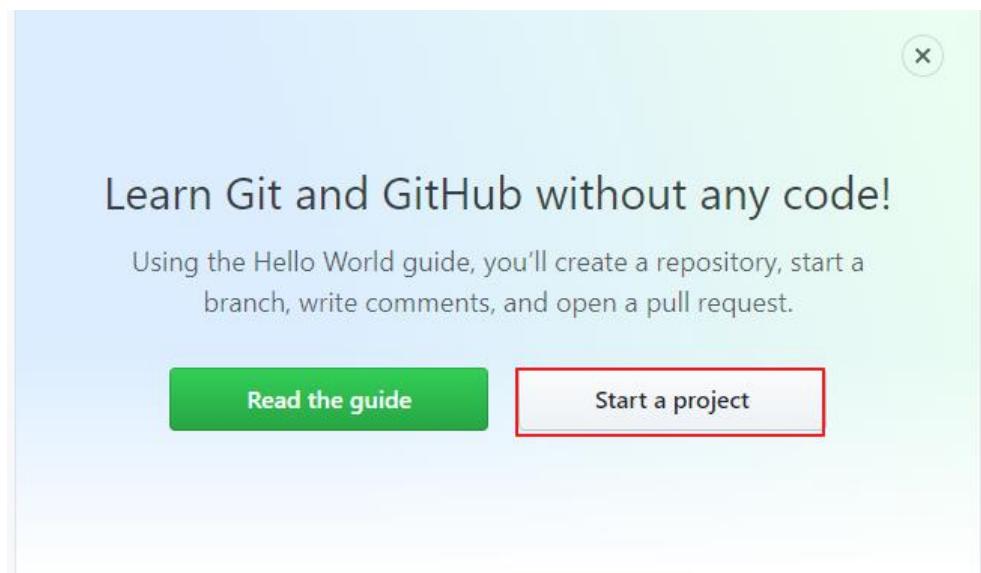
- 1) 访问 GitHub 网站 <https://github.com/>, 首页即是注册页面



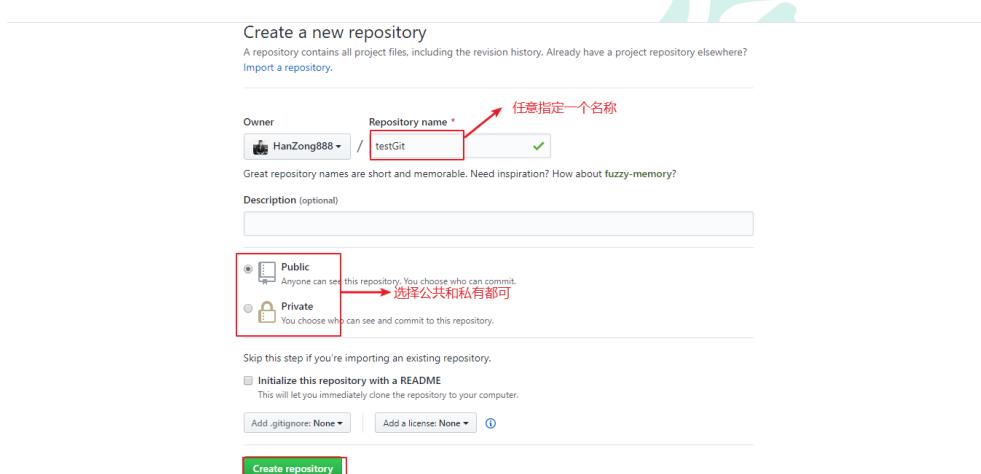
- 2) 输入用户名、邮箱、密码点击注册之后登录邮箱激活即可

3.3.2 上传本地库

- 1) 登录 GitHub 在首页点击 Start a project



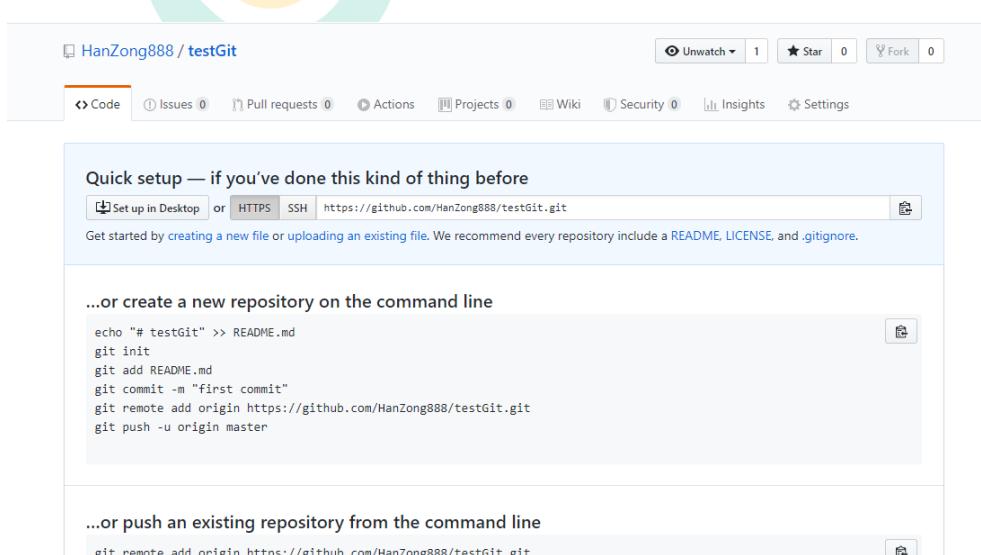
2) 指定仓库的名称和类型



任意指定一个名称

选择公共和私有都可

3) 仓库创建成功



Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH <https://github.com/HanZong888/testGit.git>

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

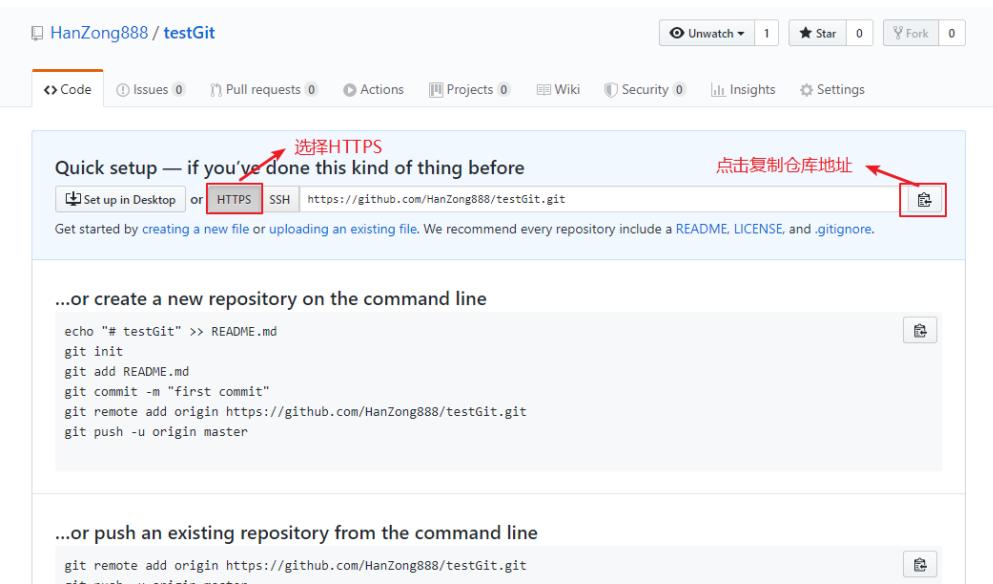
...or create a new repository on the command line

```
echo "# testGit" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/HanZong888/testGit.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/HanZong888/testGit.git
```

4) 复制仓库地址



选择HTTPS 点击复制仓库地址

Quick setup — if you've done this kind of thing before

Set up in Desktop or **HTTPS** SSH <https://github.com/HanZong888/testGit.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

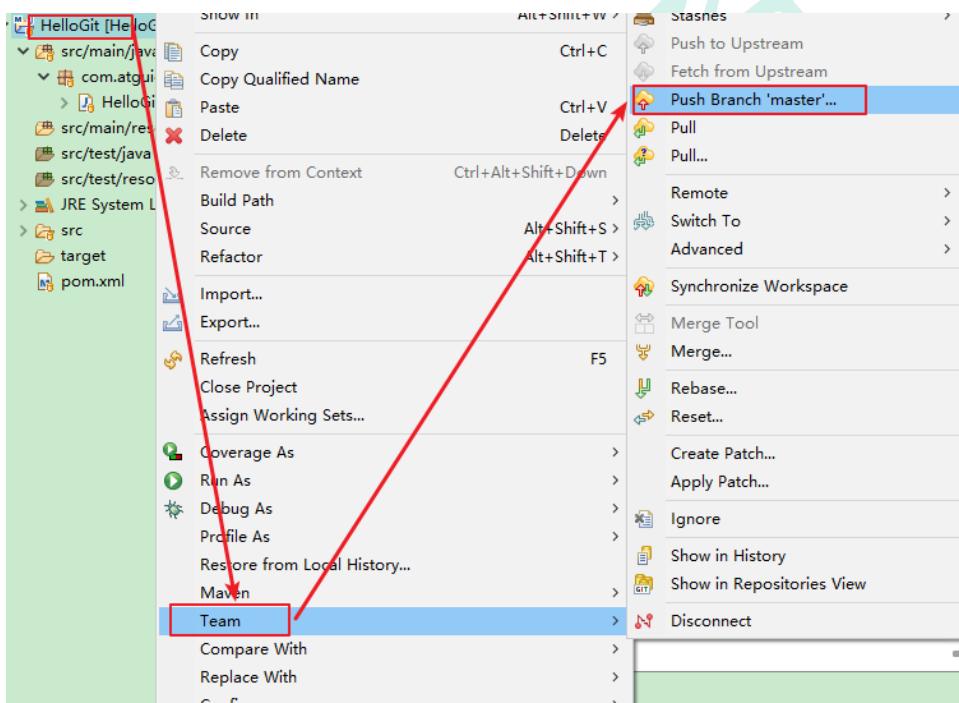
...or create a new repository on the command line

```
echo "# testGit" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git remote add origin https://github.com/HanZong888/testGit.git  
git push -u origin master
```

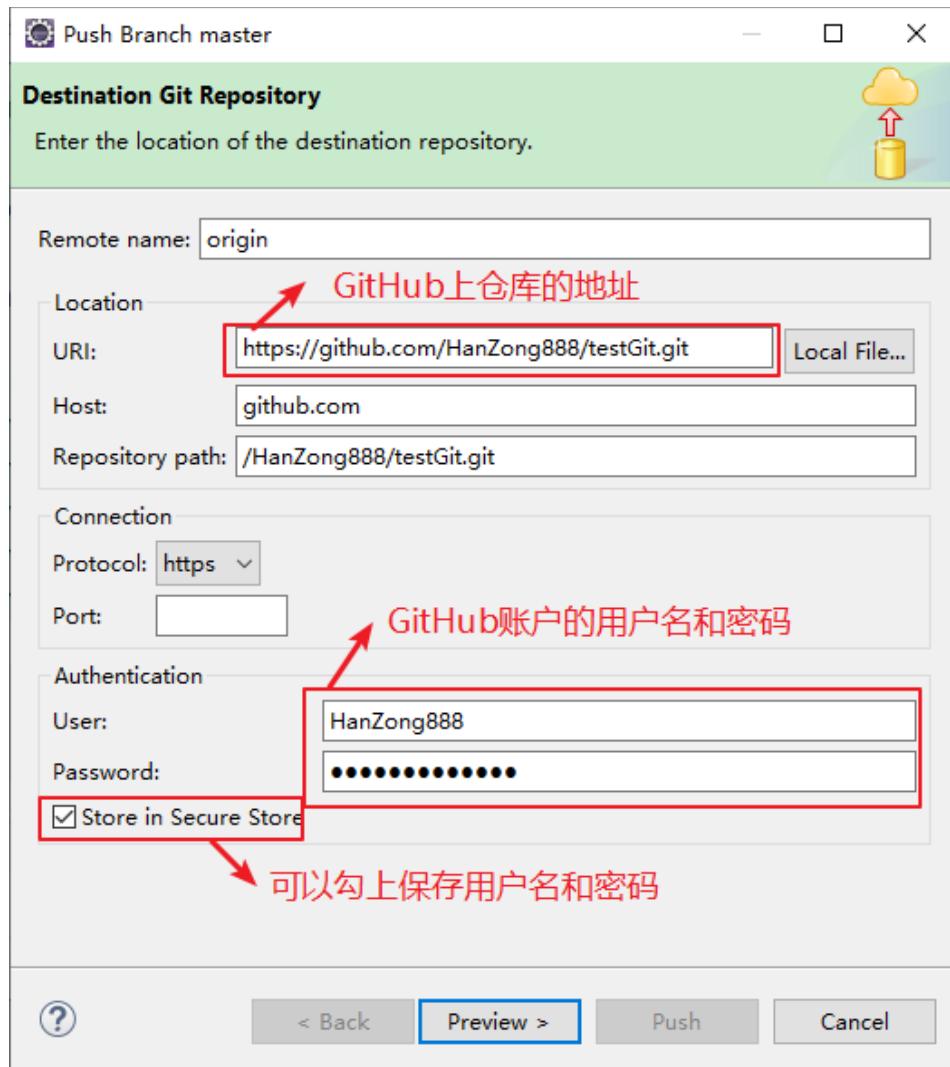
...or push an existing repository from the command line

```
git remote add origin https://github.com/HanZong888/testGit.git  
git push -u origin master
```

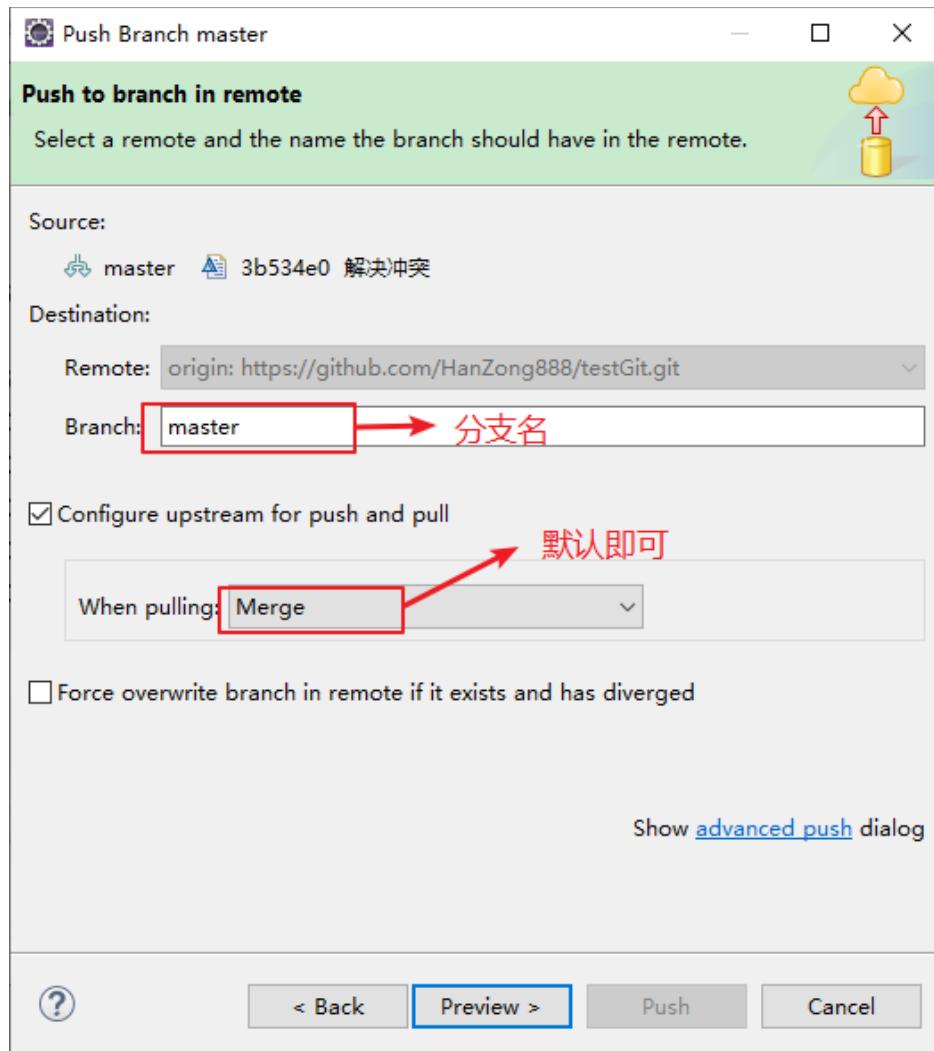
5) 将本地库上传到 GitHub 上创建的仓库中



6) 指定仓库地址、用户名和密码



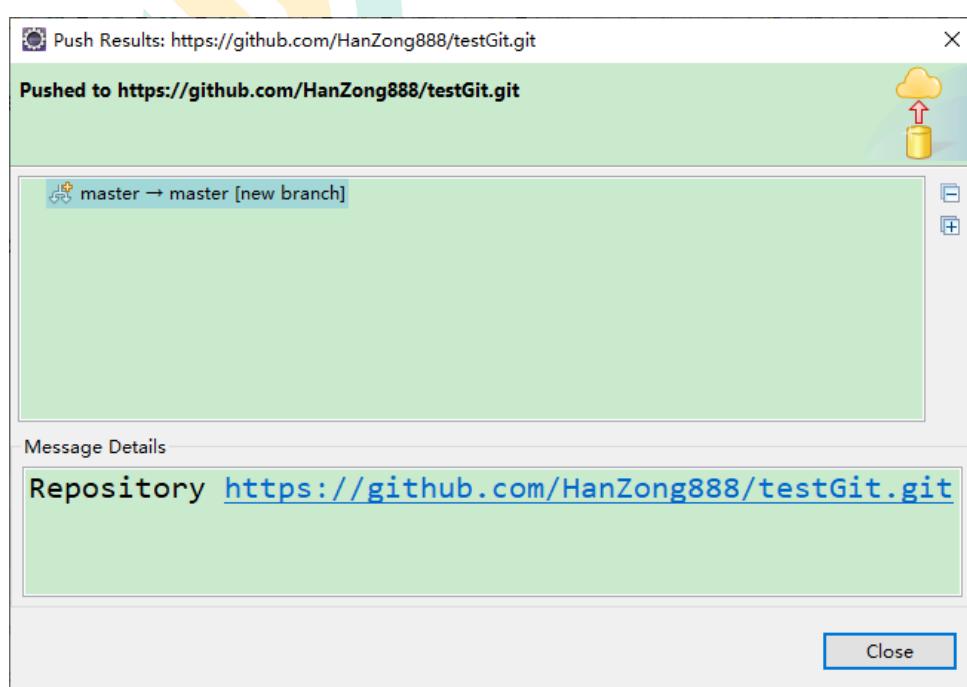
7) 点击 Preview



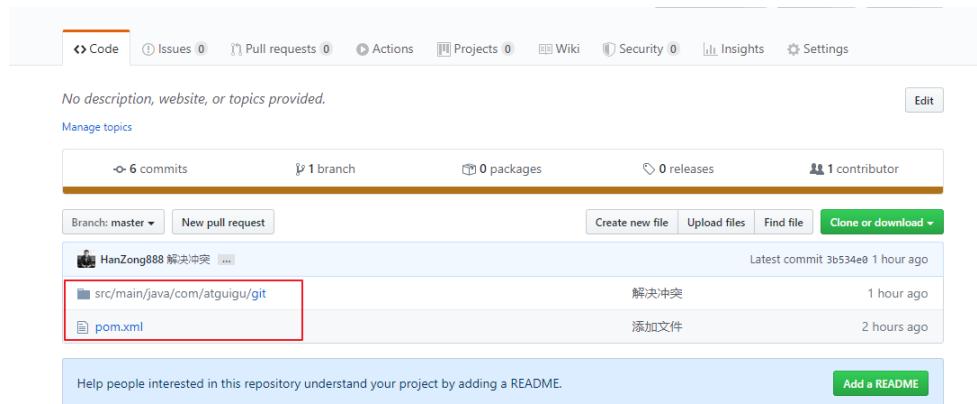
8) 点击 Preview 开始连接 GitHub, 然后点击 Push 开始上传



9) 上传成功



10) 查看 GitHub 仓库

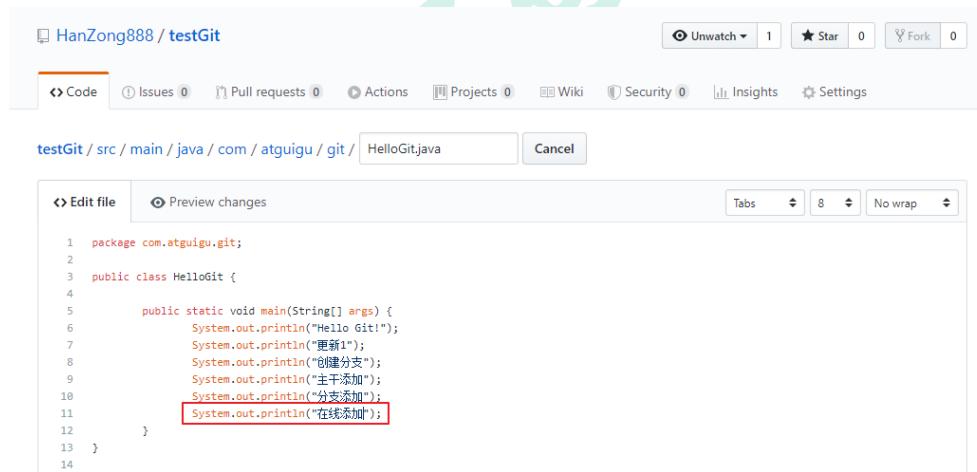


The screenshot shows a GitHub repository page. At the top, there are navigation links: Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security (0), Insights, and Settings. Below the header, it says "No description, website, or topics provided." and has a "Edit" button. A "Manage topics" link is also present. The main area displays commit statistics: 6 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. A dropdown menu shows the branch is "master". There are buttons for "Create new file", "Upload files", "Find file", and "Clone or download". A recent commit by "HanZong888" is shown, which resolved a conflict and added a file named "pom.xml". The commit was made 1 hour ago. A note at the bottom encourages adding a README, with a "Add a README" button.

3.3.3 更新本地库

项目在 GitHub 上被合作伙伴更新之后，我们就需要将 GitHub 上最新的代码拉到本地库，否则会上传失败！接下来我们以在 GitHub 上在线添加内容演示如何更新本地库。

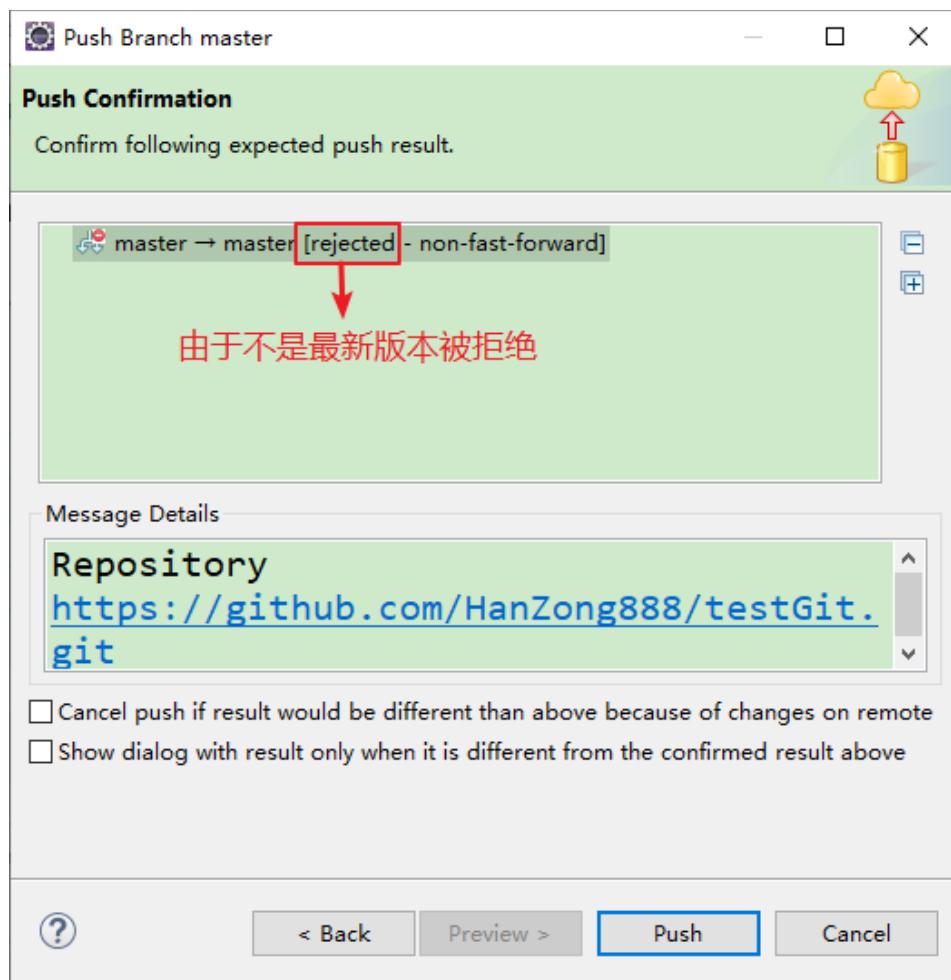
1) 在 GitHub 上在线修改文件



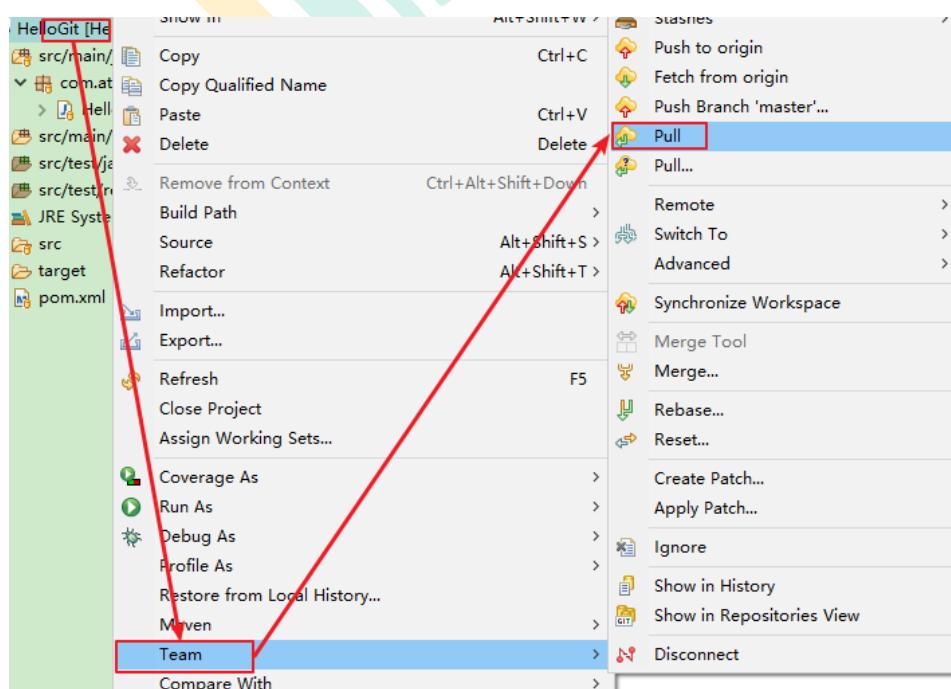
The screenshot shows the GitHub inline code editor for the file "HelloGit.java" in the repository "HanZong888/testGit". The editor interface includes tabs for "Edit file" and "Preview changes", and settings for "Tabs" (set to 8) and "No wrap". The code editor displays the Java code for "HelloGit.java", with the line "System.out.println("在线添加");" highlighted and enclosed in a red box. The code content is as follows:

```
1 package com.atguigu.git;
2
3 public class HelloGit {
4
5     public static void main(String[] args) {
6         System.out.println("Hello Git!");
7         System.out.println("更新1");
8         System.out.println("创建分支");
9         System.out.println("主干添加");
10        System.out.println("分支添加");
11        System.out.println("在线添加");
12    }
13 }
14
```

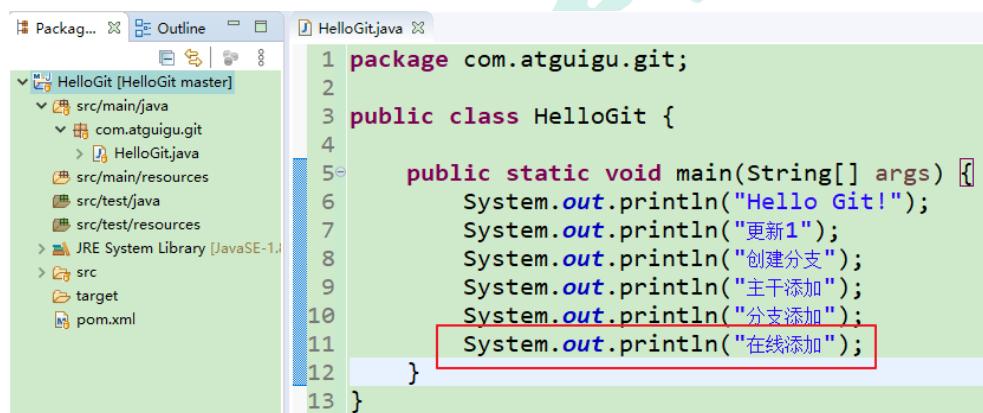
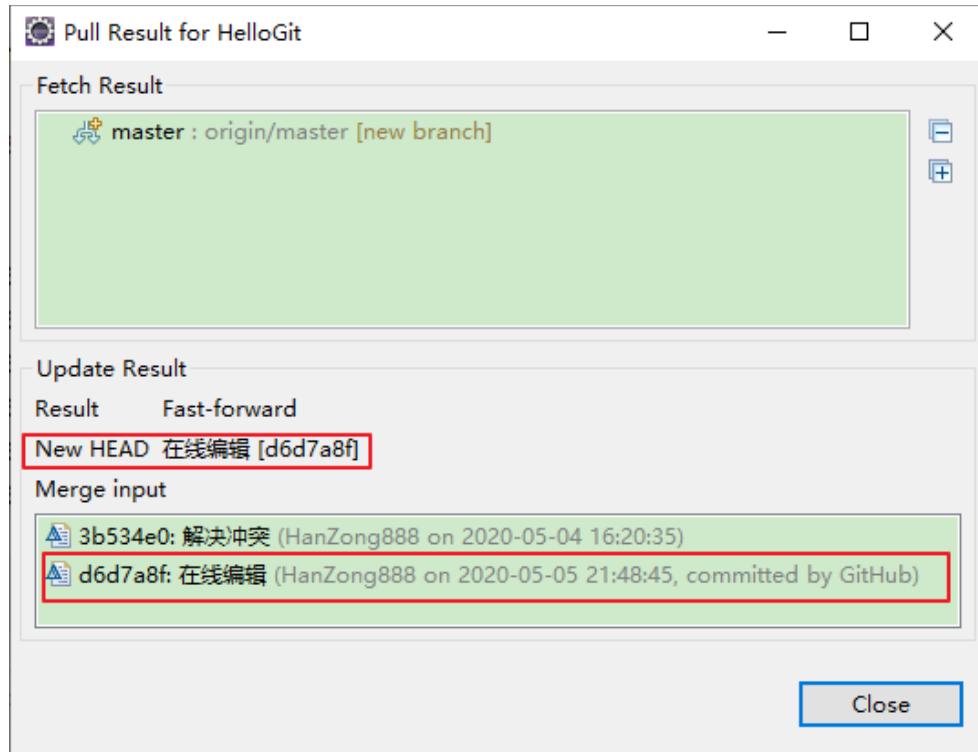
2) 在 Eclipse 如果不更新本地库直接上传会由于不是最新的版本而被拒绝



3) 将 GitHub 上最新的内容 Pull 下来



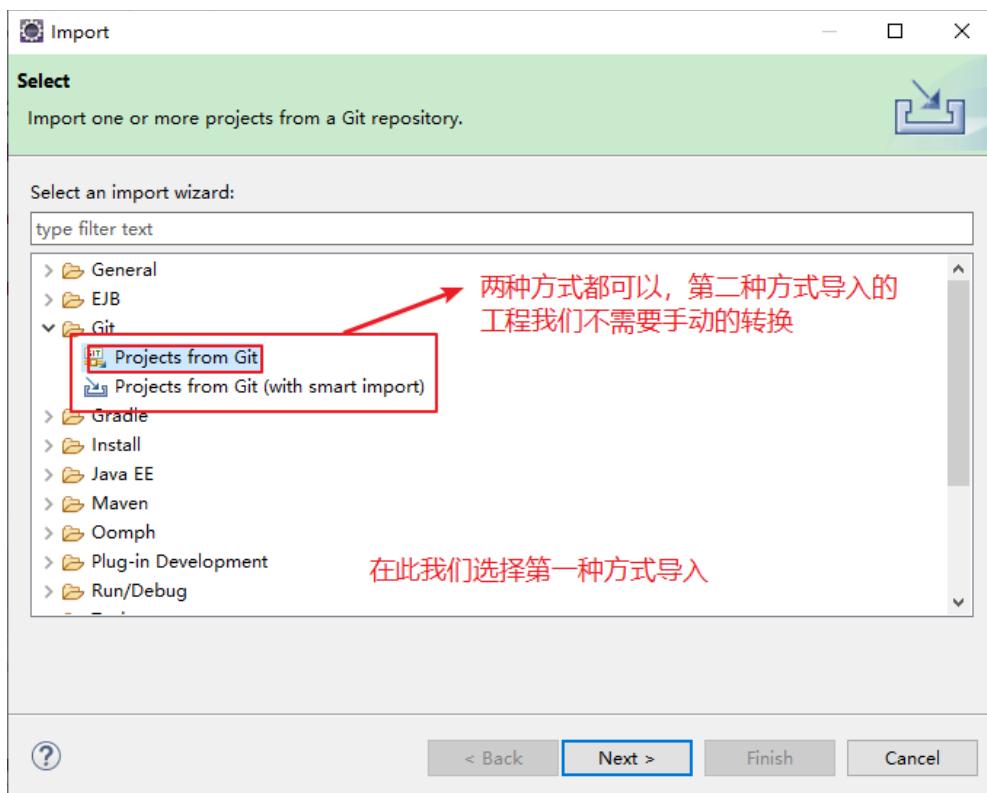
4) 更新本地库成功



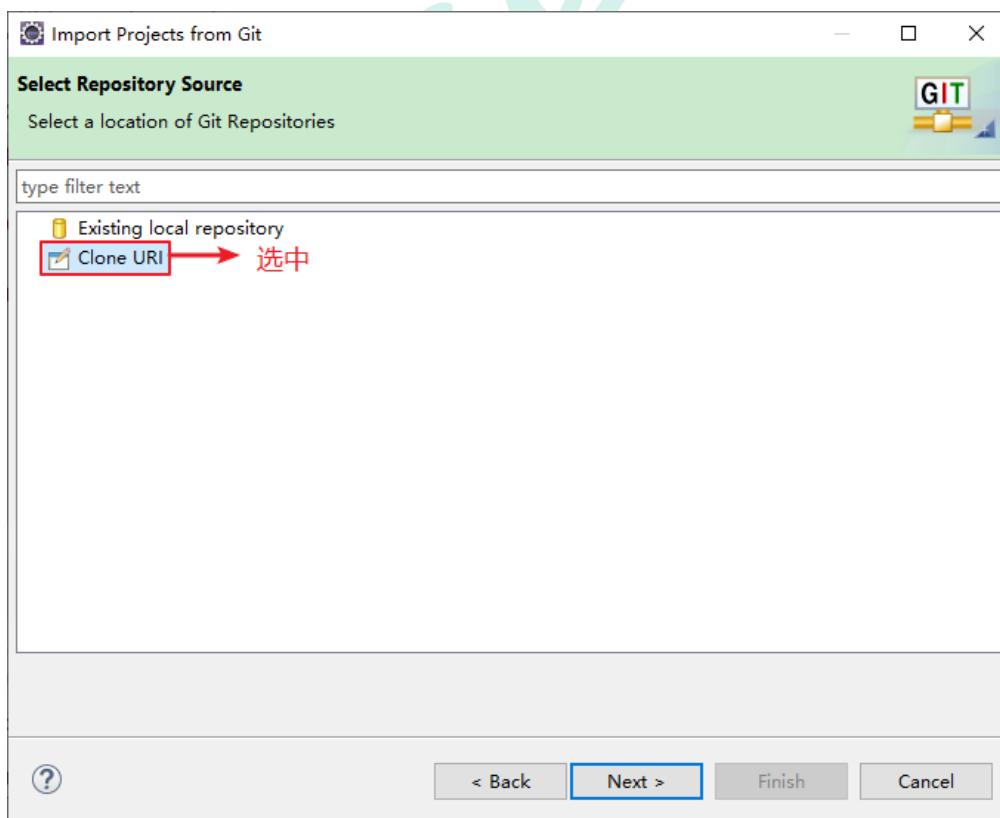
3.4 从 GitHub 上克隆项目到本地

3.4.1 克隆项目

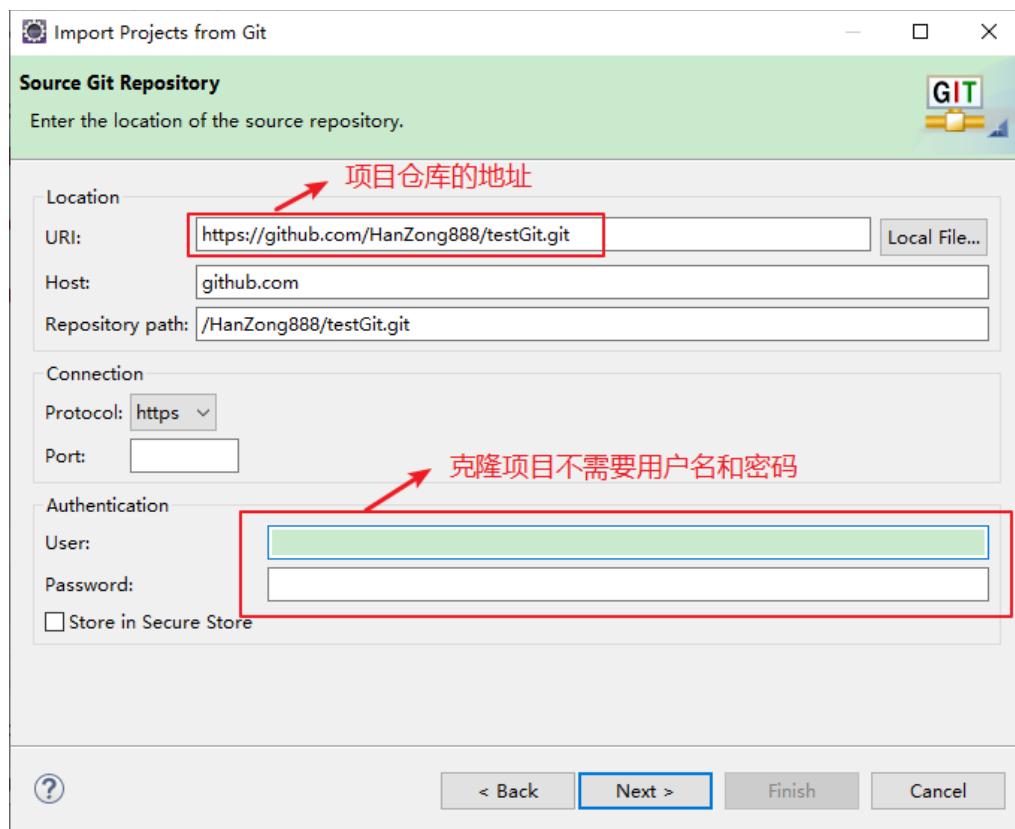
- 1) 在 Eclipse 中点击 File→Import...→Git



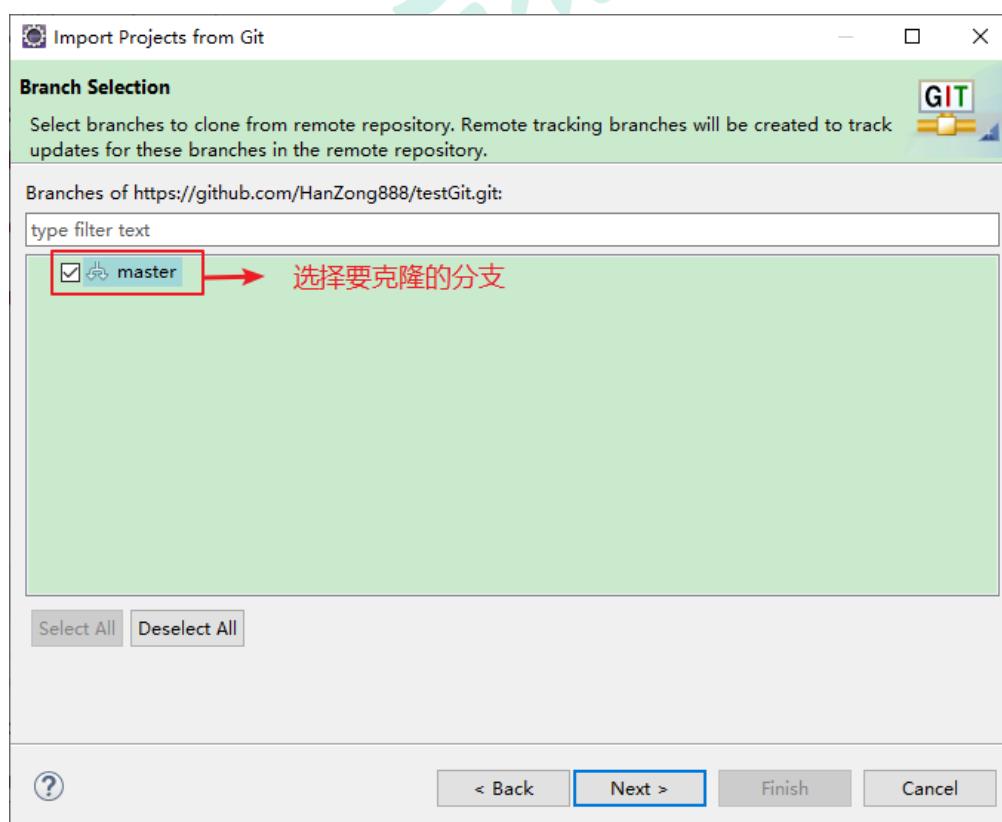
2) 选中 Clone URI



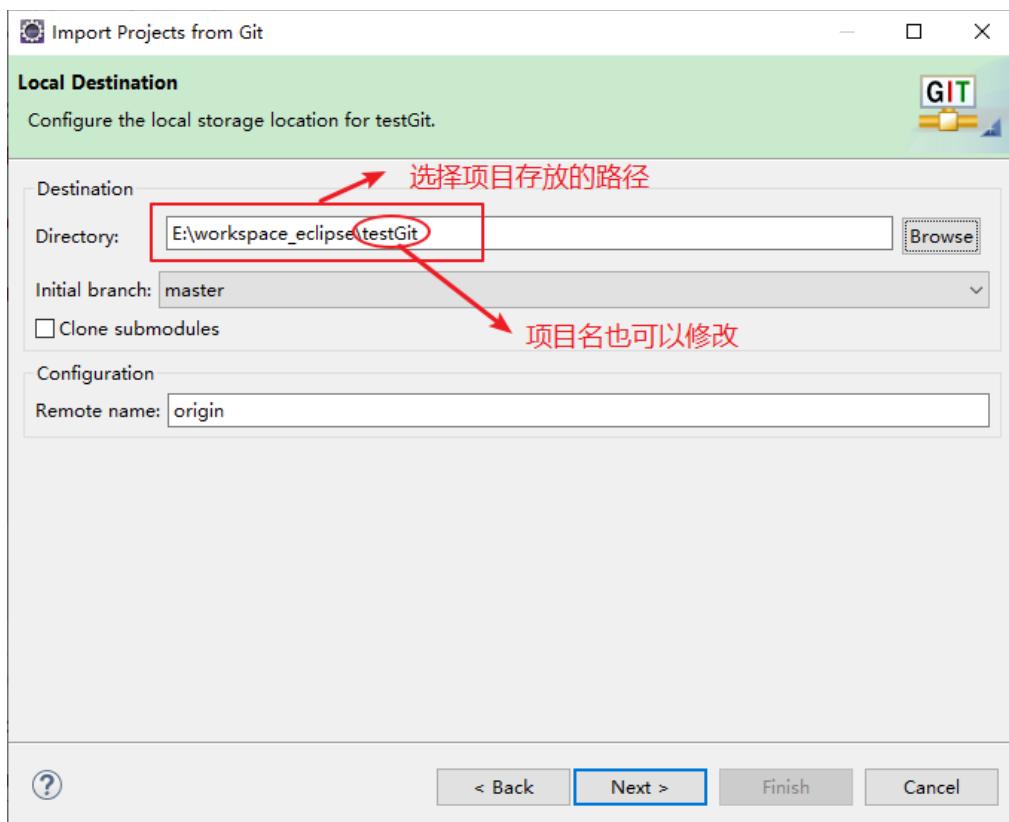
3) 输入克隆的项目在 GitHub 上仓库的地址



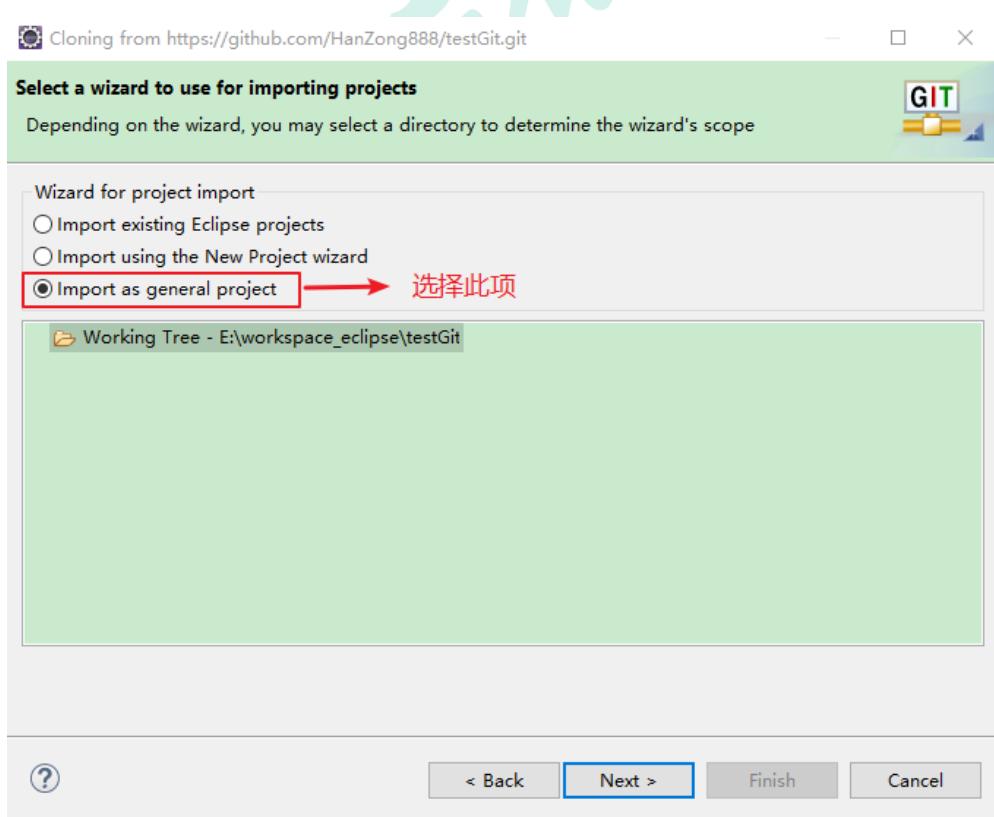
4) 选择要克隆的分支



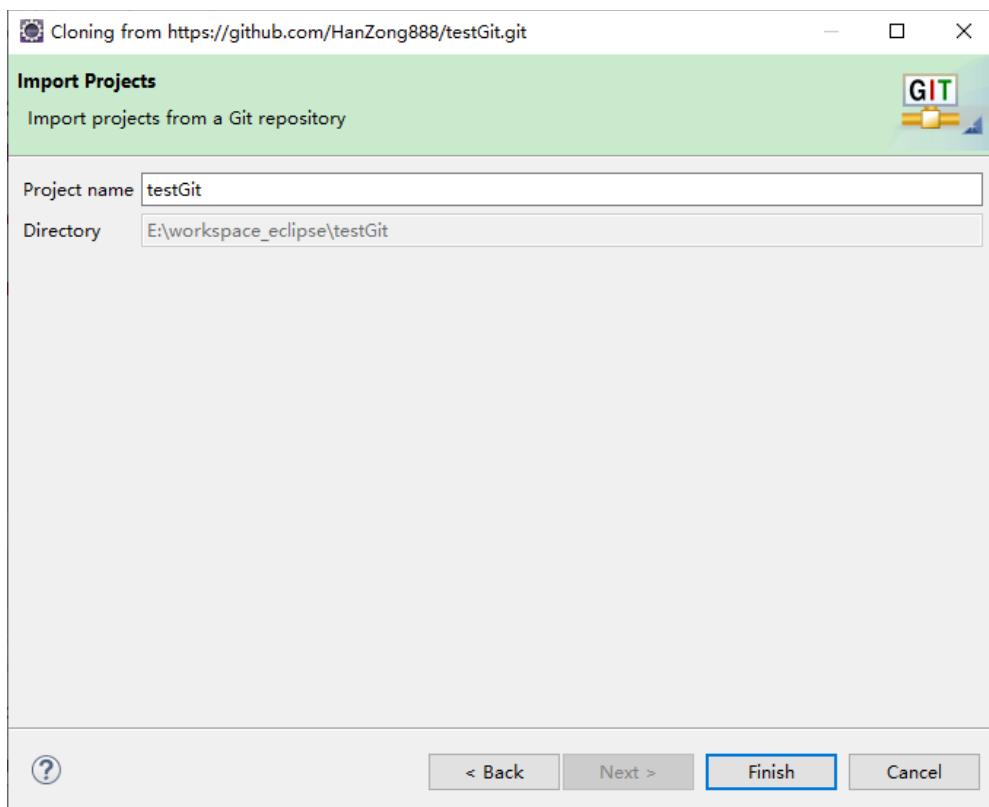
5) 选择项目存放的路径



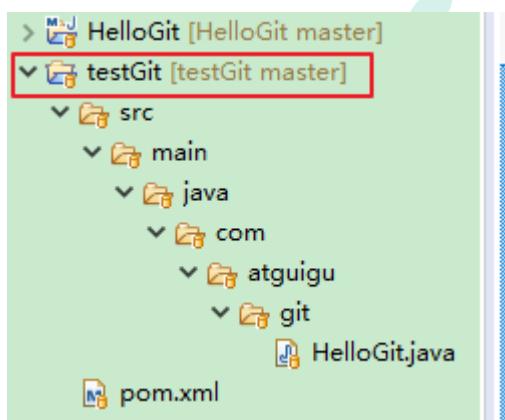
6) 选择作为一个普通工程导入（通过方式二导入没有这一步）



7) 点击完成

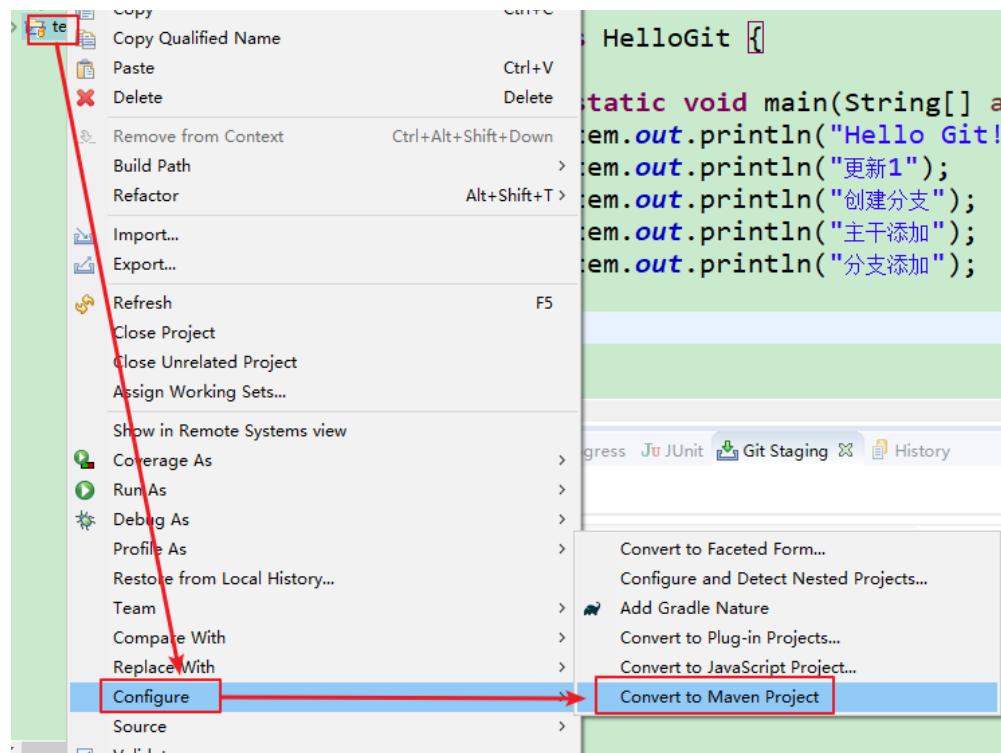


8) 导入之后并不是一个 Maven 工程 (如果通过方式二导入会自动识别为 Maven 工程)

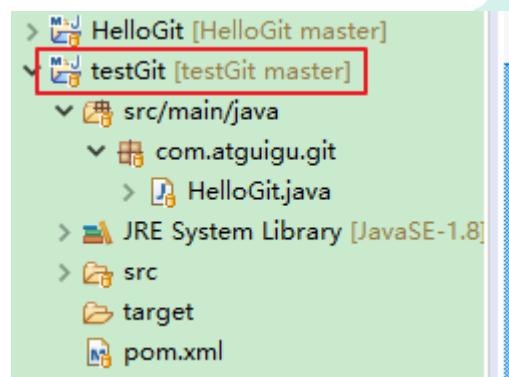


9) 转换为 Maven 工程

右键→Configure→Convert to Maven Project

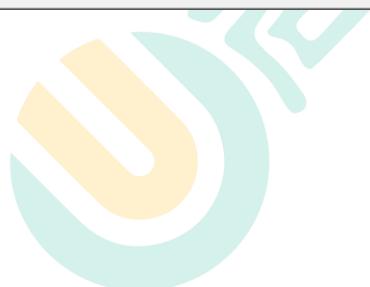
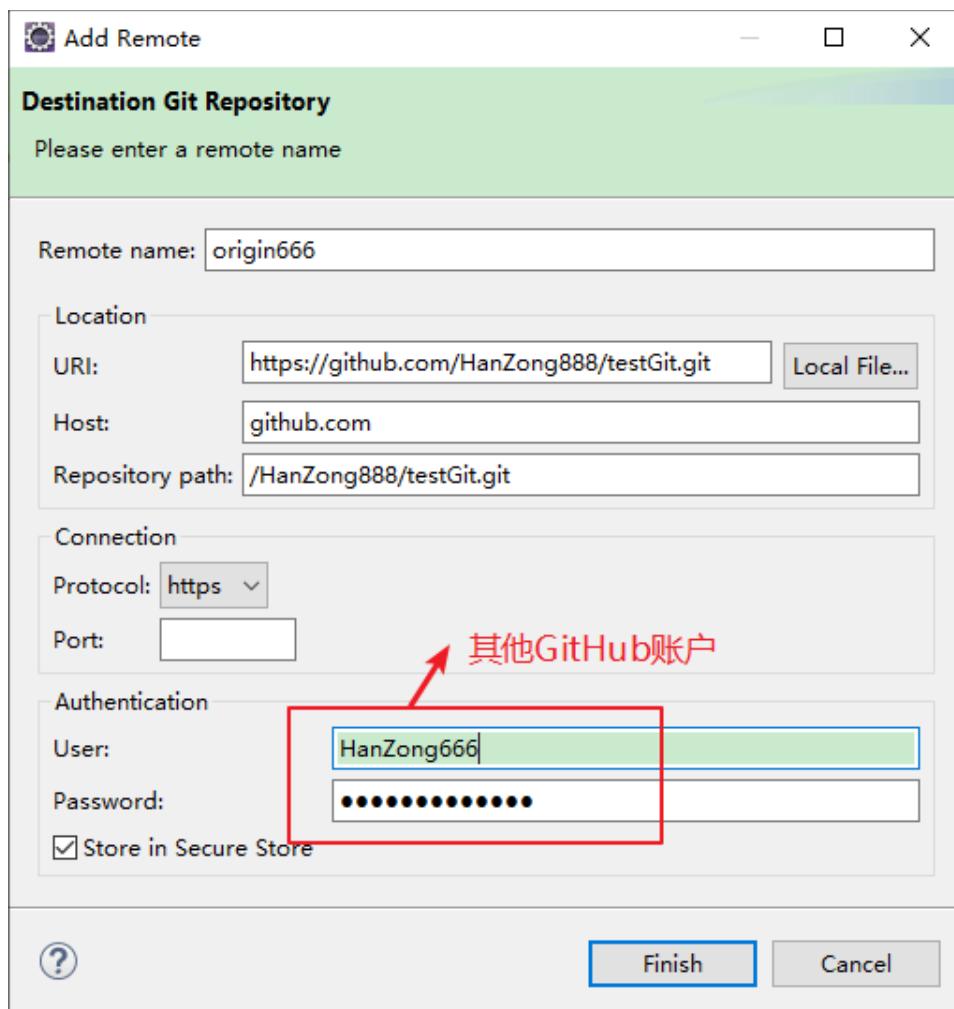


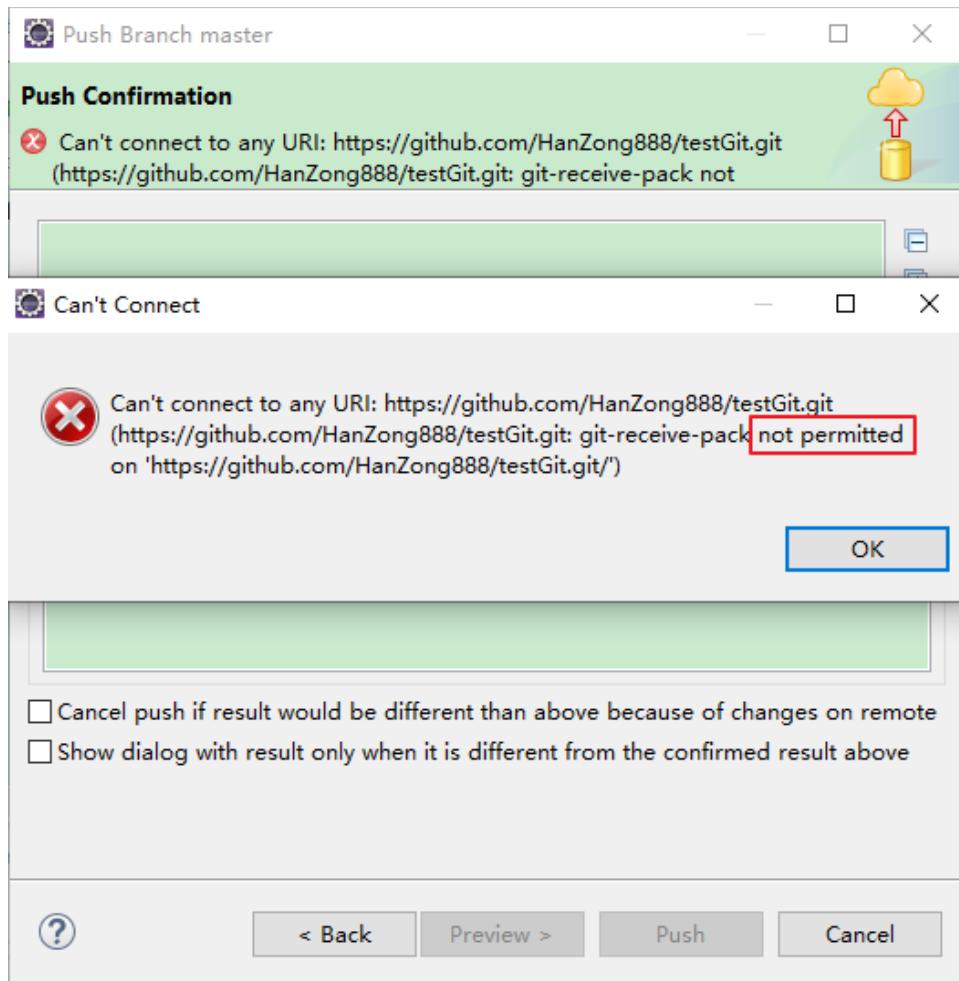
10) 转换之后



3.4.2 添加合作伙伴

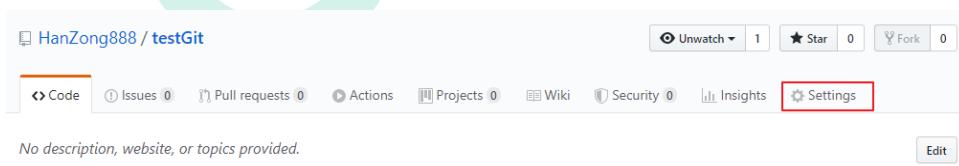
在项目的协同开发过程中，如果 GitHub 上的仓库不是你创建的，你克隆下来的项目完成代码的编辑之后上传会失败，如下图：



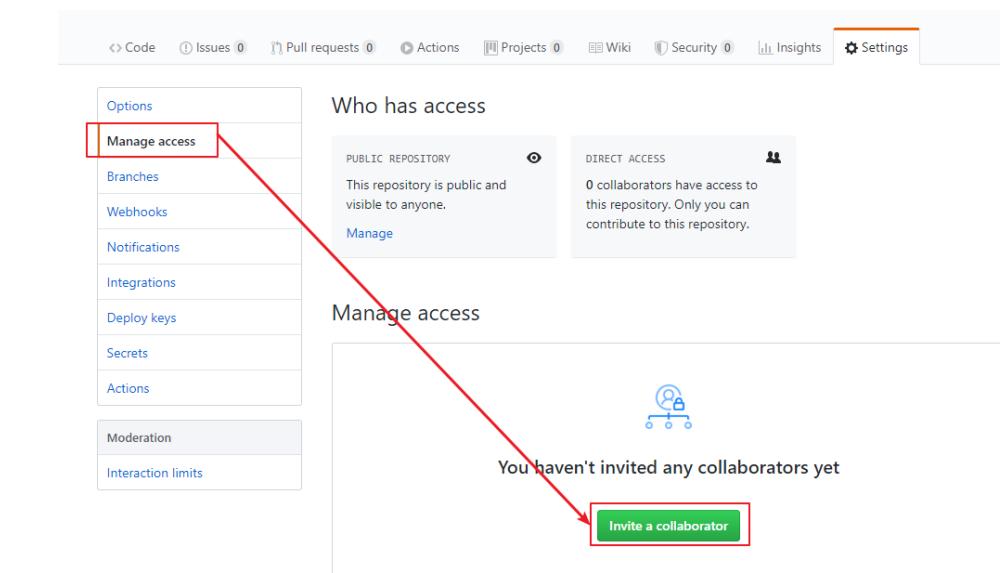


此时如果想要上传成功，必须让 GitHub 上仓库的拥有者添加你为合作伙伴，
添加合作伙伴的步骤：

1) 让仓库拥有者在仓库上点击 settings



2) 点击 Manage Access



The screenshot shows the GitHub repository settings page for a public repository. On the left, a sidebar lists various options like Manage access, Branches, Webhooks, Notifications, Integrations, Deploy keys, Secrets, Actions, Moderation, and Interaction limits. The 'Manage access' option is highlighted with a red box and has a red arrow pointing to the 'Manage access' section on the right. In the 'Who has access' section, there are two tabs: 'PUBLIC REPOSITORY' (selected) and 'DIRECT ACCESS'. The 'PUBLIC REPOSITORY' tab indicates the repository is public and visible to anyone. The 'DIRECT ACCESS' tab shows 0 collaborators have access, with a note that only the user can contribute. Below this, the 'Manage access' section displays a message: 'You haven't invited any collaborators yet' and features a green 'Invite a collaborator' button.

3) 搜索合作伙伴，即搜索你的 GitHub 账户



x

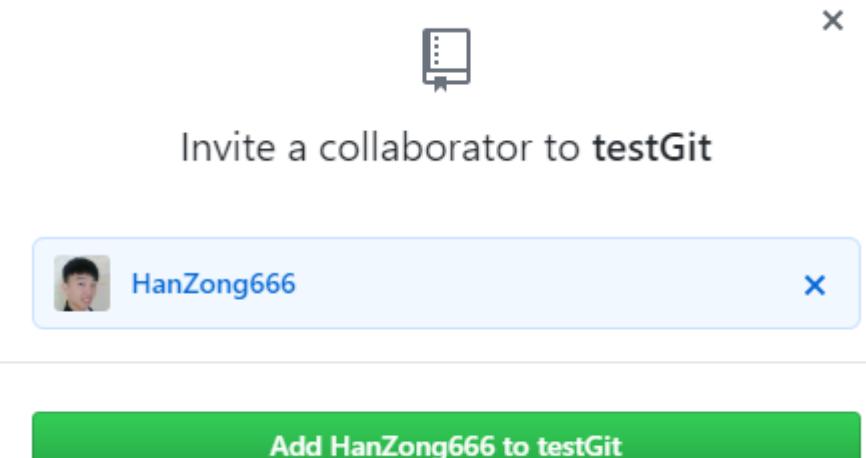
Invite a collaborator to testGit



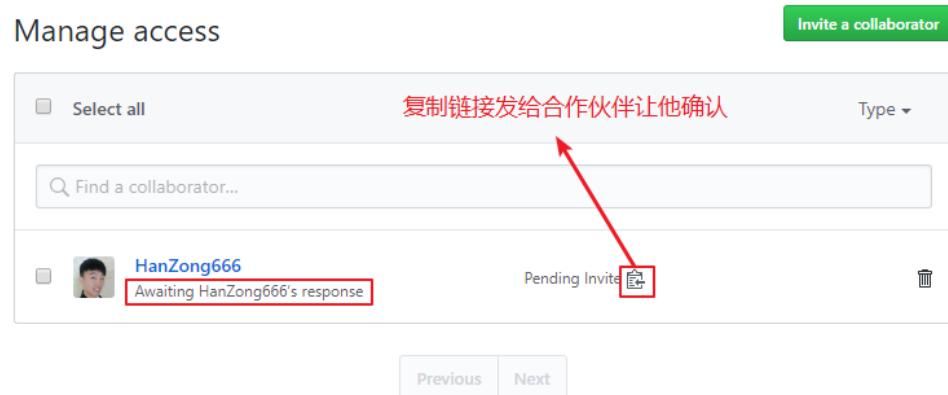
A search result for 'HanZong666' is shown, featuring a small profile picture and the name 'HanZong666' followed by the text 'Invite collaborator'.

4) 点击邀请



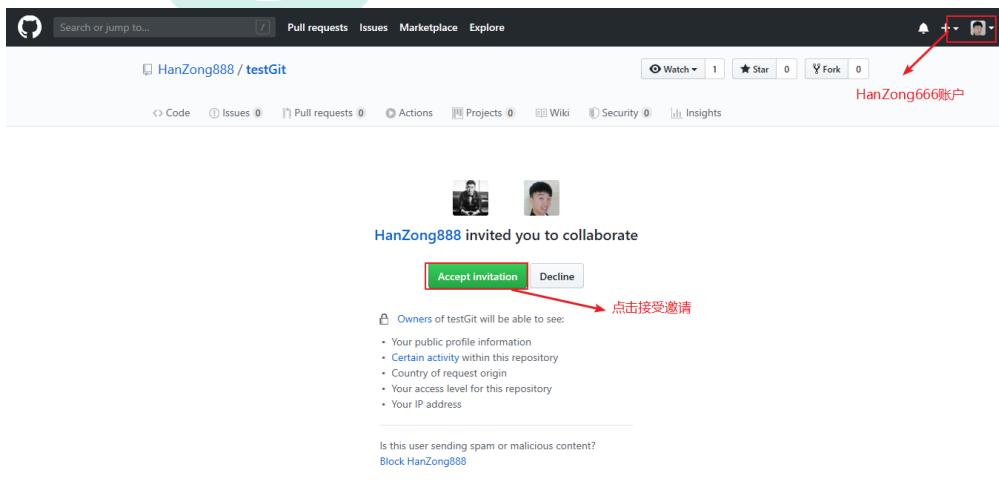


5) 等待你回复



The screenshot shows the "Manage access" section of a GitHub repository. It lists a single collaborator named "HanZong666" with the status "Awaiting HanZong666's response". To the right of the status is a red-bordered "Pending Invite" button with a copy icon. Above the list, a note says "复制链接发给合作伙伴让他确认" (Copy the link and send it to the partner for confirmation). A red arrow points from this note to the "Pending Invite" button.

6) 仓库拥有者可以将链接发送给你让你确认，当然你的邮箱也会收到等待确认的邮件



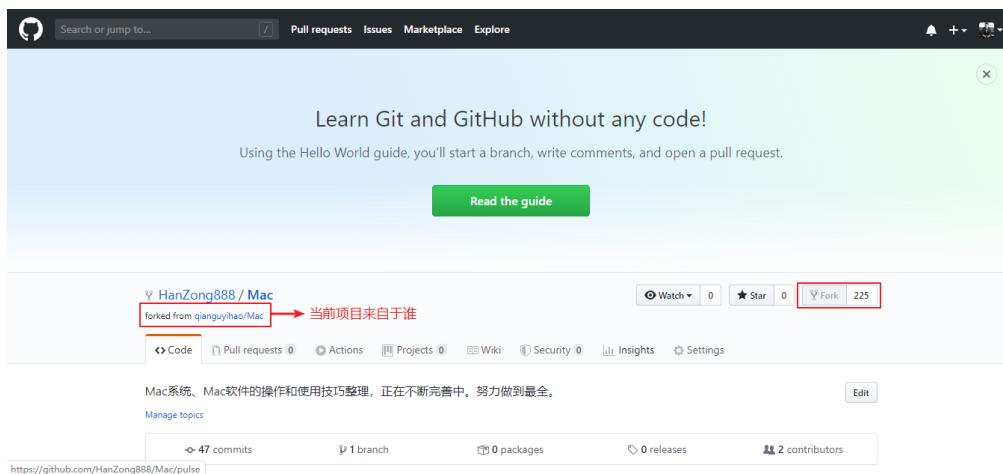
The screenshot shows a GitHub user profile for "HanZong888". In the top right corner, there is a notification badge with the number "1" and a red arrow pointing to it, indicating a new message. The message content is: "HanZong888 invited you to collaborate". Below the message are two buttons: "Accept invitation" (green) and "Decline". A red arrow points from the text "点击接受邀请" (Click to accept the invitation) to the "Accept invitation" button. Below the buttons, there is a list of what the owner can see: "Owners of testGit will be able to see:" followed by a bulleted list: "Your public profile information", "Certain activity within this repository", "Country of request origin", "Your access level for this repository", and "Your IP address". At the bottom of the message, there is a link "Is this user sending spam or malicious content? Block HanZong888".

7) 等你接收之后就与仓库拥有者成为了合作伙伴，就可以向仓库上传项目了

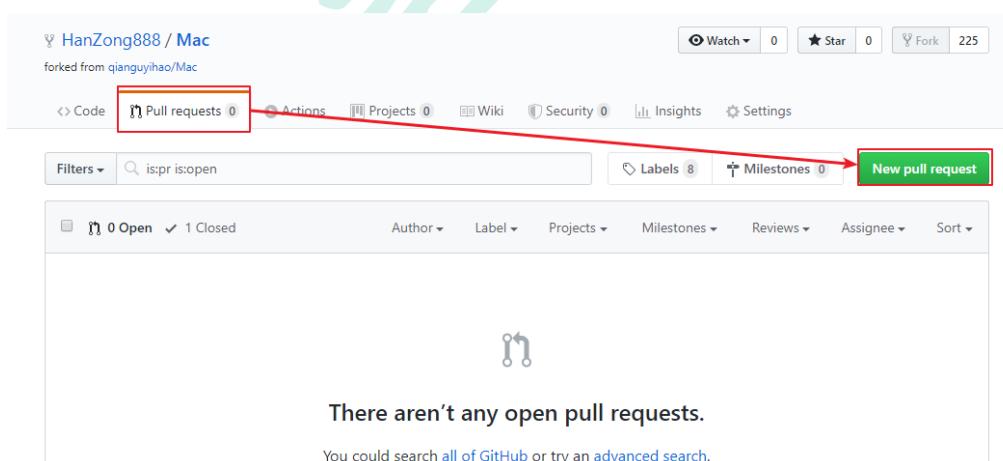
3.4.3 非合作伙伴如何共同开发项目

GitHub 上有好多开源的好项目，我们可以下载下来查看、借鉴别人的代码。但是如果我们修改了，由于不是对方的合作伙伴，我们无法将代码上传到别人的仓库，此时我们可以选择使用 fork 和 pullrequest 操作

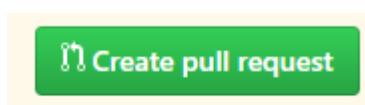
- 1) 看到喜欢的项目点击 fork 操作将别人的项目复制一份作为自己的仓库，同时仓库下面会显示当前项目来自于哪里



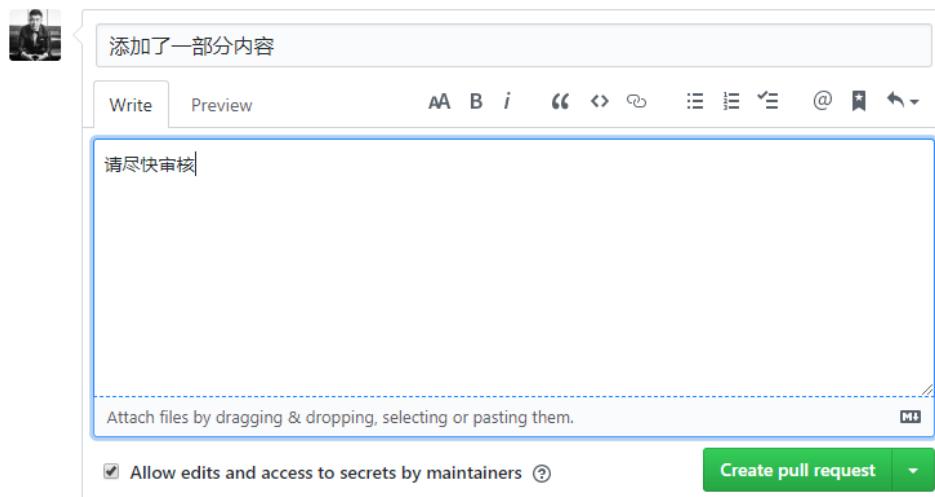
- 2) 修改代码之后如果想合并到作者那里，需要让作者审核，点击 Pull requests→New pull request



- 3) 点击 Create pull request



- 4) 填入标题、描述后点击 Create pull request



- 5) 你创建了 pull request 之后作者会收到 Pull requests 信息，作者可以选择拒绝和接受你的请求

第 4 章 在 Idea 中使用 Git

4.1 安装 Git 核心程序

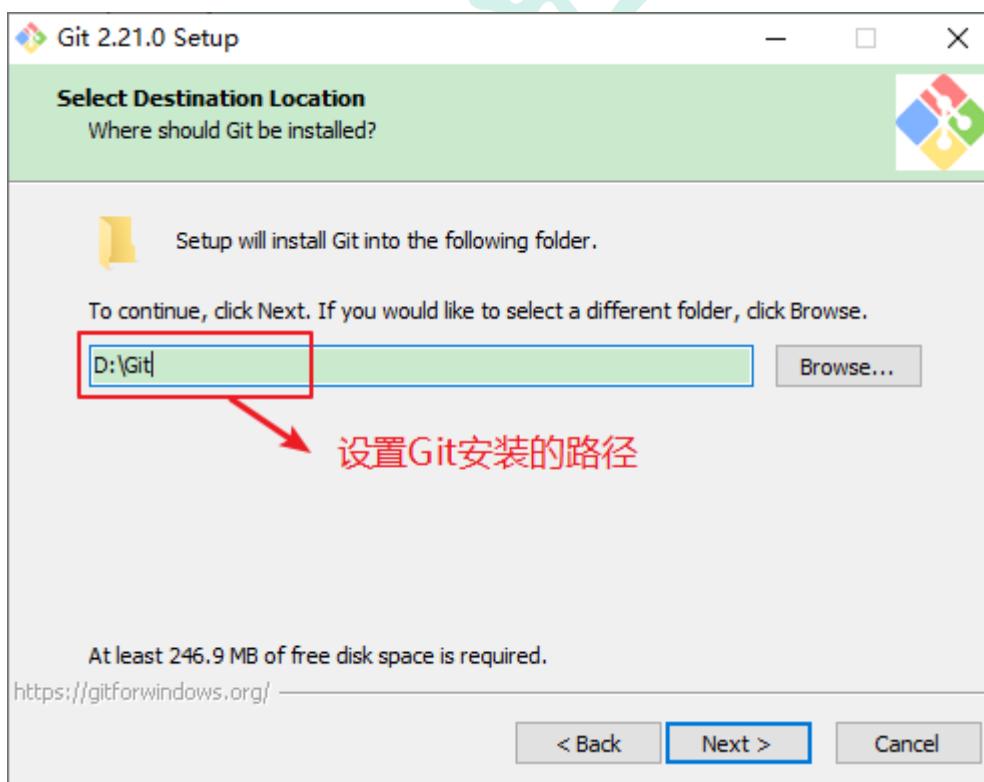
根据自己的电脑操作系统从 Git 官网 <https://git-scm.com/> 下载对应的 Git 核心程序。

以 git-2.21.0 为例说明安装步骤：

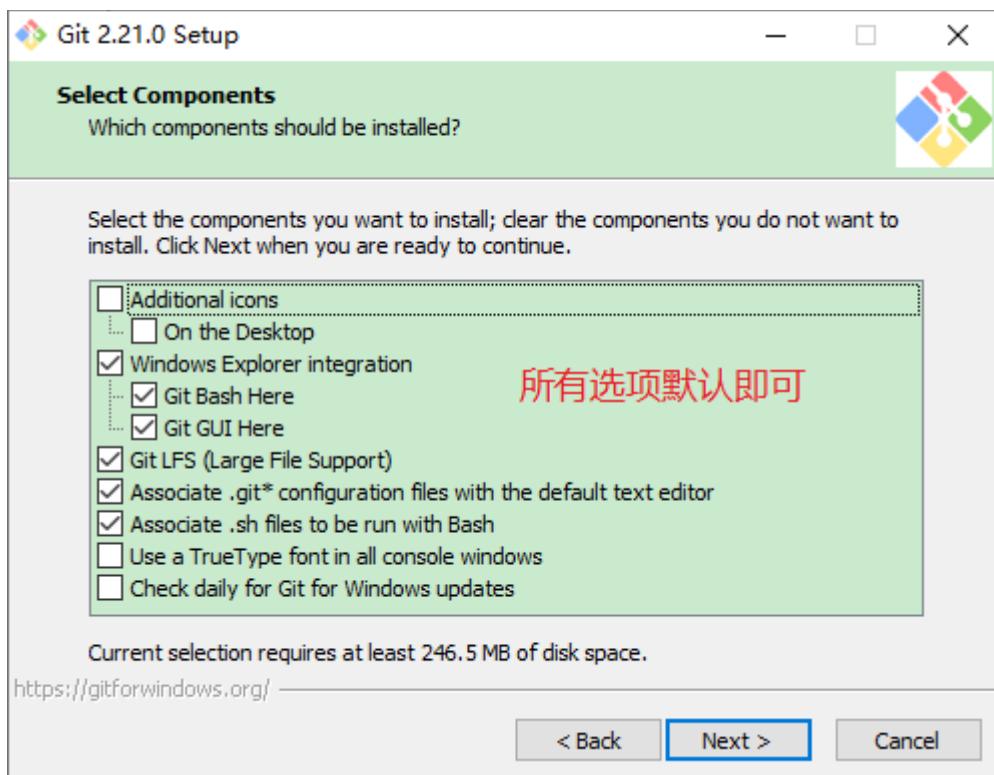
- 1) 双击 Git-2.21.0-64-bit.exe



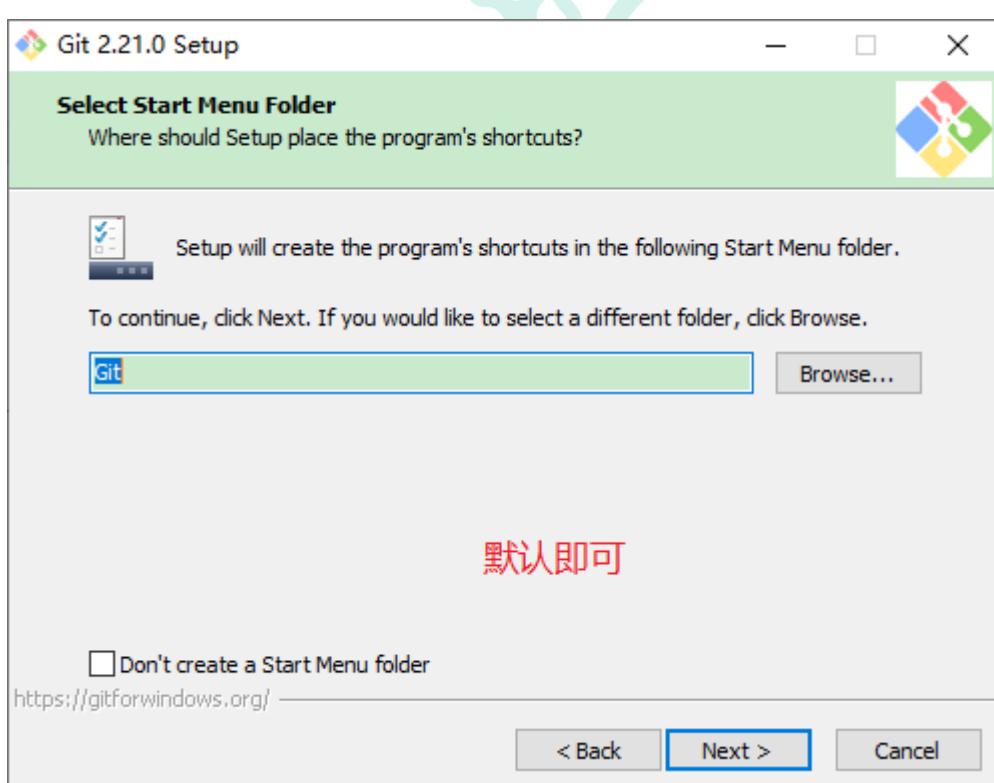
2) 点击 Next 设置安装路径



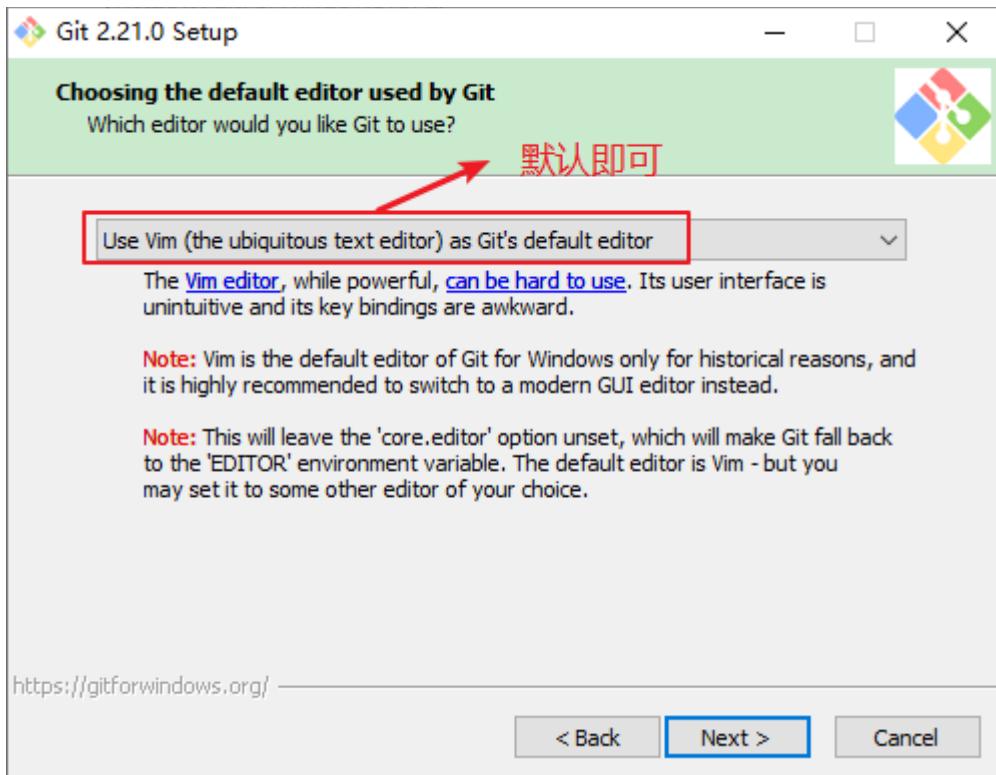
3) 点击 Next



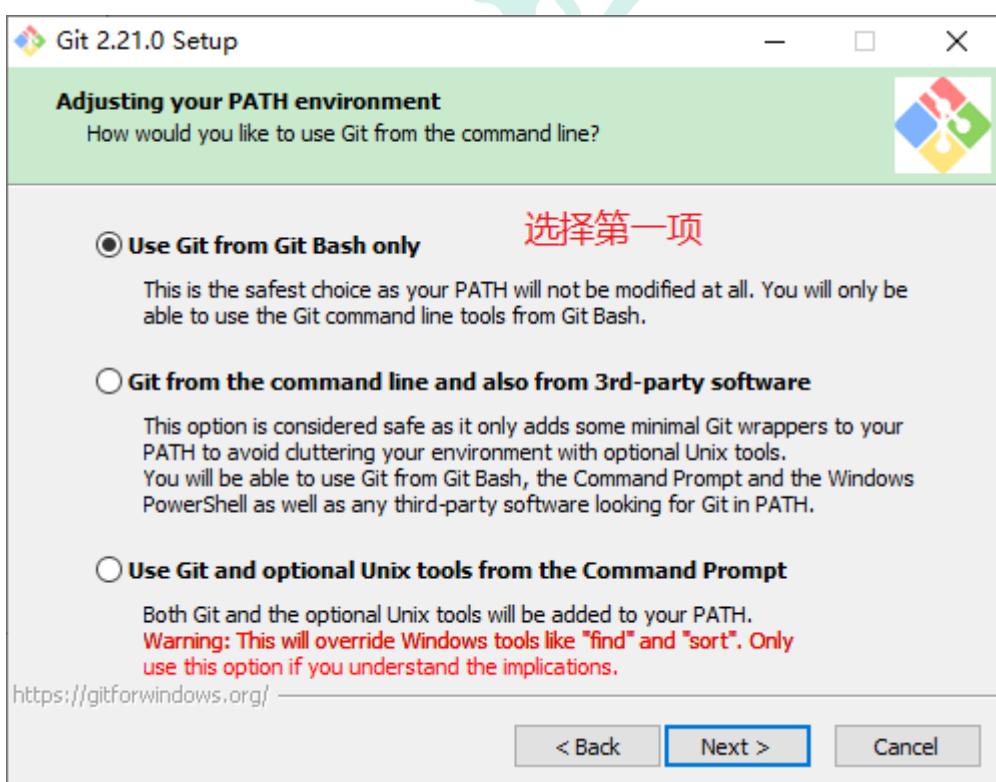
4) 点击 Next



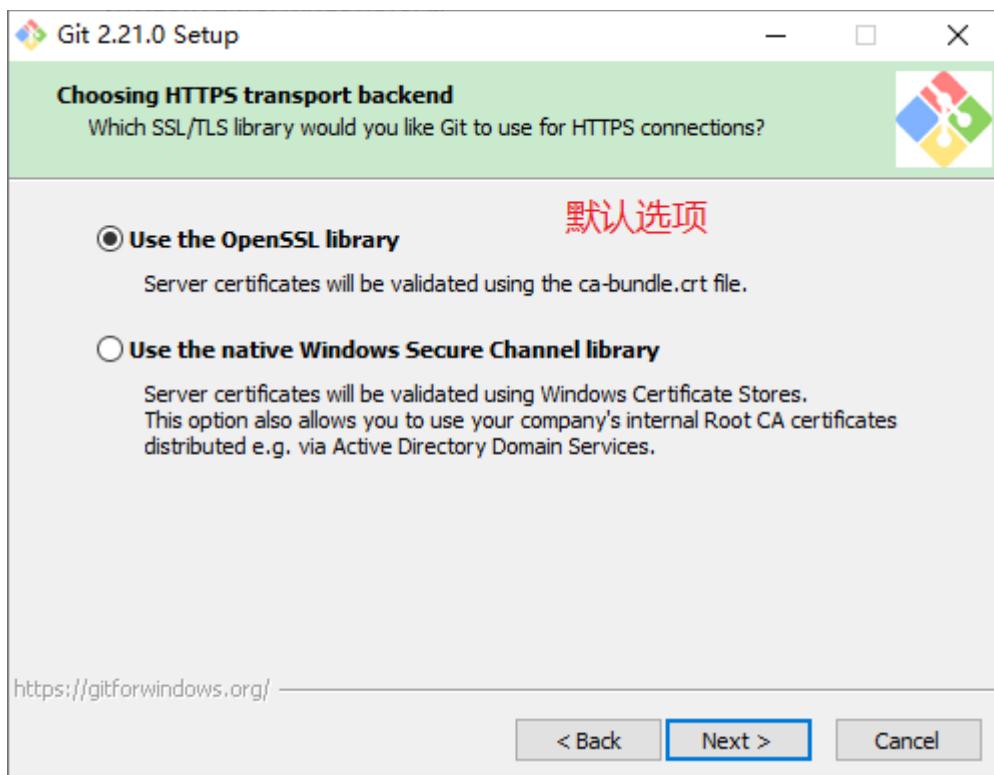
5) 点击 Next, 选择默认的编辑器



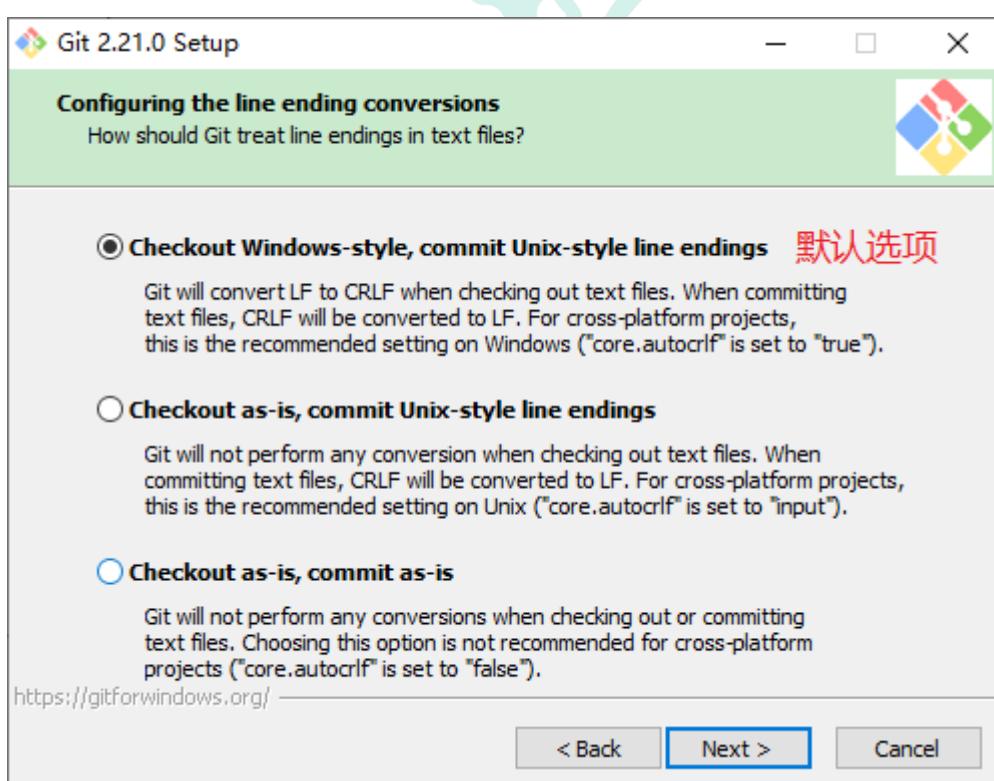
6) 点击 Next, 选择第一项



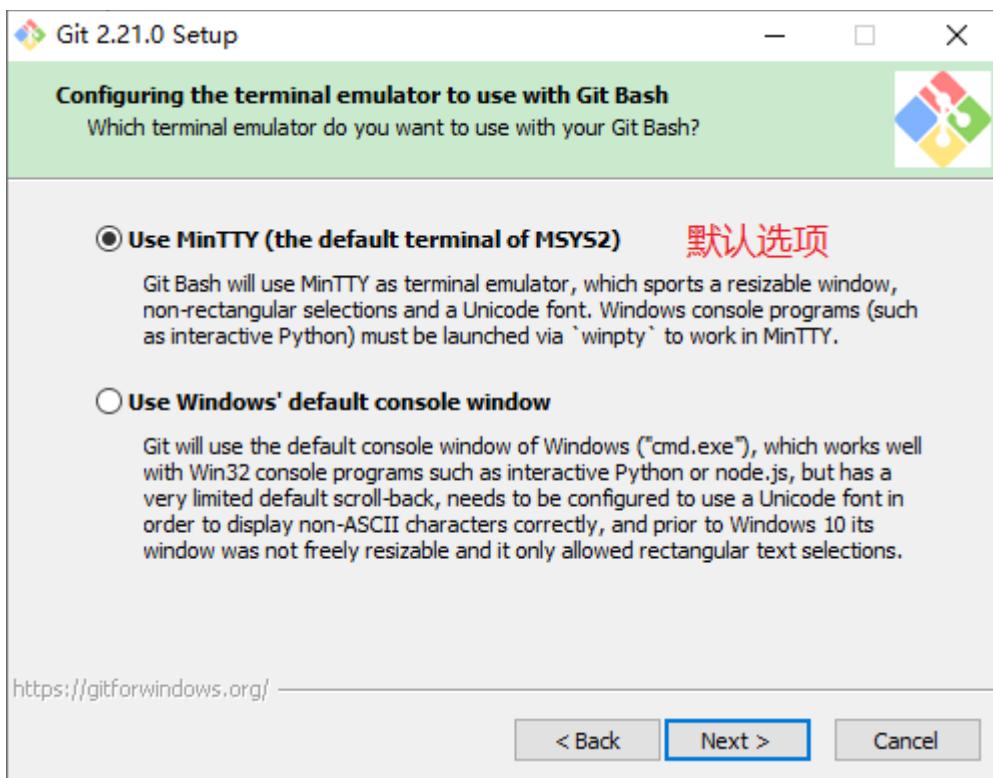
7) 点击 Next



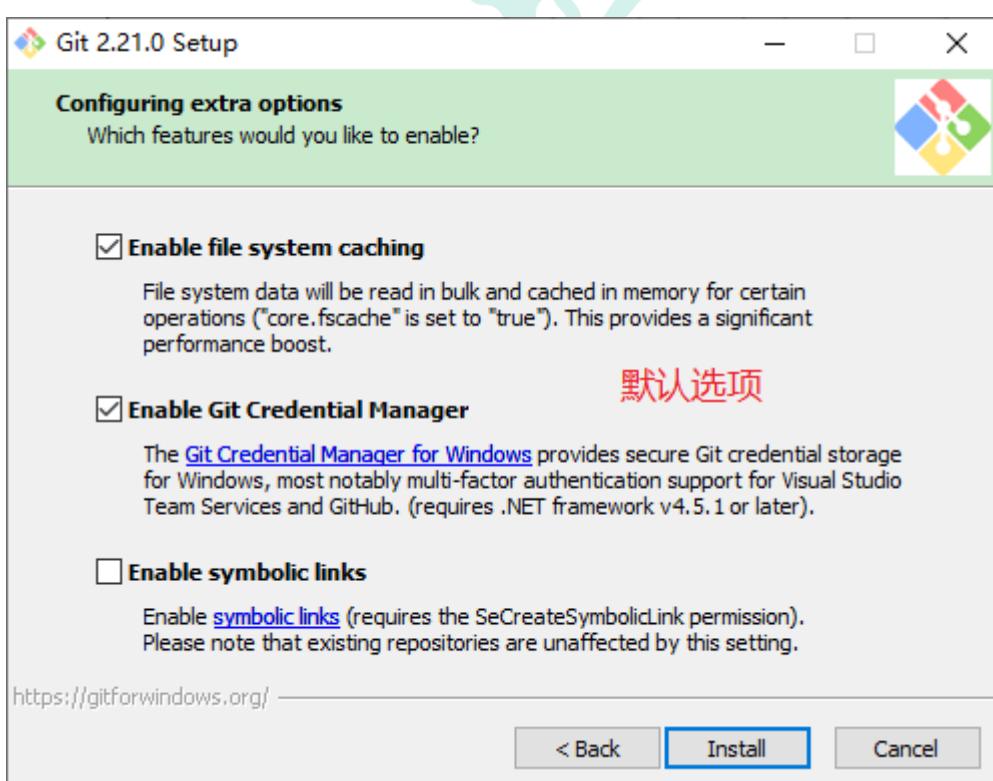
8) 点击 Next



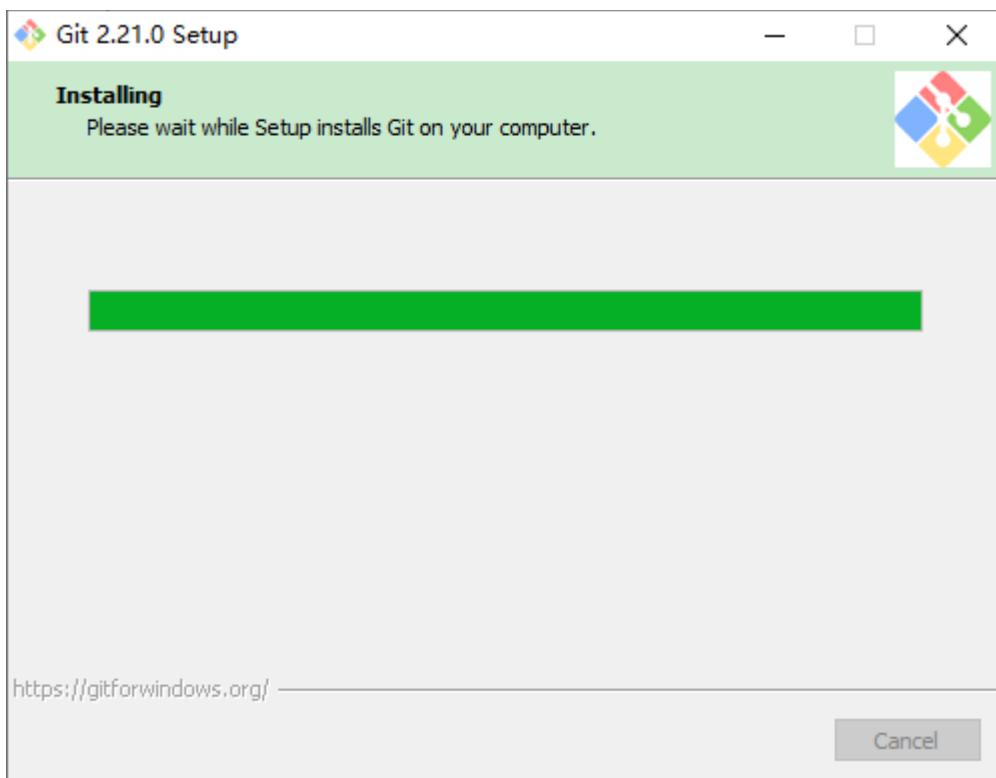
9) 点击 Next



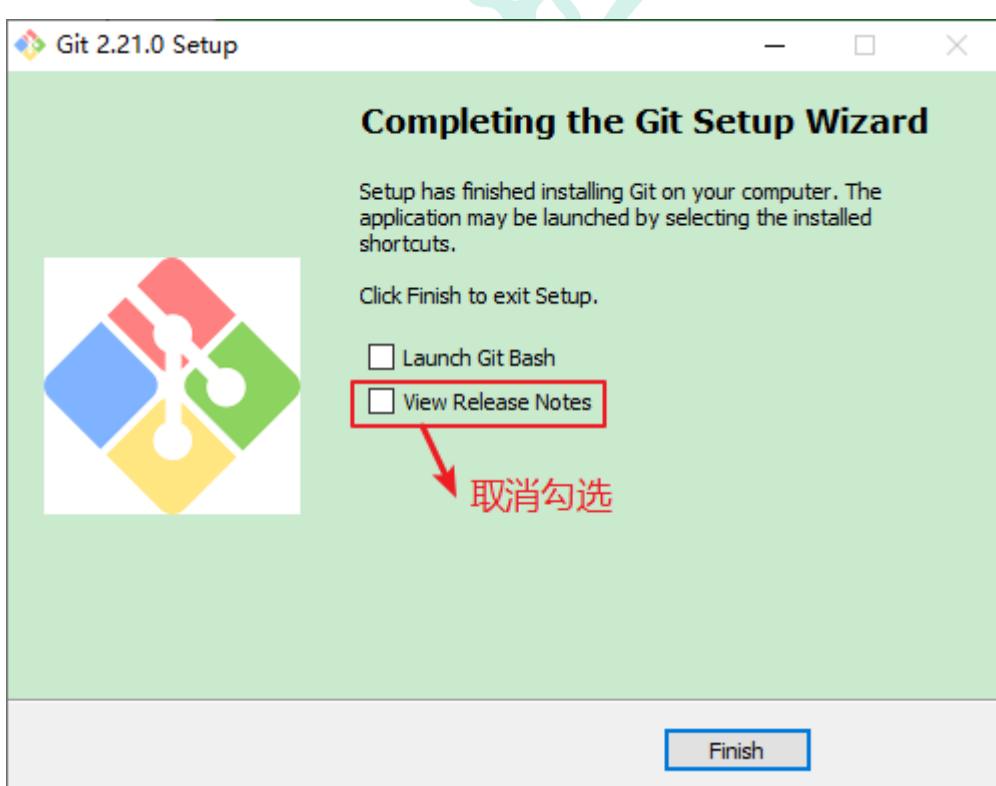
10) 点击 Next



11) 点击 Install 开始安装



12) 点击 Finish 安装完成

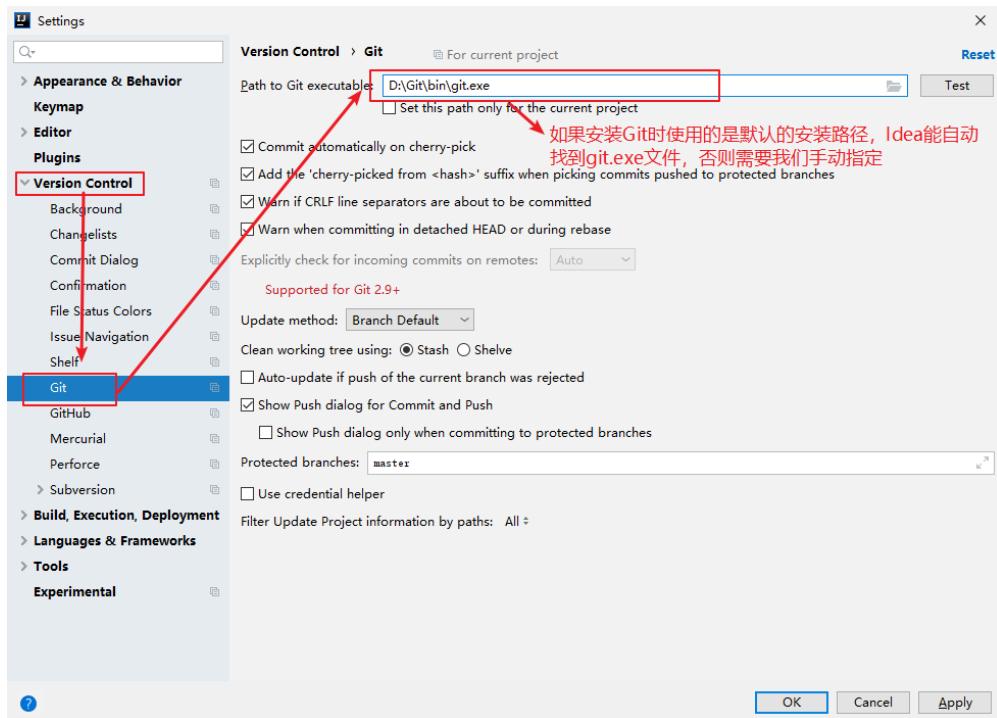


4.2 全局配置

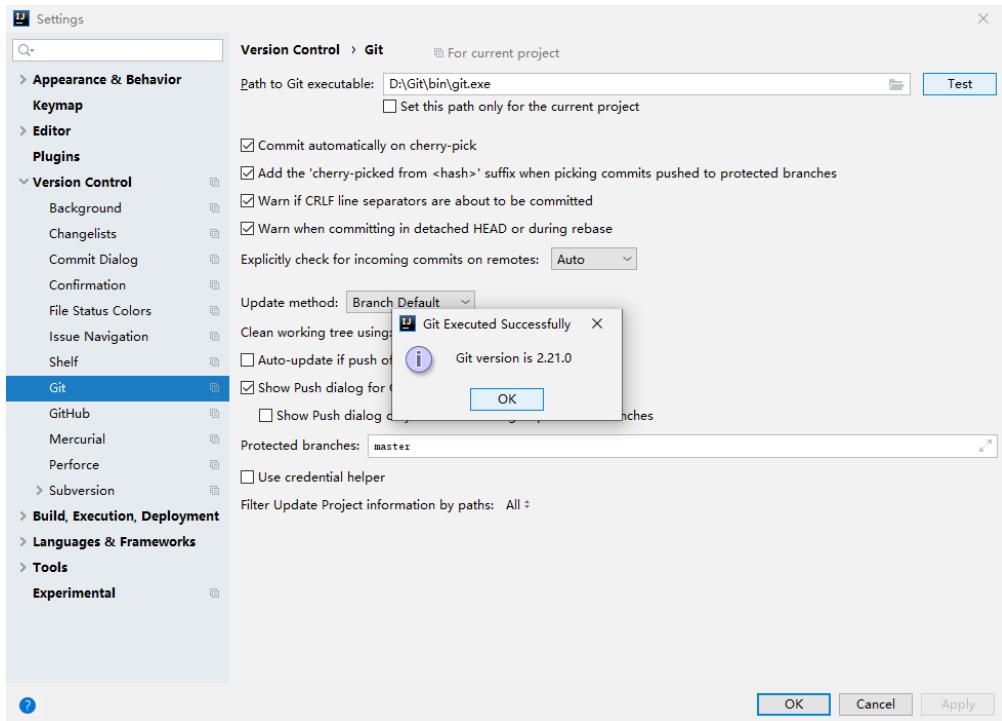
4.2.1 配置 Git 核心程序

1) 配置 git.exe 执行文件

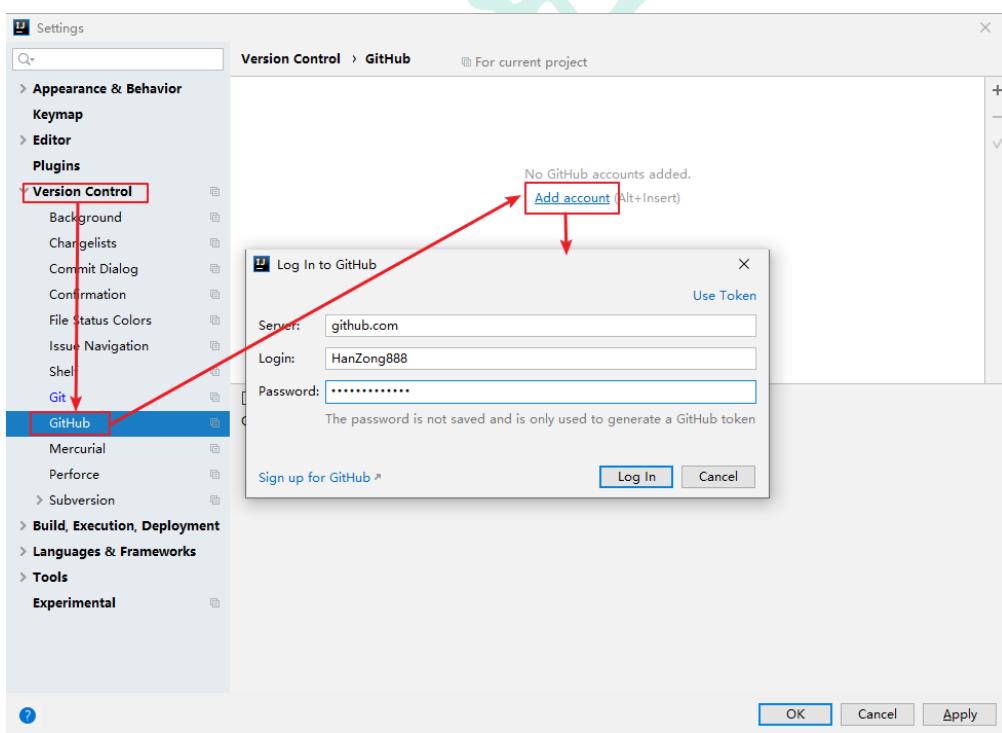
点击工具栏中的 settings→Version Control→Git

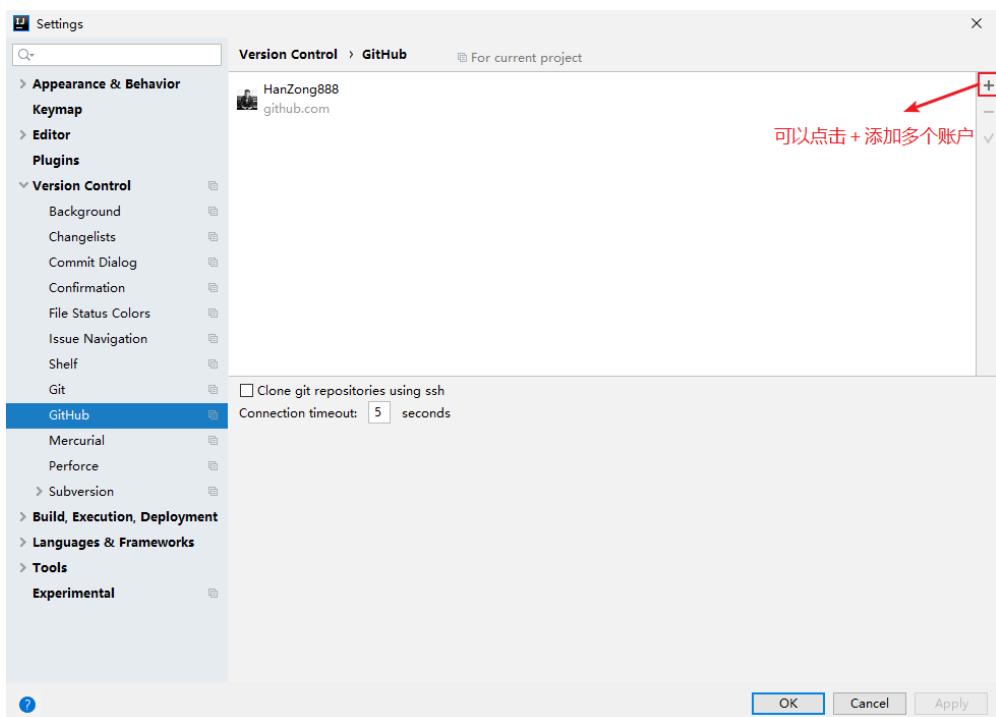


2) 点击 Test 测试



4.2.2 配置 GitHub 账户



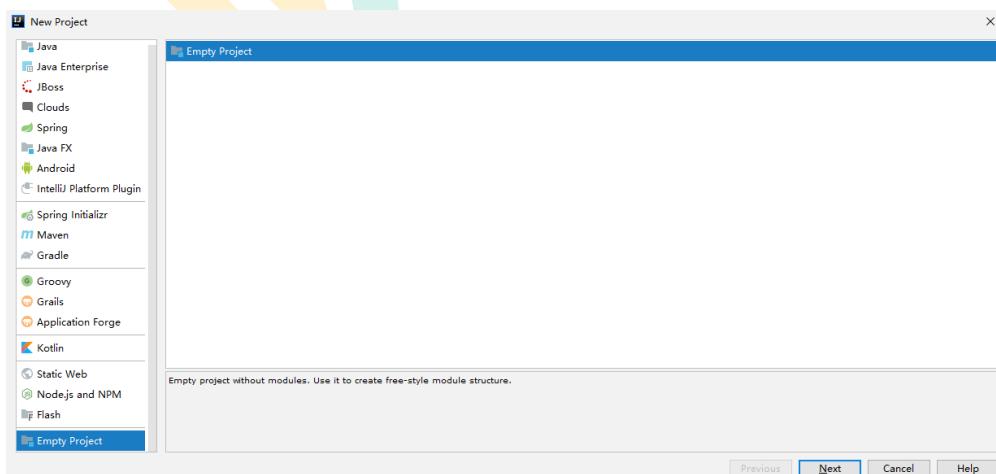


3) 点击 OK 完成配置

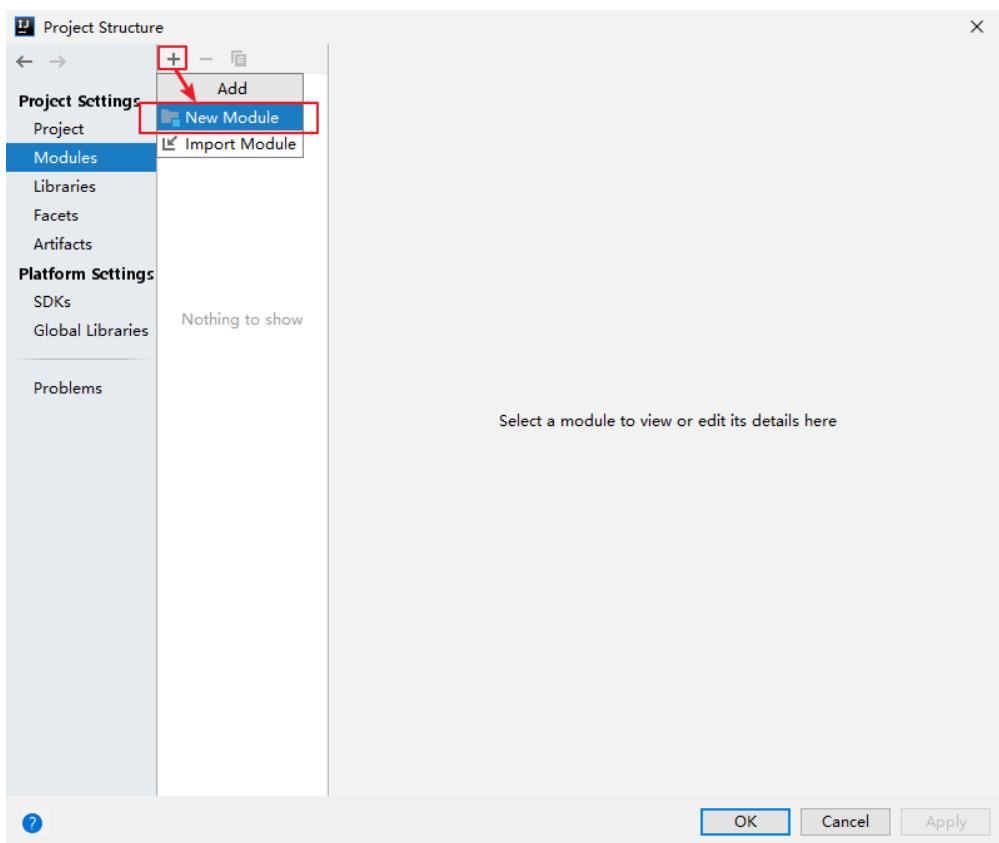
4.3 创建本地库

4.3.1 新建本地库

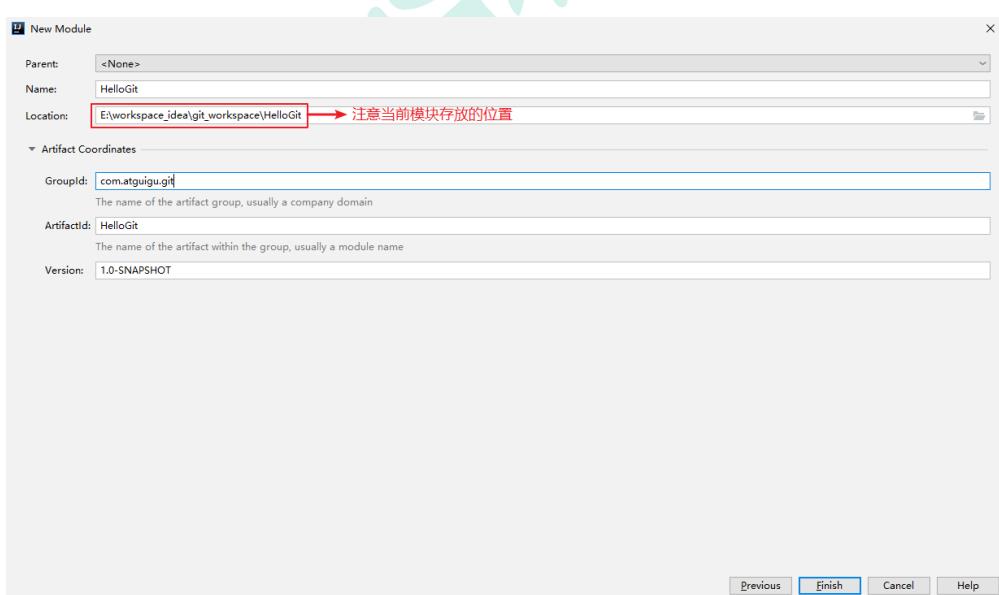
1) 创建一个 Empty Project



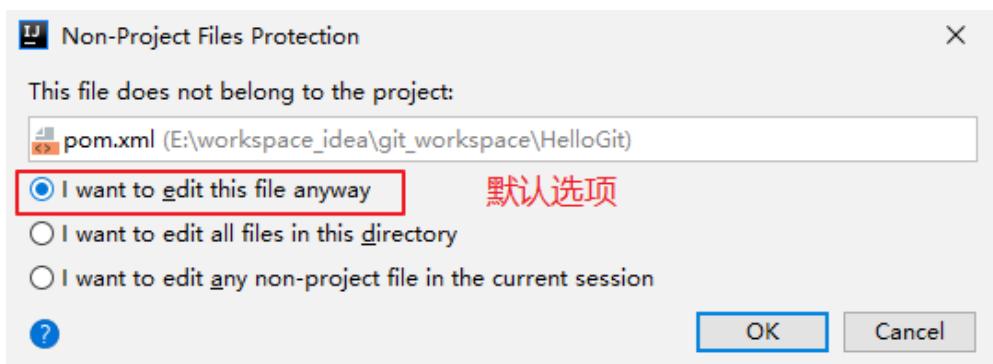
2) 在空工程 (Empty Project) 中添加模块 (Modules)



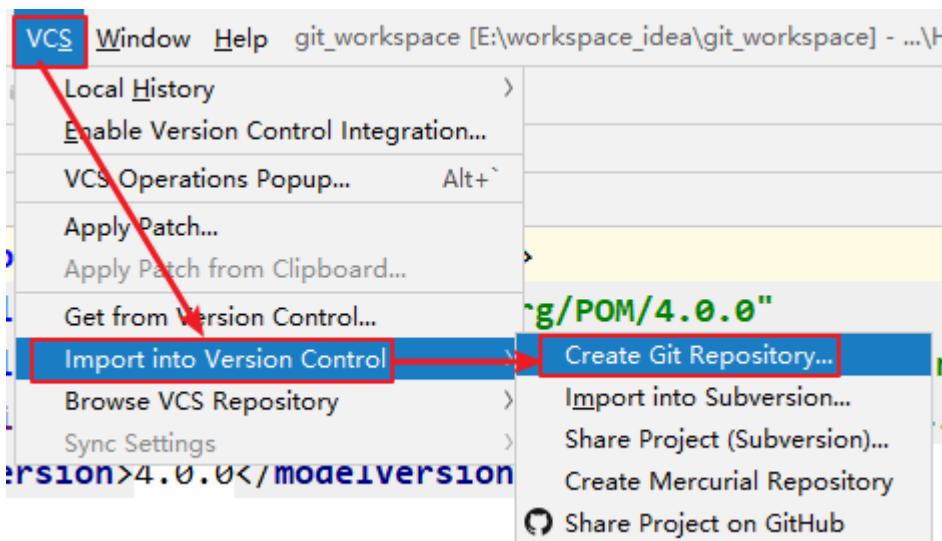
3) 添加一个 Maven 模块



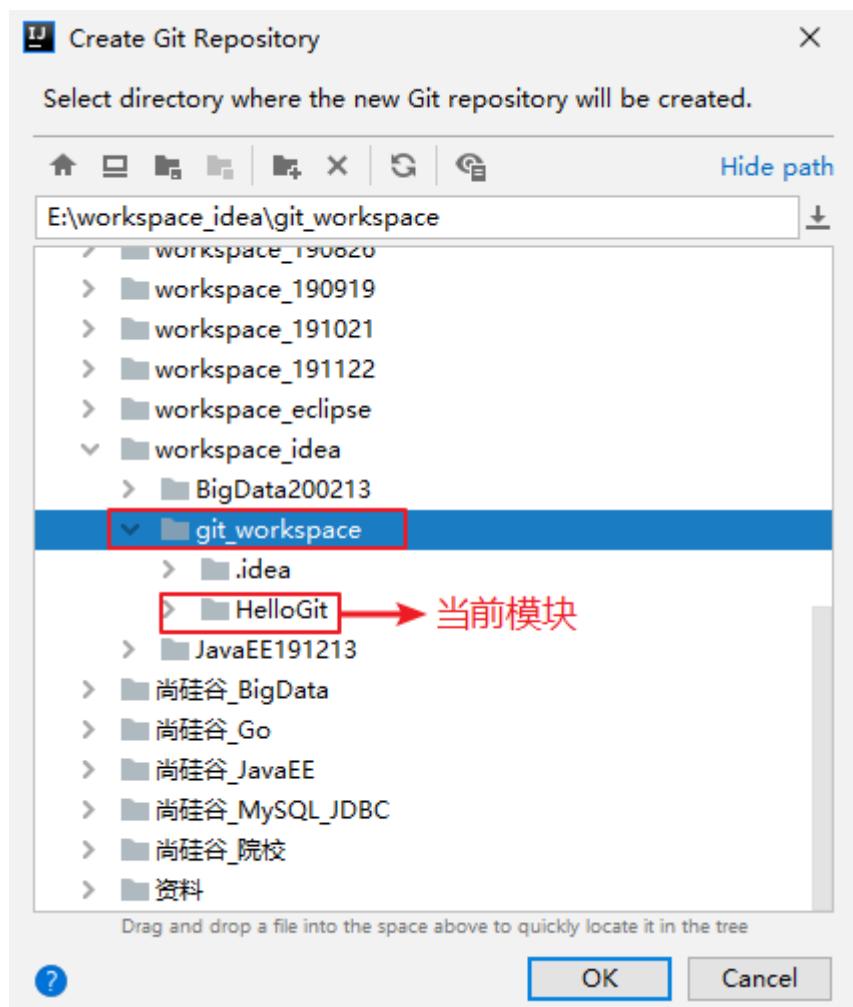
4) 选择我无论如何都想编辑这个文件



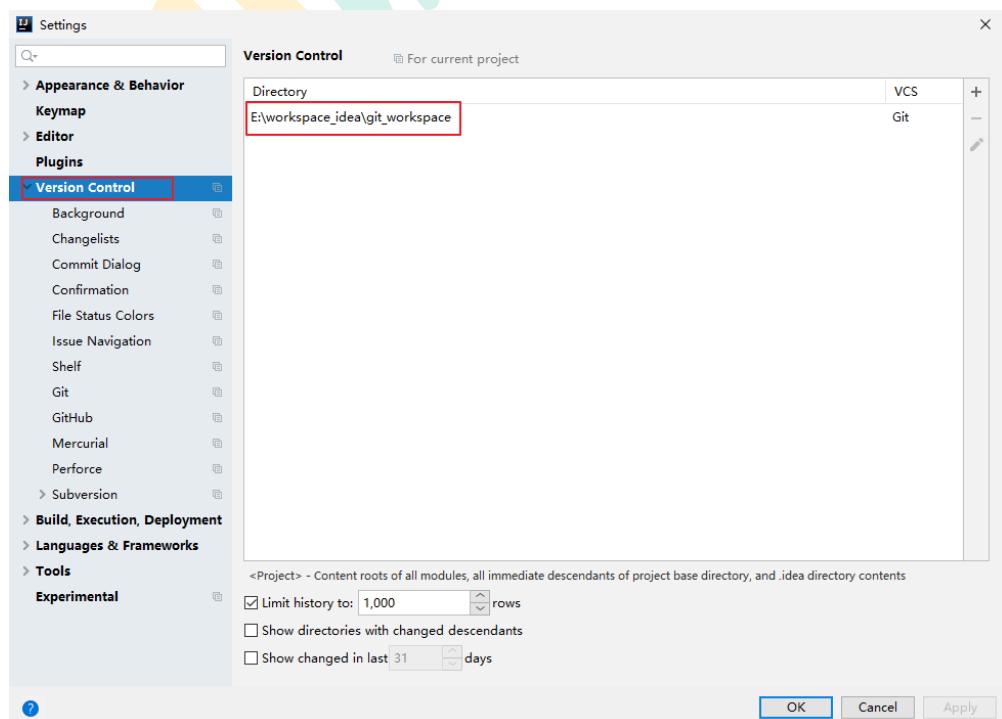
5) 创建本地库



6) 选择当前模块的上一级目录



7) 点击 OK 本地库创建成功

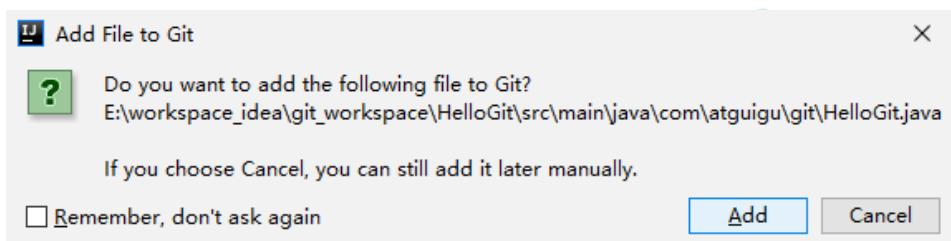


此电脑 > Work (E:) > workspace_idea > git_workspace >		
名称	修改日期	类型
.git	2020/5/4 22:18	文件夹
.idea	2020/5/4 22:18	文件夹
HelloGit	2020/5/4 22:13	文件夹

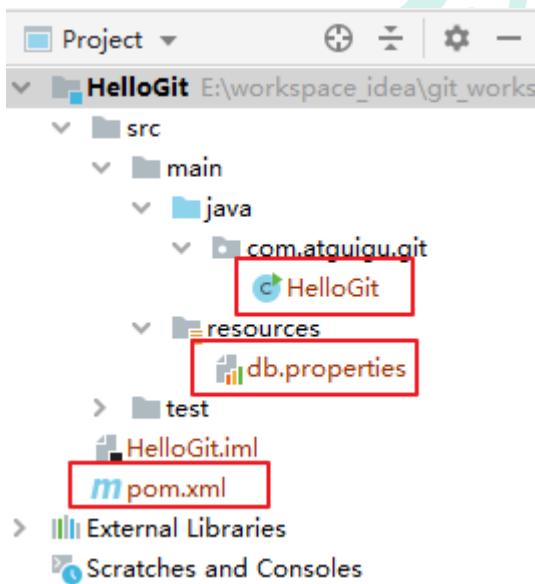
8) 同时工具栏会出现 Git 相关操作



9) 在 src/main/java 和 src/main/resources 目录下创建文件，创建了新文件之后会提示是否添加到暂存区

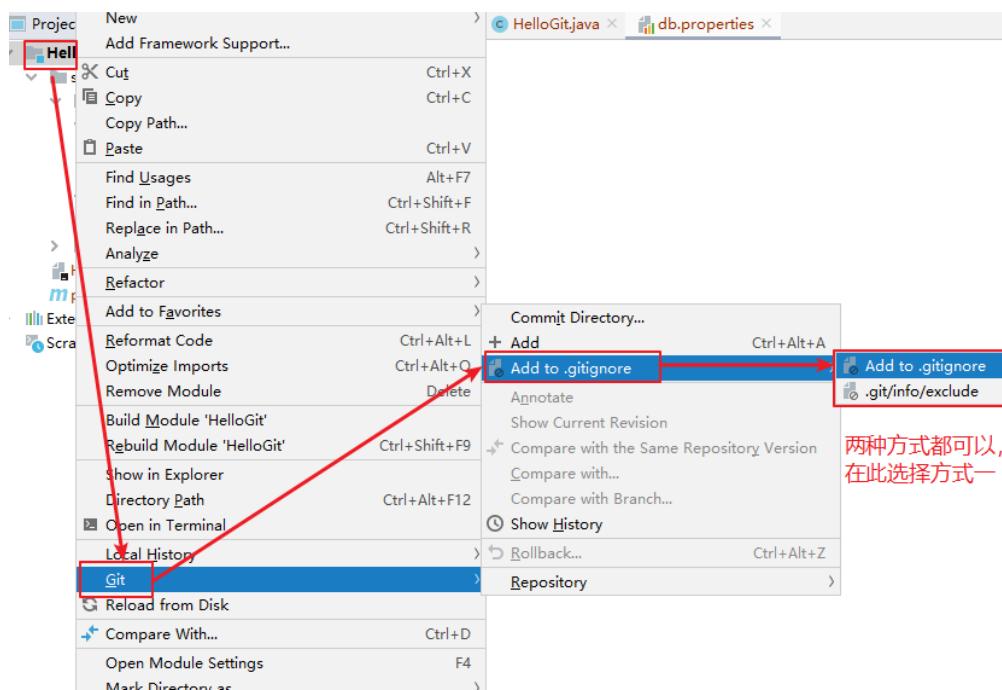


10) 如果点击了 Cancel，此时文件只存在于工作区，文件的状态如下图：



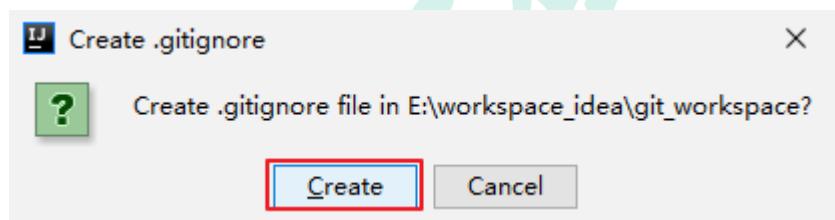
11) 设置忽略文件

在模块上右键，选择一种方式设置忽略的文件



Tips: 方式二只需要修改.git/info 目录下的 exclude 文件即可，不需要创建新的文件，所以建议大家选择这种方式。

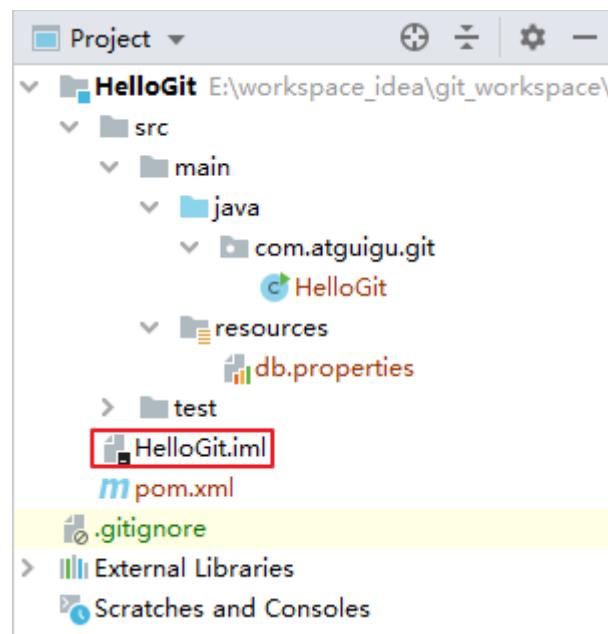
12) 弹出提示框，提示是否在当前工作区创建.gitignore 文件



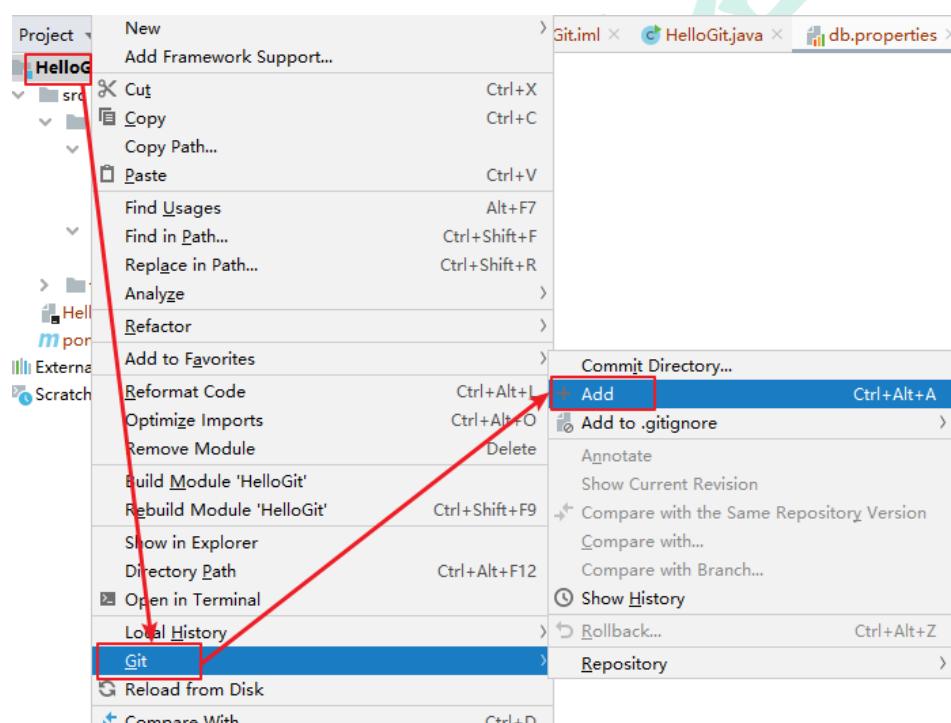
13) 点击 Create，添加如下内容

```
.idea  
*.iml
```

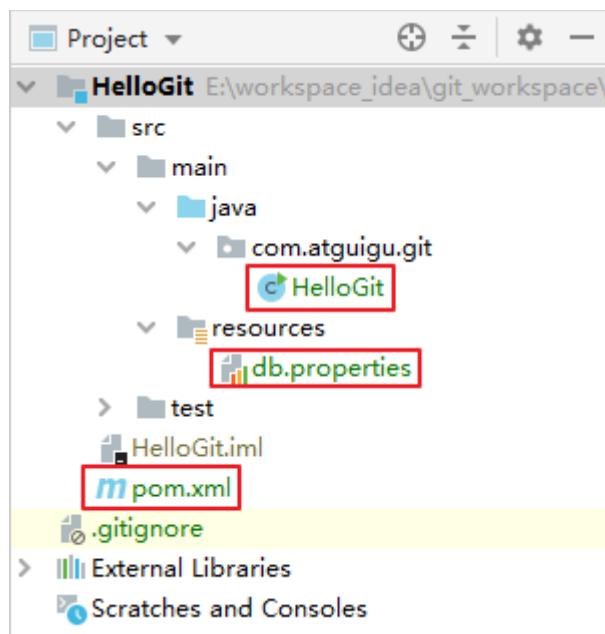
14) 创建.gitignore 文件之后发现被忽略的文件变成了灰色（有时候可能需要刷新模块或重启 Idea 才能看到）



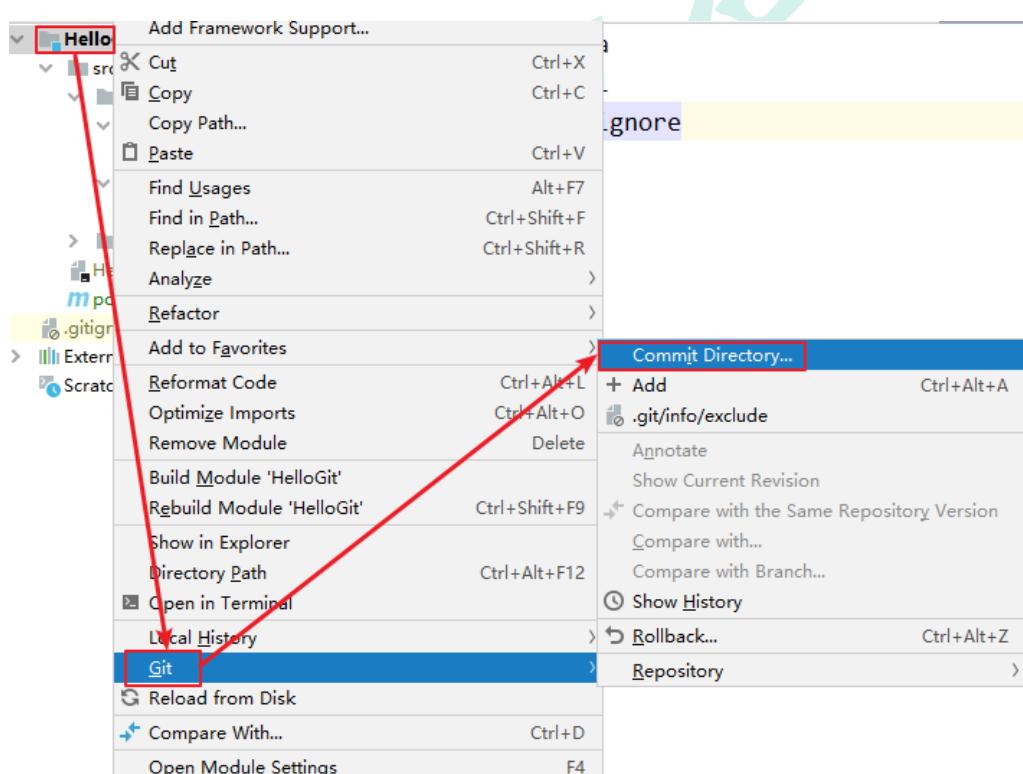
15) 在模块上右键将文件添加到暂存区



16) 添加到暂存区之后文件的状态如下图：



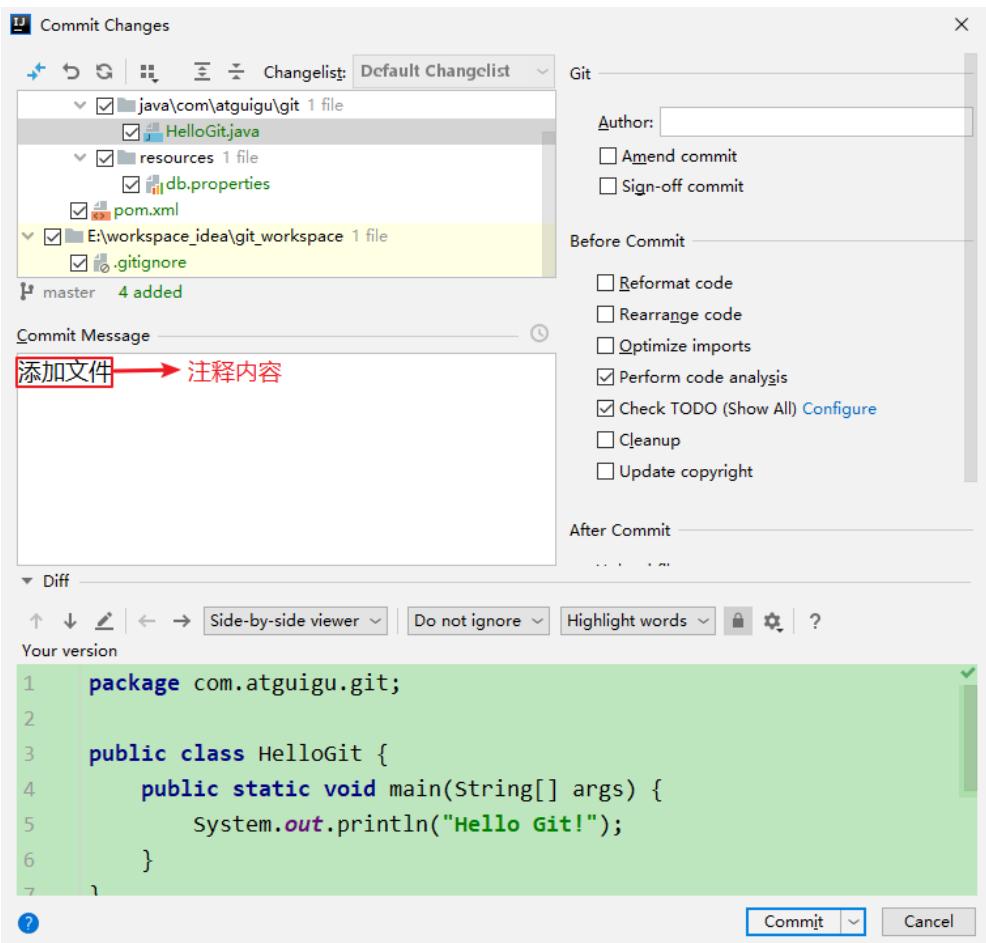
17) 在模块上右键或点击工具栏将文件添加到本地库



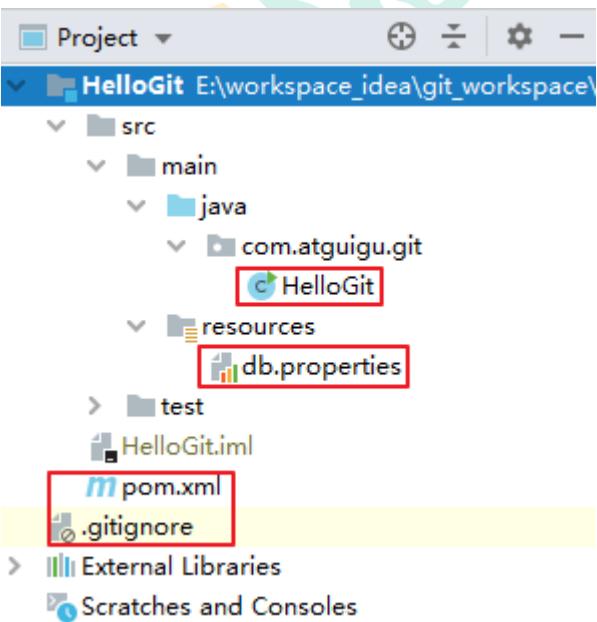
工具栏



18) 添加注释内容后提交



19) 提交到本地库之后文件的状态如下图：

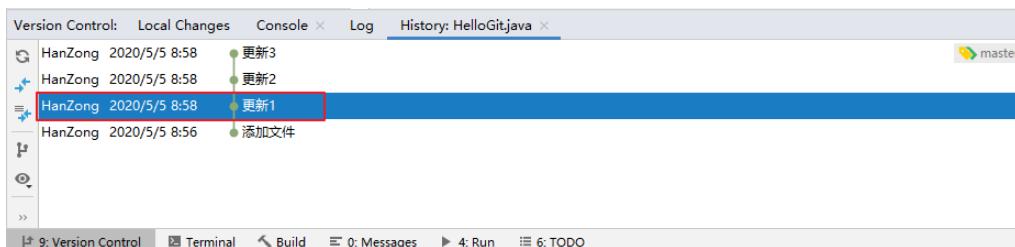


4.3.2 版本间切换

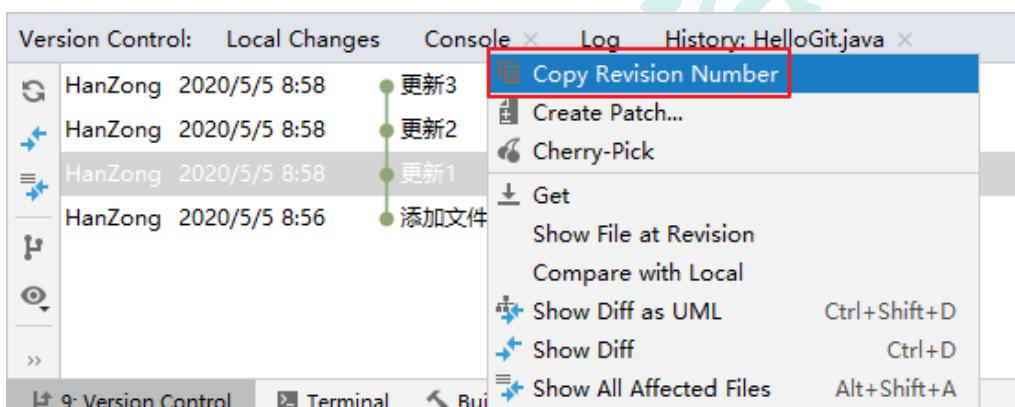
- 在模块上右键或者点击工具栏查看历史



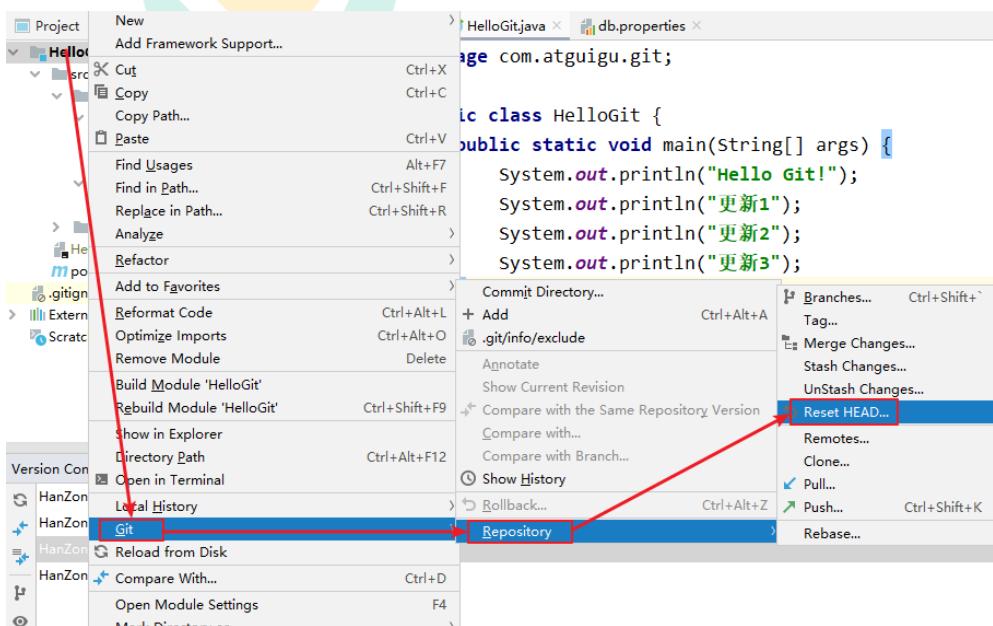
- 选择要切换的版本



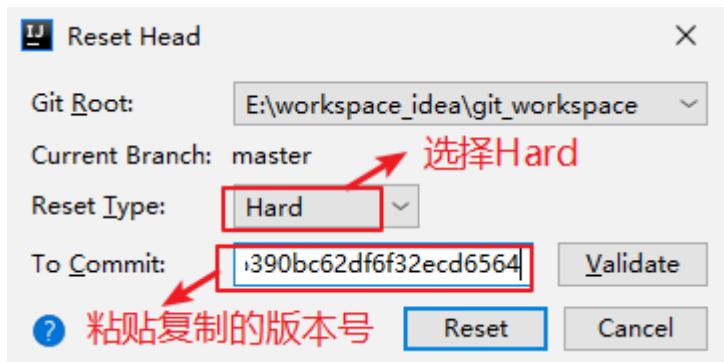
- 右键→Copy Revision Number



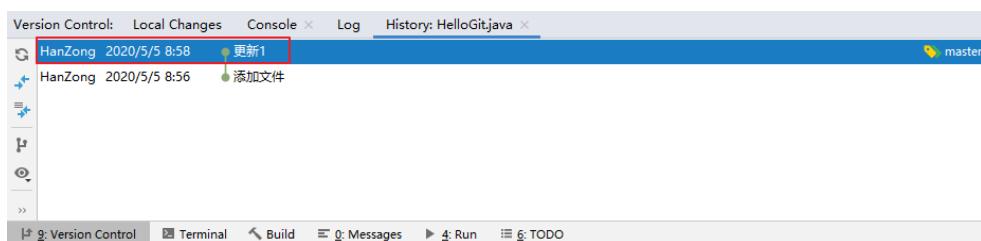
- 在模块上右键



5) 选择 Hard 并粘贴版本号

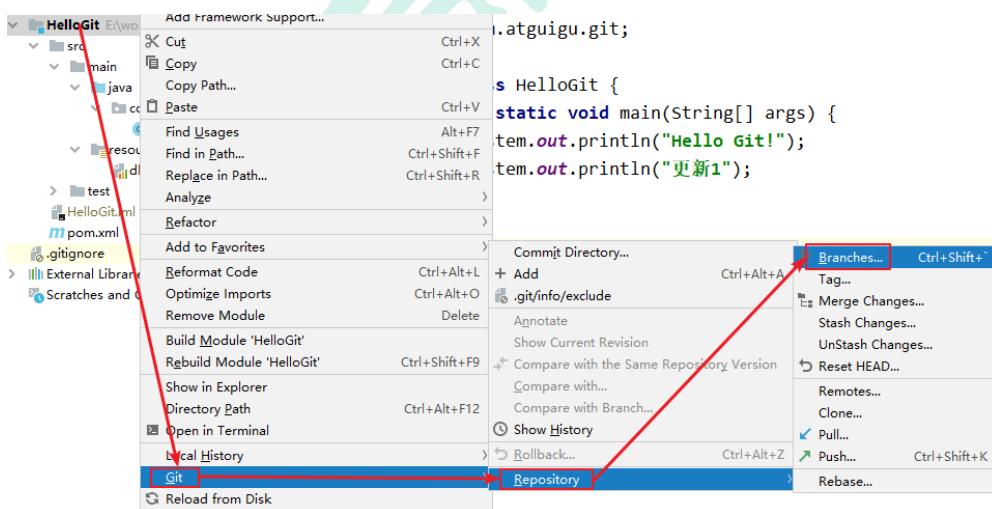


6) 版本切换成功

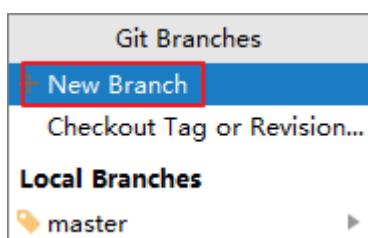


4.3.3 创建分支

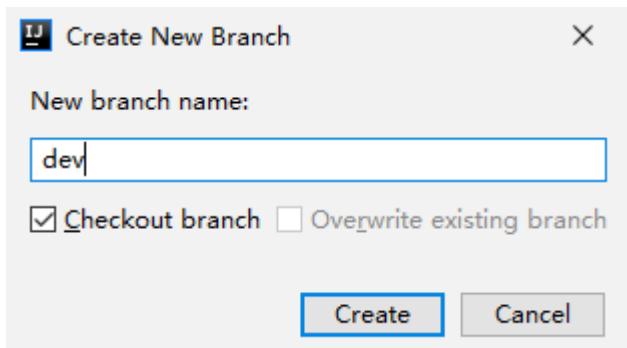
1) 在模块上右键



2) 点击 New Branch

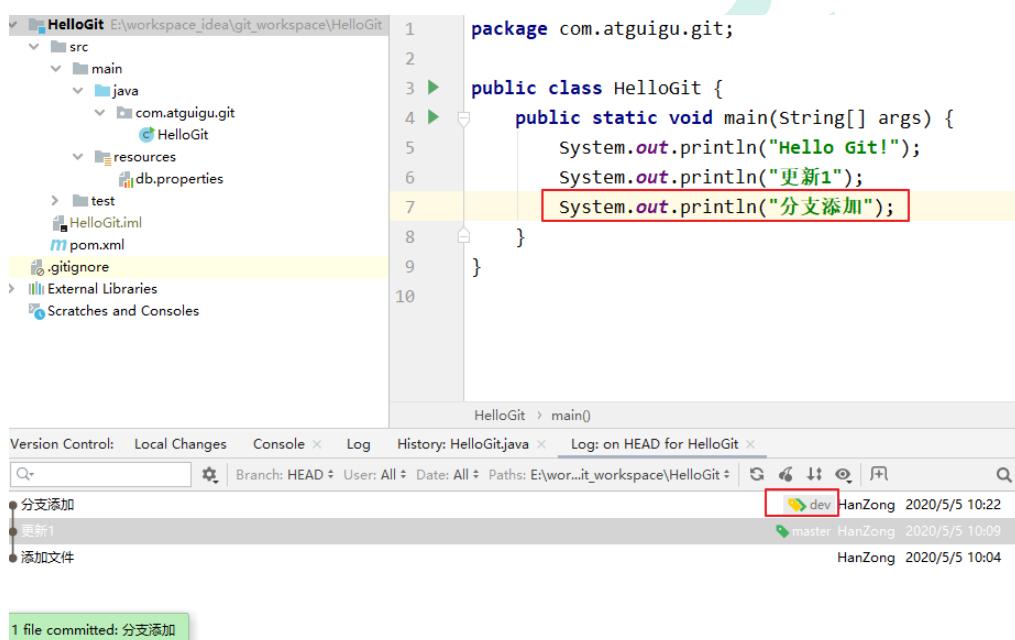


3) 给新分支命名



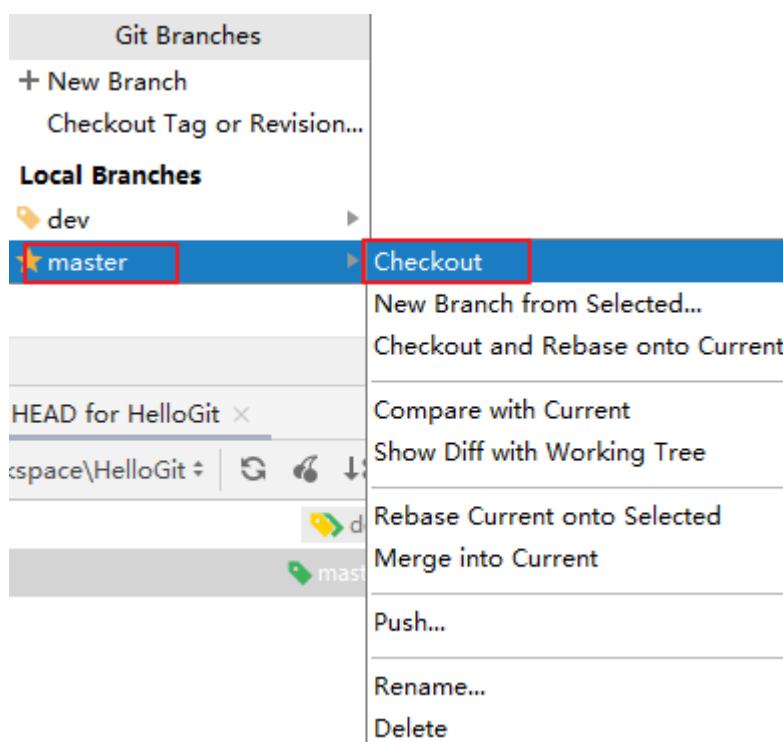
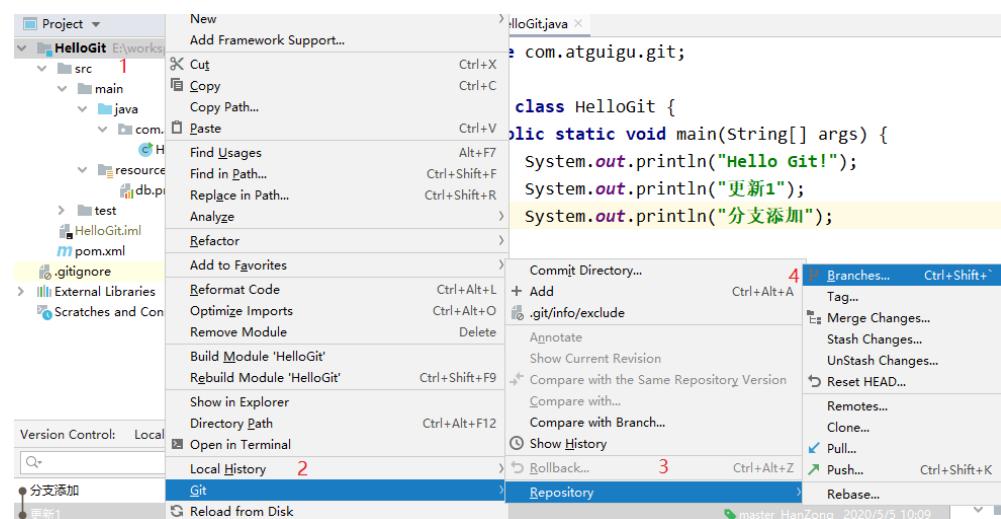
4) 点击 Create 后自动切换到新分支

5) 在新分支添加新的代码并提交

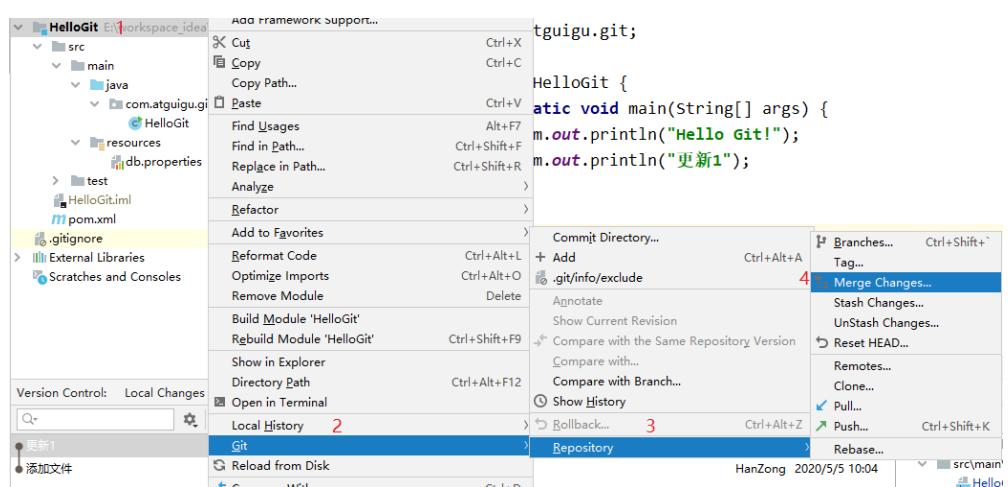


4.3.4 合并分支

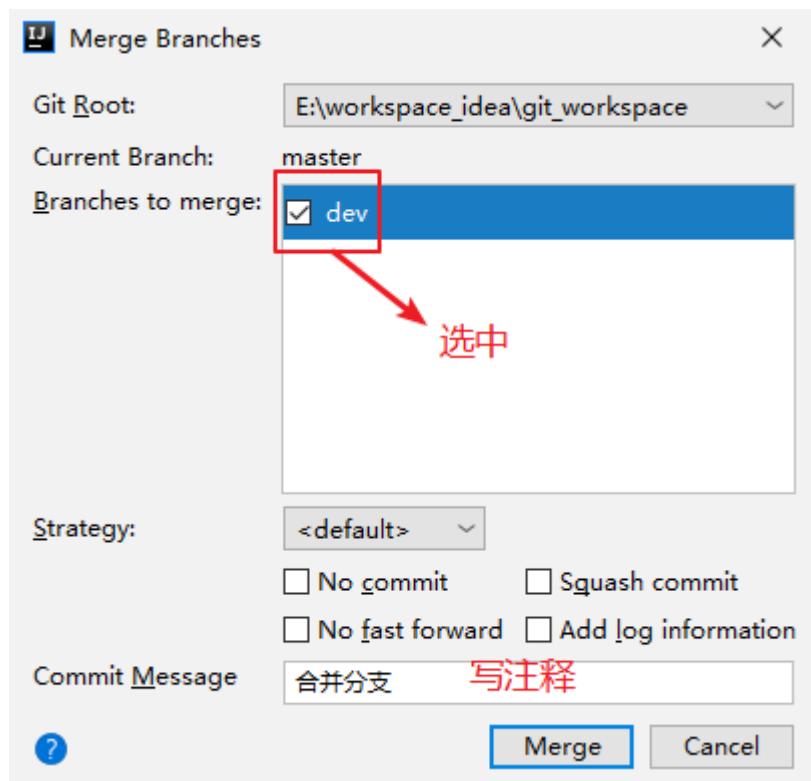
1) 在模块上右键切换到主干



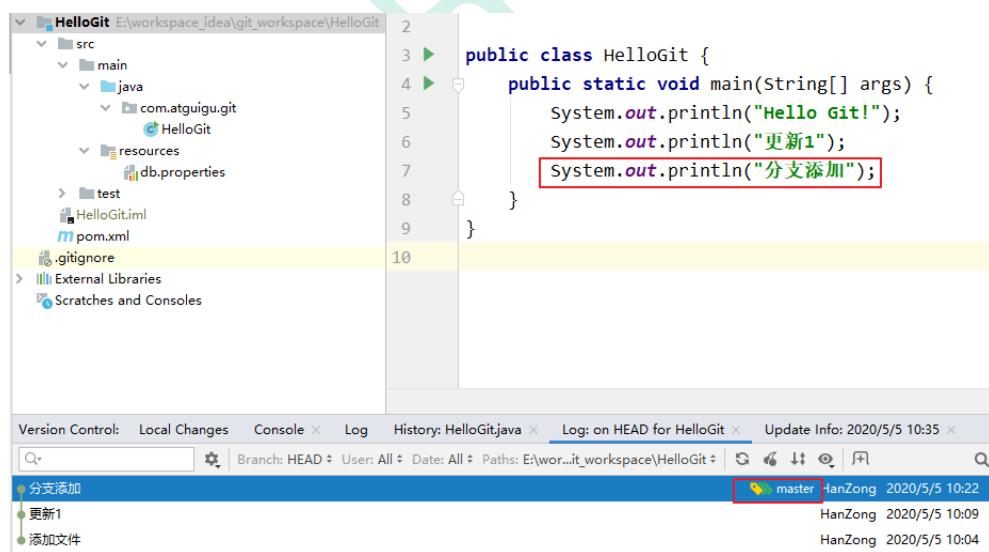
2) 在模块上右键选择合并改变



3) 选择要合并的分支



4) 合并成功



4.3.5 解决冲突

让主干和分支在同一个位置添加一行代码

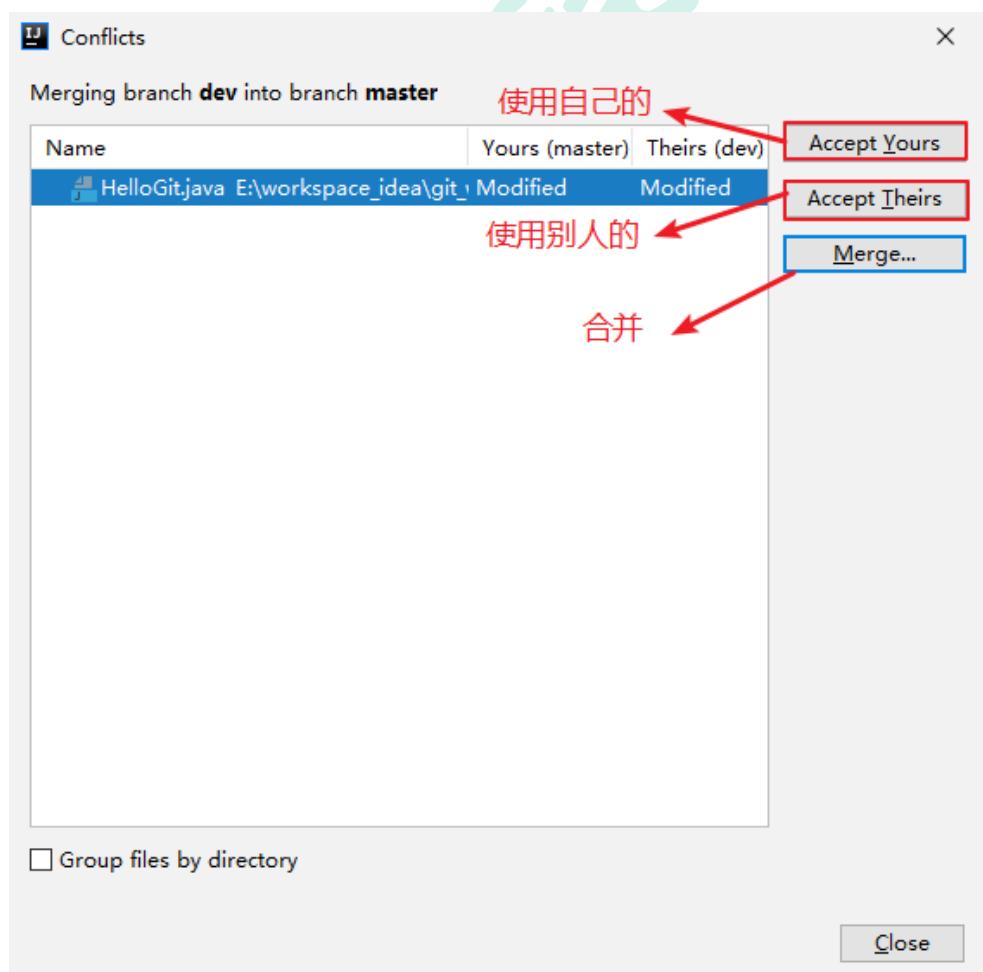
1) 分支添加内容，并添加到暂存区和本地库

```
1 package com.atguigu.git;
2
3 public class HelloGit {
4     public static void main(String[] args) {
5         System.out.println("Hello Git!");
6         System.out.println("更新1");
7         System.out.println("分支添加");
8         System.out.println("分支在第8行添加");
9     }
10 }
```

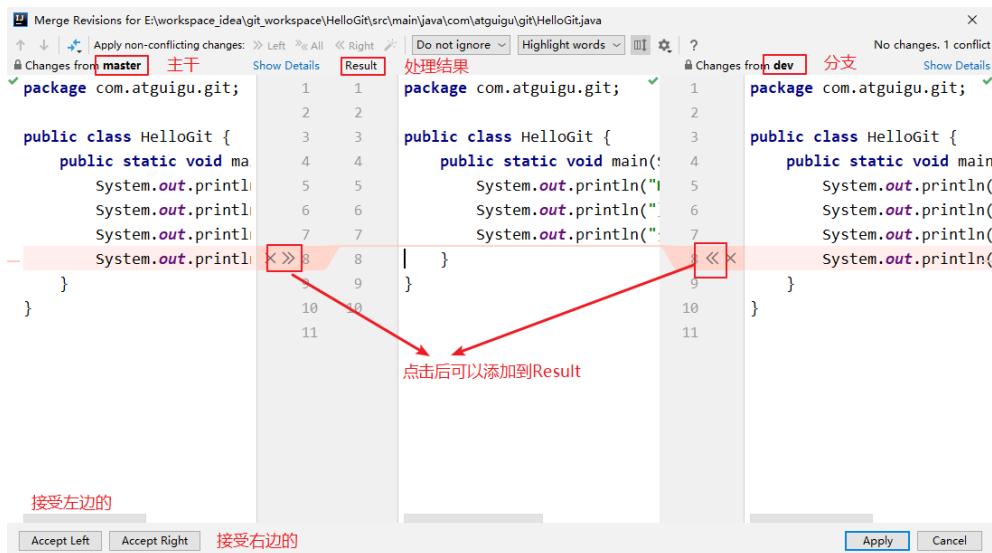
2) 主干添加内容，并添加到暂存区和本地库

```
1 package com.atguigu.git;
2
3 public class HelloGit {
4     public static void main(String[] args) {
5         System.out.println("Hello Git!");
6         System.out.println("更新1");
7         System.out.println("分支添加");
8         System.out.println("主干在第8行添加");
9     }
10 }
```

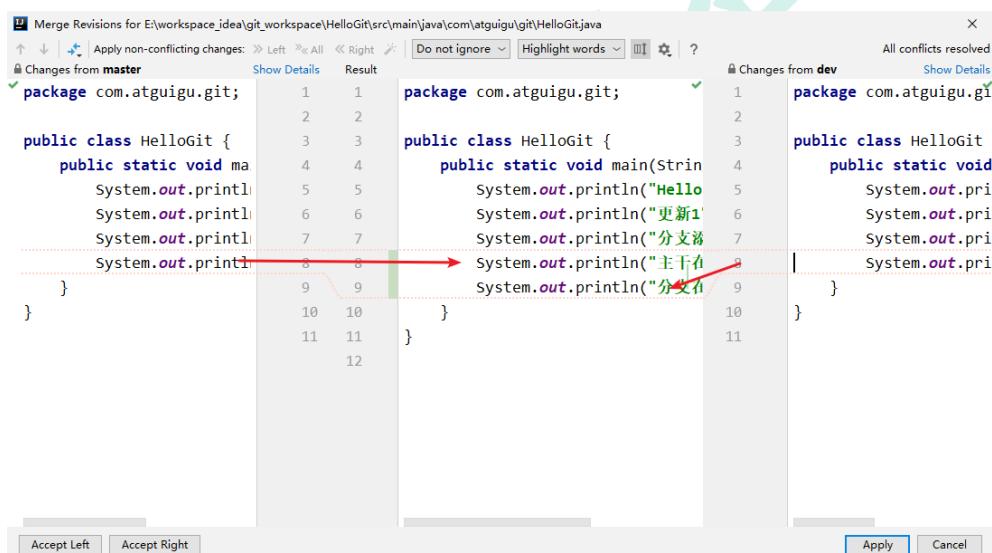
3) 在主干合并分支内容，出现冲突



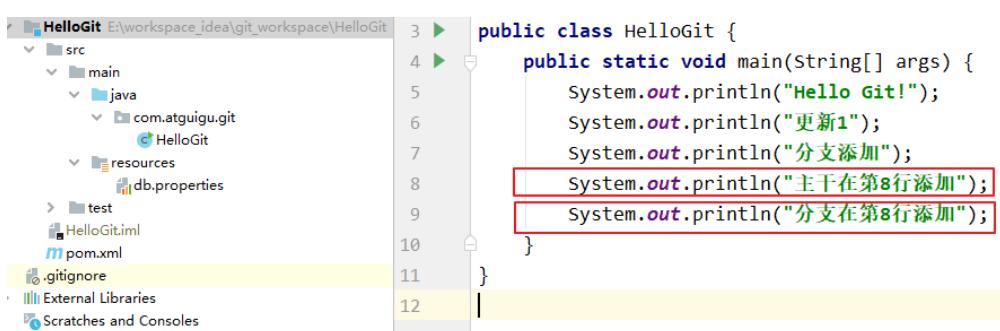
4) 选择合并，出现处理窗口



5) 处理之后



6) 点击 Apply 应用



4.4 将本地库上传到 GitHub

4.4.1 上传本地库

1) 在 GitHub 网站上创建仓库

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner: HanZong888 / Repository name *: helloGit

Great repository names are helloGit is available. Need inspiration? How about friendly-guacamole?

Description (optional):

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

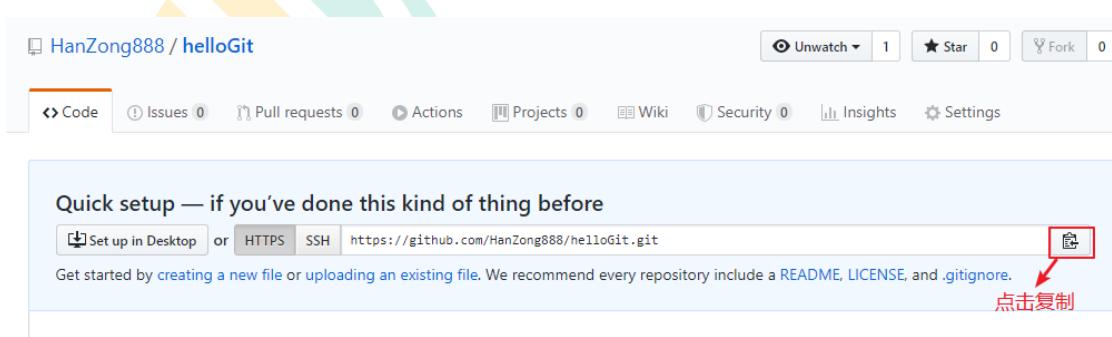
Skip this step if you're importing an existing repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None | ⓘ

Create repository

2) 复制仓库地址



HanZong888 / helloGit

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security 0 Insights Settings

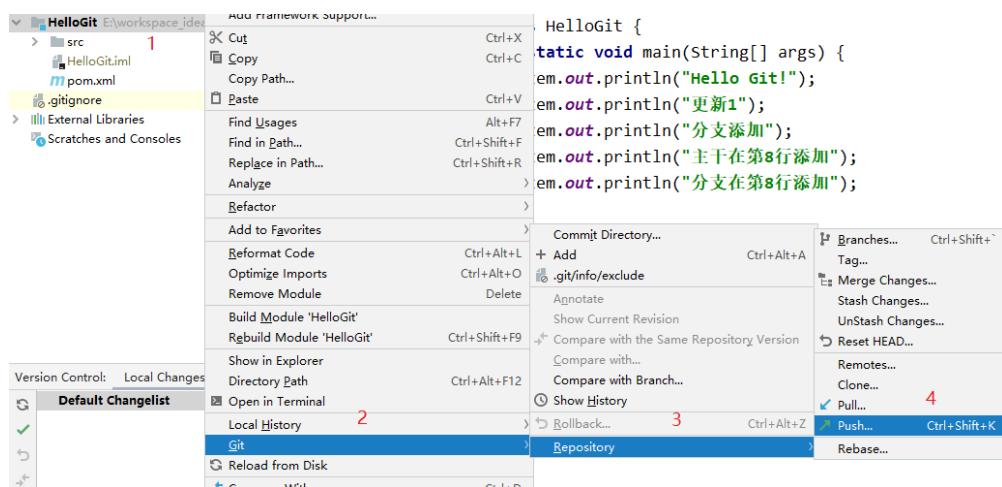
Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/HanZong888/helloGit.git

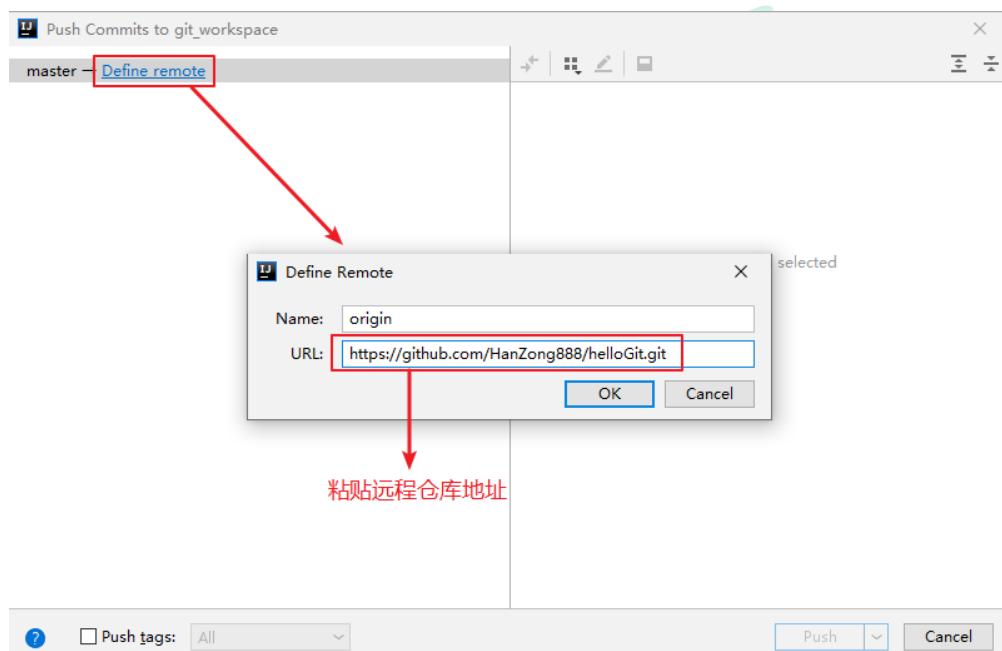
Get started by creating a new file or uploading an existing file. We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

点击复制

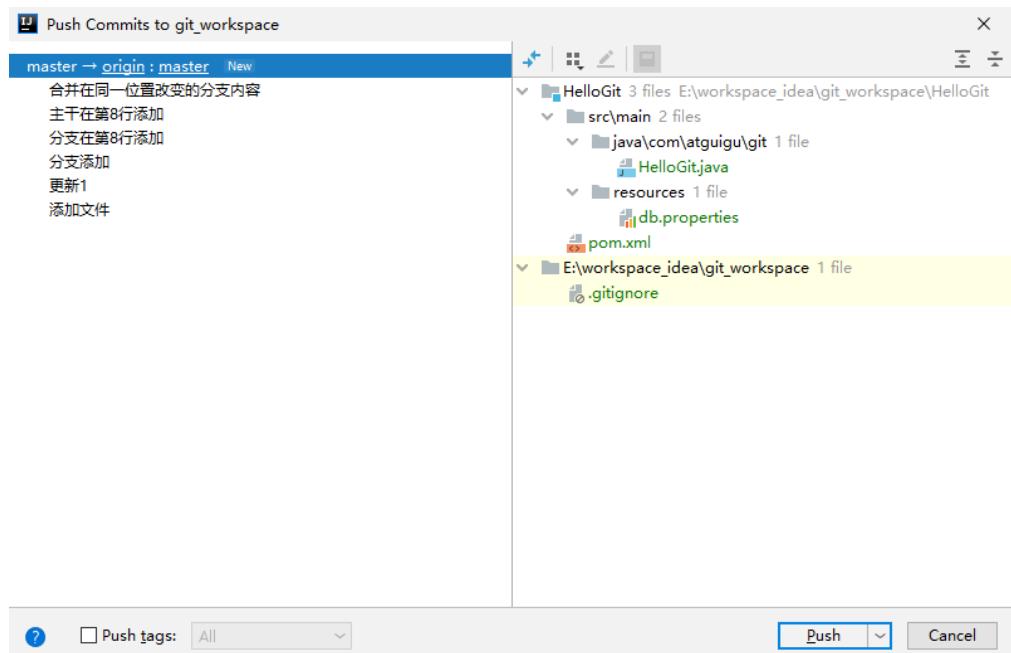
3) 在 Idea 中的模块上右键



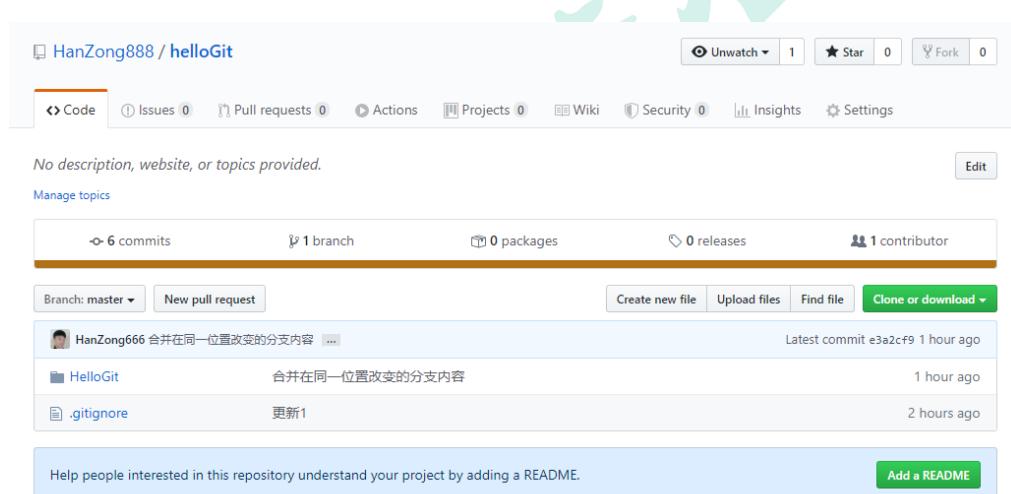
4) 设置远程地址别名



5) 点击 Push 推送到 GitHub 仓库



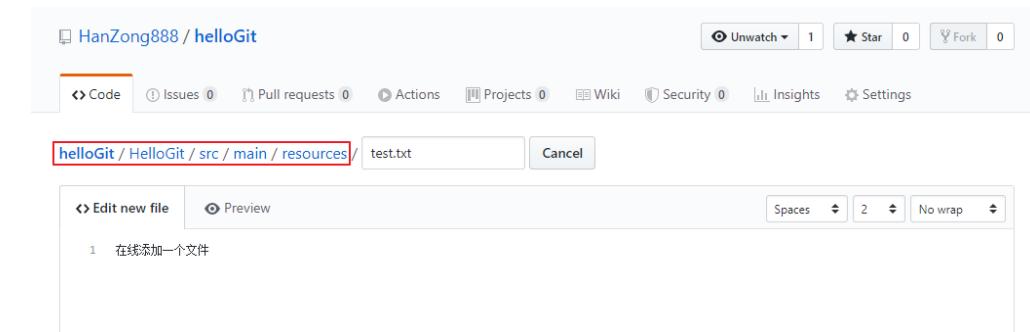
6) 上传成功



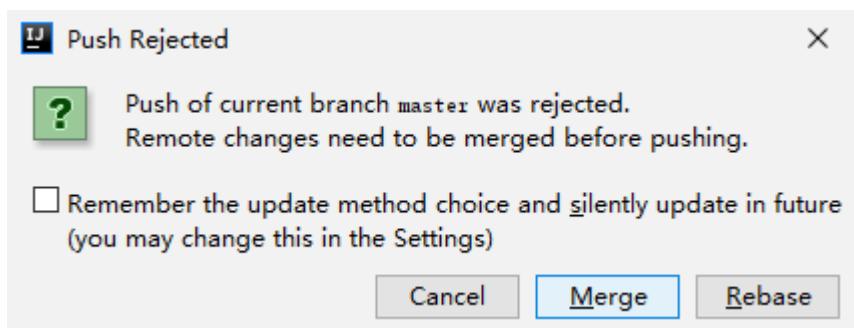
4.4.2 更新本地库

正常情况下是合作伙伴上传新的代码到 GitHub，如果此时本地库不更新将无法上传，为了简单起见，我们直接在 GitHub 上在线修改文件。

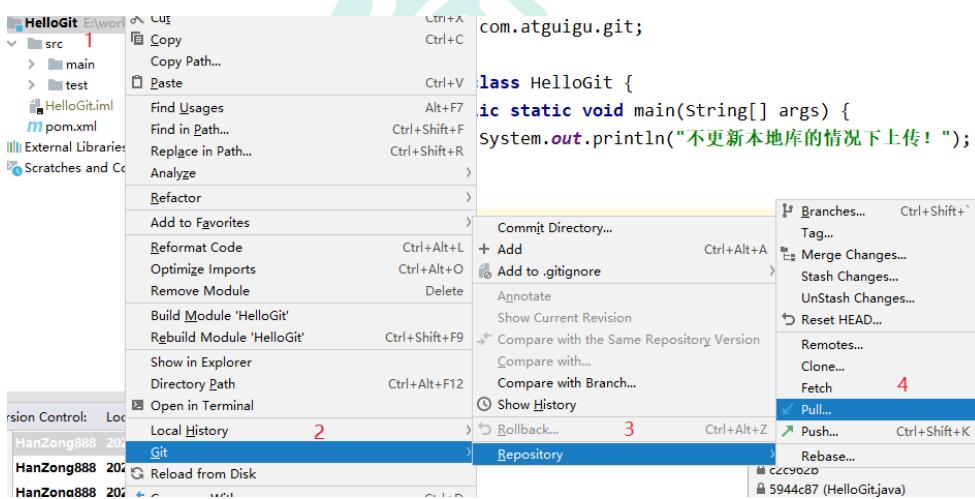
1) 在 GitHub 上在线添加一个文件



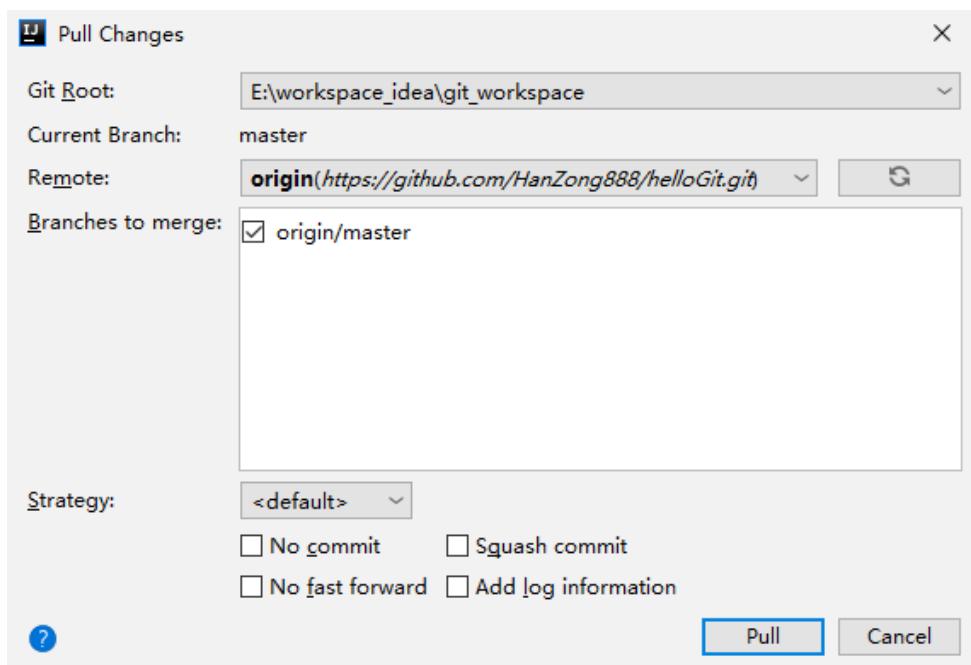
- 2) Idea 中的本地库也修改文件、添加到暂存库、添加到本地库，然后上传，发现上传被拒绝



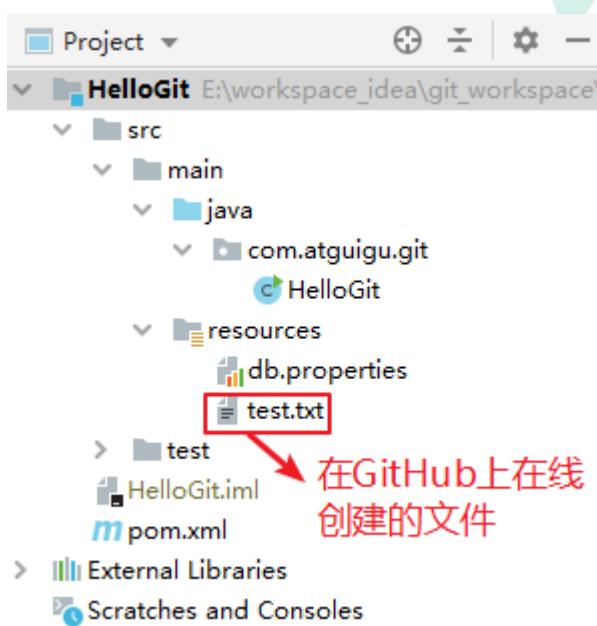
- 3) 此时点击 Merge 或 Rebase 都可以实现本地库与远程 GitHub 的同步
 4) 也可以点击 Cancel 之后通过以下方式更新本地库



- 5) 点击 Pull 将 GitHub 上最新的代码合并都本地库

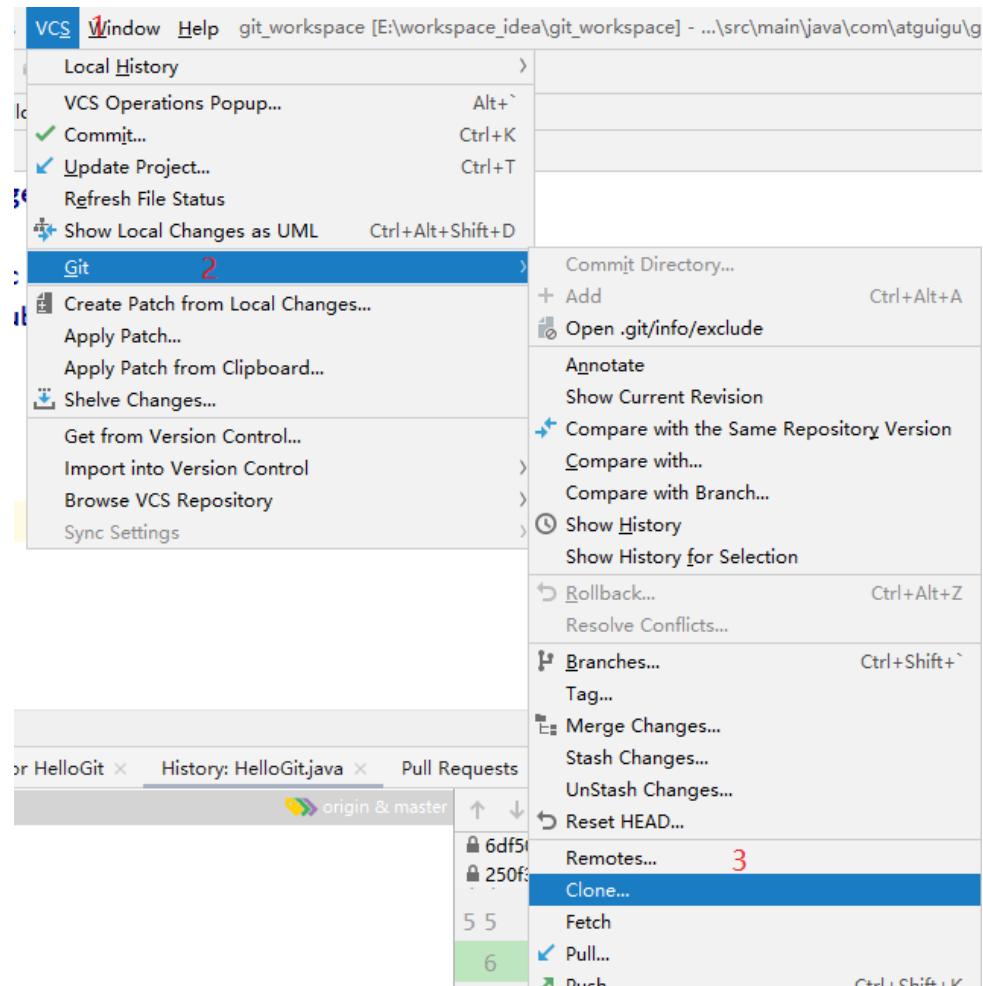


6) 点击 Pull 之后更新本地库成功

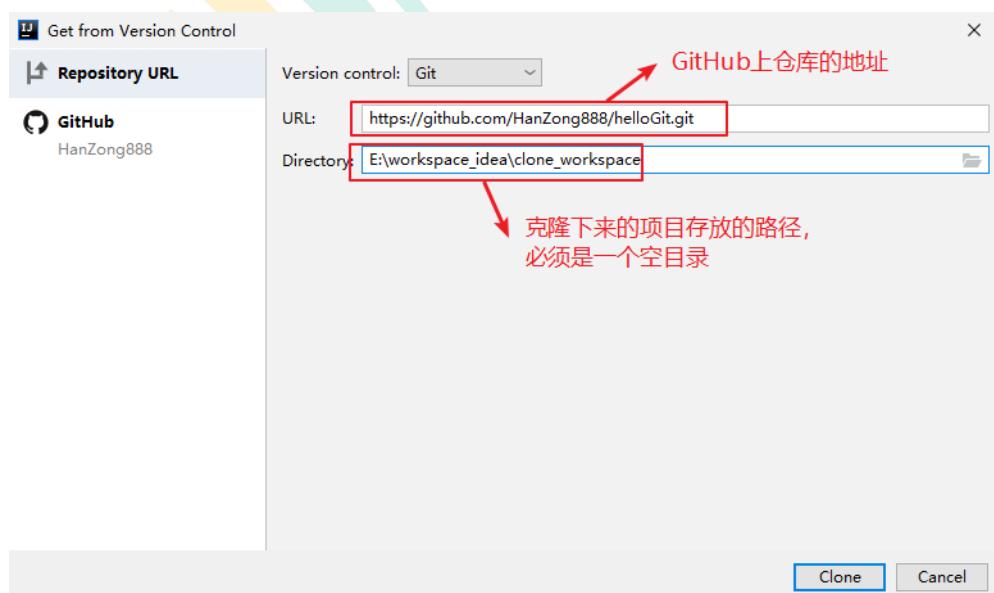


4.5 从 GitHub 上克隆项目到本地

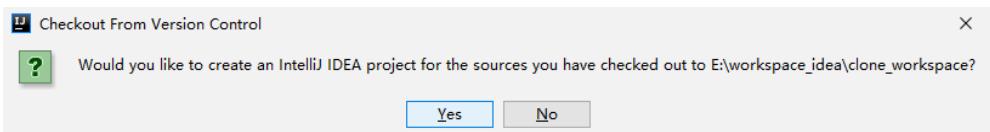
1) 点击 Idea 中的 CVS 选项



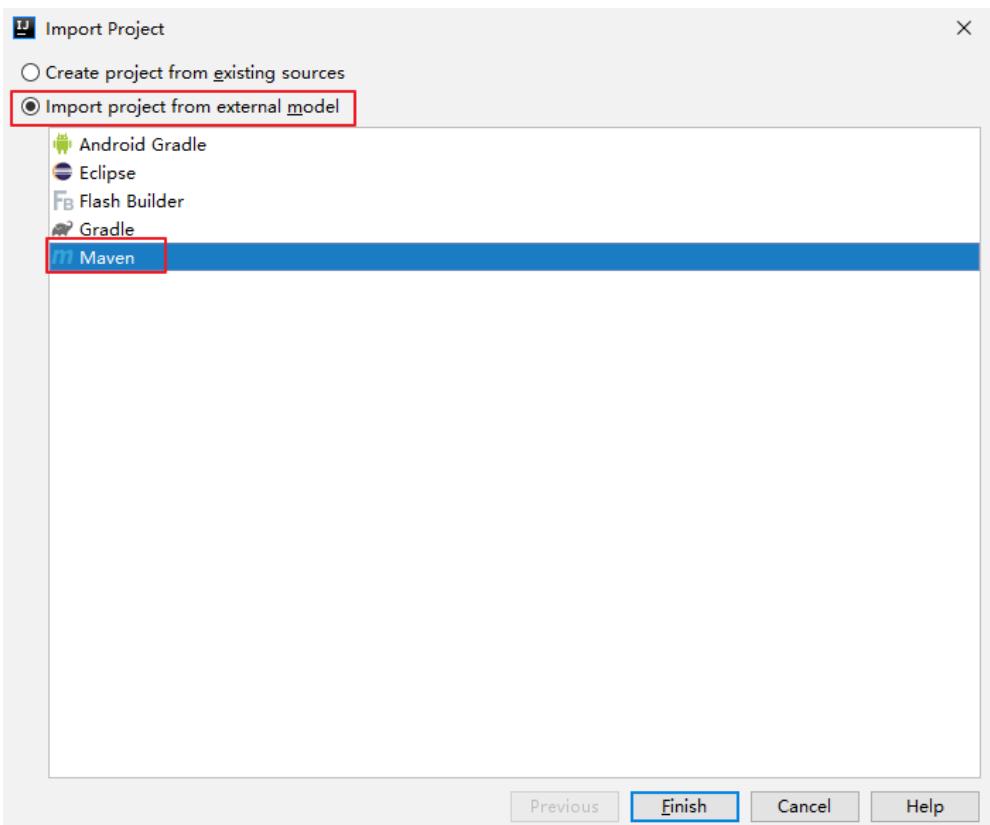
2) 输入 GitHub 中的仓库地址并指定项目的存放路径



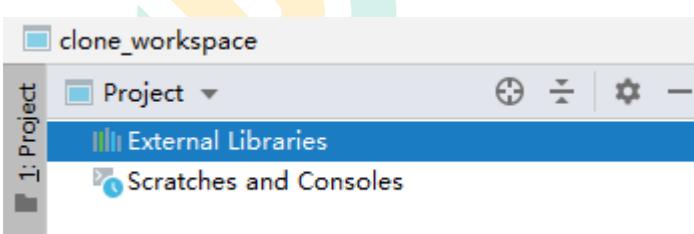
3) 提示是否为克隆的项目创建一个新工程



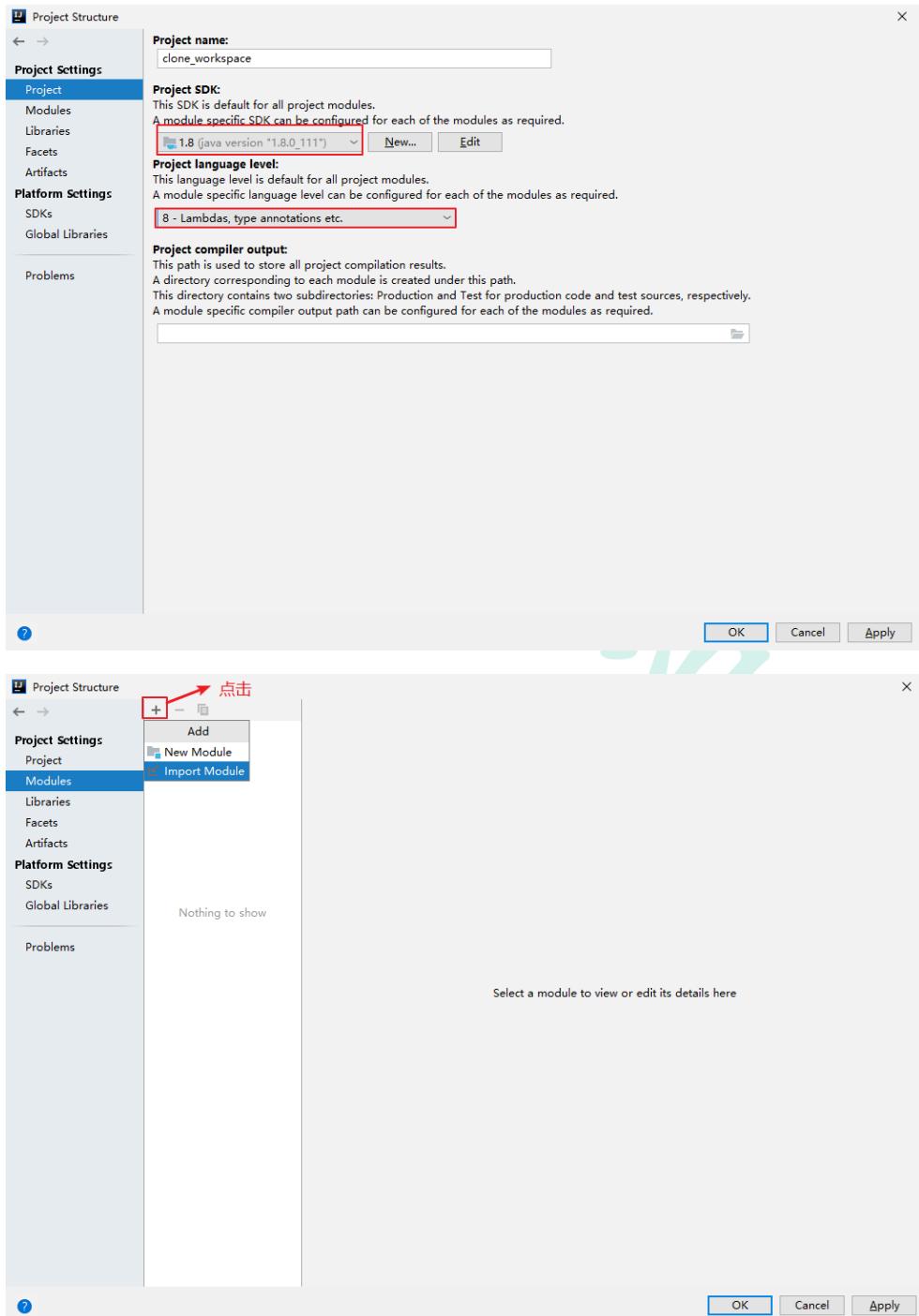
4) 点击 Yes 弹出导入工程的提示框



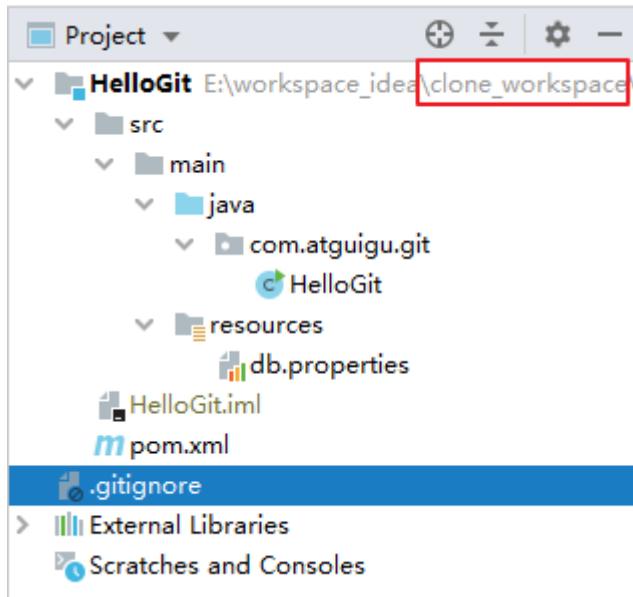
5) 点击 Finish 之后在 Idea 中显示的是一个空工程



6) 需要为新工程配置一下 JDK、导入 Module



7) 克隆成功



第 5 章 国内的项目托管网站-码云

5.1 简介

使用 GitHub 作为项目托管网站如果网速不好很影响效率，大家也可以使用国内的项目托管网站-码云。网址是 <https://gitee.com/>，使用方式跟 GitHub 一样，而且它还是一个中文网站，如果你英文不是很好它是最好的选择。

5.2 配置 SSH 免密登录

在码云上通过 HTTPS 的模式上传项目跟在 GitHub 上一样，但是在码云上上传项目总是输入用户名和密码，比较麻烦，所以给大家演示一下通过 SSH 模式免密登录上传项目，使用 SSH 模式的好处是每次上传项目不需要输入用户名和密码，SSH 免密登录同样适用于 GitHub。

使用 SSH 模式前提是必须是这个项目的拥有者或者合作者，且配好了 SSH Key，配置 SSH Key 的步骤如下：

- 1) 进入电脑的用户目录，在用户目录右键打开 Git 命令行窗口



2) 创建 SSH Key

在命令行窗口输入以下命令

```
ssh-keygen -t rsa -C 任意内容
```

命令参数说明：

-t = The type of the key to generate

密钥的类型

-C = comment to identify the key

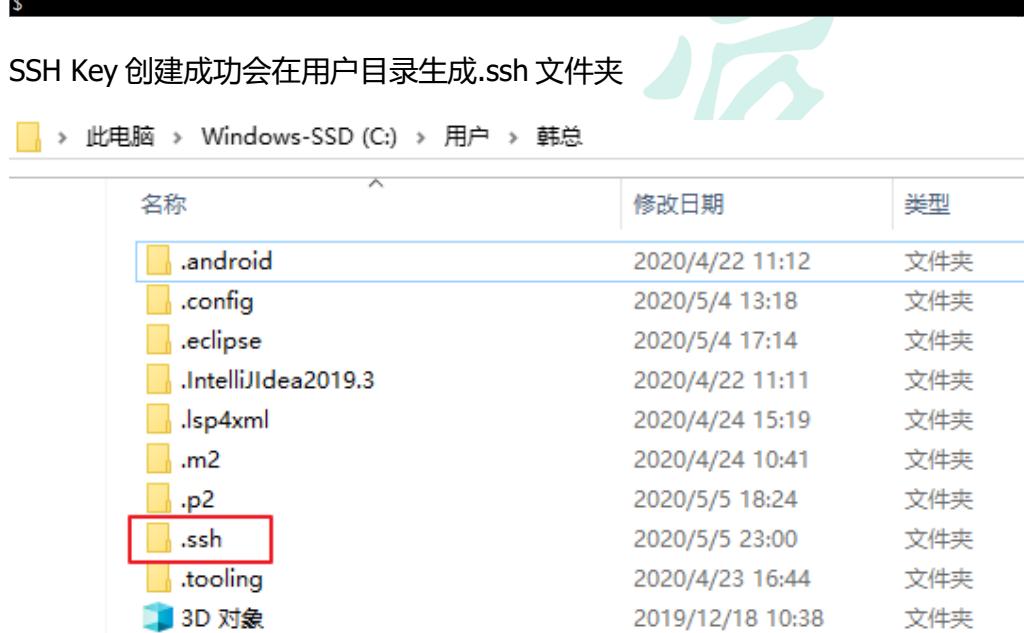
用于识别这个密钥的注释



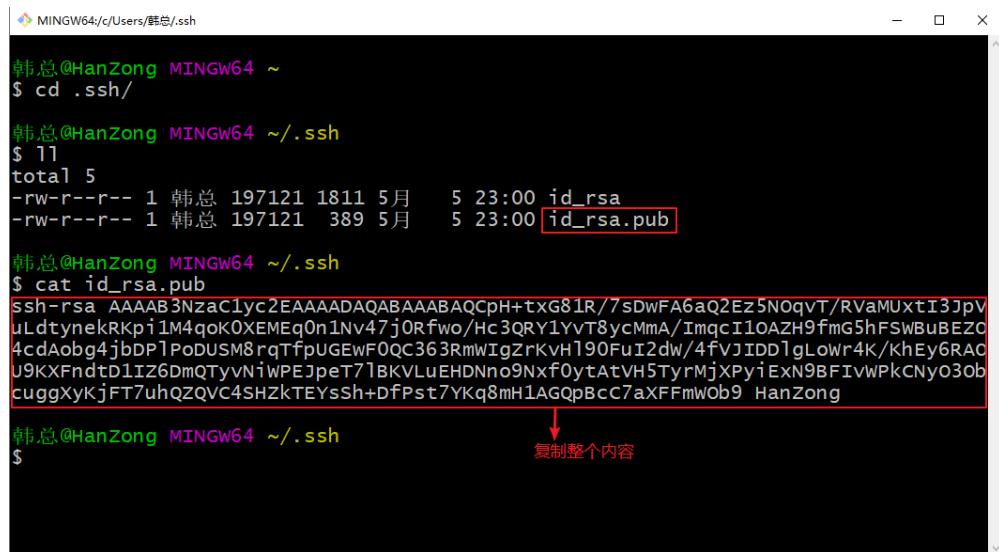
输入命令后回车，然后再回车、回车、回车

```
MINGW64:/c/Users/韩总
韩总@HanZong MINGW64 ~
$ ssh-keygen -t rsa -C HanZong 回车
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/韩总/.ssh/id_rsa): 回车
Created directory '/c/Users/韩总/.ssh'.
Enter passphrase (empty for no passphrase): 回车
Enter same passphrase again: 回车
Your identification has been saved in /c/Users/韩总/.ssh/id_rsa.
Your public key has been saved in /c/Users/韩总/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:nCytwBiw61z9eLdYSXLexgs3zX0UpakiWPbR1SuQnQ HanZong
The key's randomart image is:
+---[RSA 2048]---+
| . . Eo. . = |
| o o oo.+ . + |
| . . .+o.o.oo.. |
| . +. X=+=. . . |
| . .oo+oS=++. |
| . .oo++o. |
| o . . . |
| . |
+---[SHA256]---+
韩总@HanZong MINGW64 ~
$
```

3) SSH Key 创建成功会在用户目录生成.ssh 文件夹



4) 进入.ssh 文件夹，查看 id_rsa.pub 文件，复制全部内容



```

MINGW64:/c/Users/韩总/.ssh
韩总@HanZong MINGW64 ~
$ cd .ssh/
韩总@HanZong MINGW64 ~/.ssh
$ ll
total 5
-rw-r--r-- 1 韩总 197121 1811 5月 5 23:00 id_rsa
-rw-r--r-- 1 韩总 197121 389 5月 5 23:00 id_rsa.pub

韩总@HanZong MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCPH+txG81R/7sDwFA6aQ2EZ5NOqvT/RVaMUxtI3JpV
uLdtynekRKpi1M4qoK0XEMEq0n1Nv47j0Rfwo/Hc3QRY1YvT8ycMmA/ImqcI1oAZH9fmG5hFSWBBeZO
4cdAobg4jbDPtPoDUSM8rqTfpUGEwF0QC363RmWlgZrKvH190FuI2dw/4fVJIDDIgLoWr4K/khEy6RAO
u9KXFndtD1IZ6DmQTyvNiwPEJpeT7lBKVLuEHDNno9Nxf0ytAtVH5TyrmjXPyiExN9BFivwPkCNy03ob
cuggXYkjFT7uhQZQVC4SHzktEYssh+DfPst7YKq8mH1AGQpBcc7aXFmWOb9 HanZong

```

5) 找到码云账户的设置



6) 点击 SSH 公钥，设置标题，粘贴公钥，点击确定



SSH公钥

使用SSH公钥可以让你在你的电脑和码云通讯的时候使用安全连接（Git的Remote要使用SSH地址）

您当前的SSH公钥数: 0
你还没有添加任何SSH公钥

添加公钥

标题 **任意命名**

公钥 **粘贴公钥**

ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCPH+txG81R/7sDwFA6aQ2EZ5NOqvT/RVaMUxtI3JpV
uLdtynekRKpi1M4qoK0XEMEq0n1Nv47j0Rfwo/Hc3QRY1YvT8ycMmA/ImqcI1oAZH9fmG5hFSWBBeZO
4cdAobg4jbDPtPoDUSM8rqTfpUGEwF0QC363RmWlgZrKvH190FuI2dw/4fVJIDDIgLoWr4K/khEy6RAO
u9KXFndtD1IZ6DmQTyvNiwPEJpeT7lBKVLuEHDNno9Nxf0ytAtVH5TyrmjXPyiExN9BFivwPkCNy03ob
cuggXYkjFT7uhQZQVC4SHzktEYssh+DfPst7YKq8mH1AGQpBcc7aXFmWOb9 HanZong

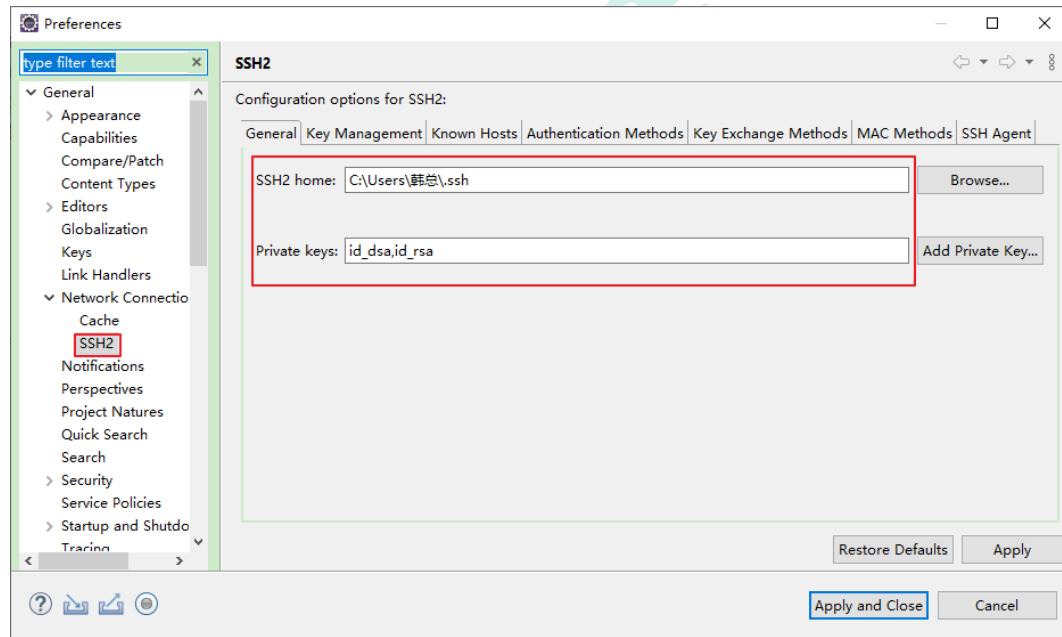
Tips: 码云账户中可以添加多个 SSH 公钥，但是一台电脑只能授权一个用户免密登录

7) 复制码云账户中仓库的 SSH 地址



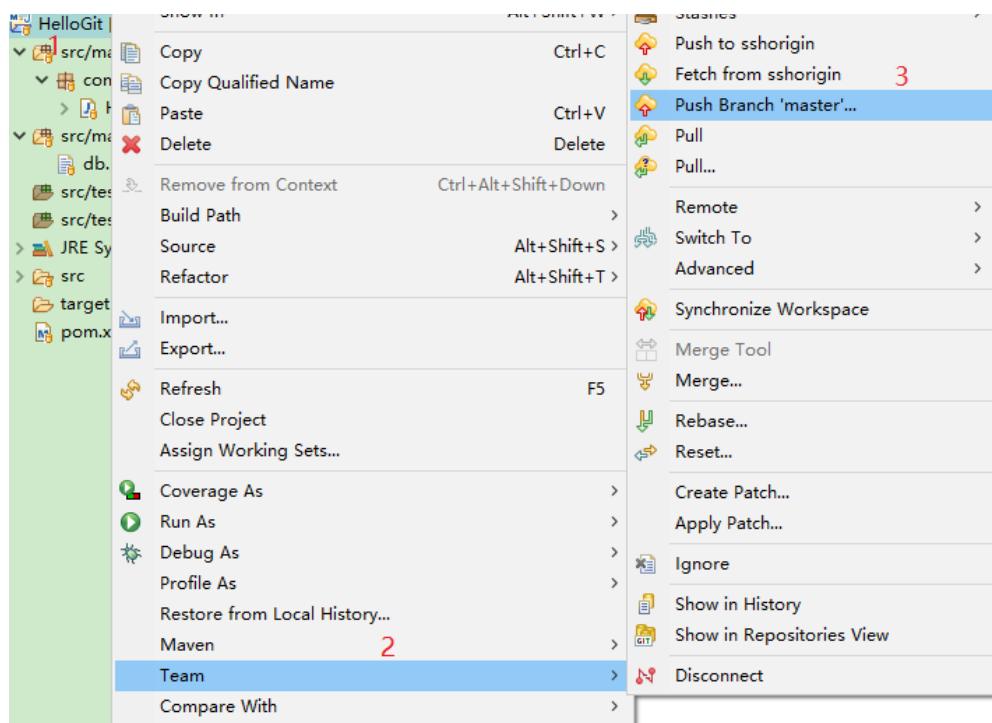
5.3 在 Eclipse 中通过 SSH 模式上传项目

创建了 SSH Key 之后 Eclipse 可以自动识别对应的公钥和私钥文件

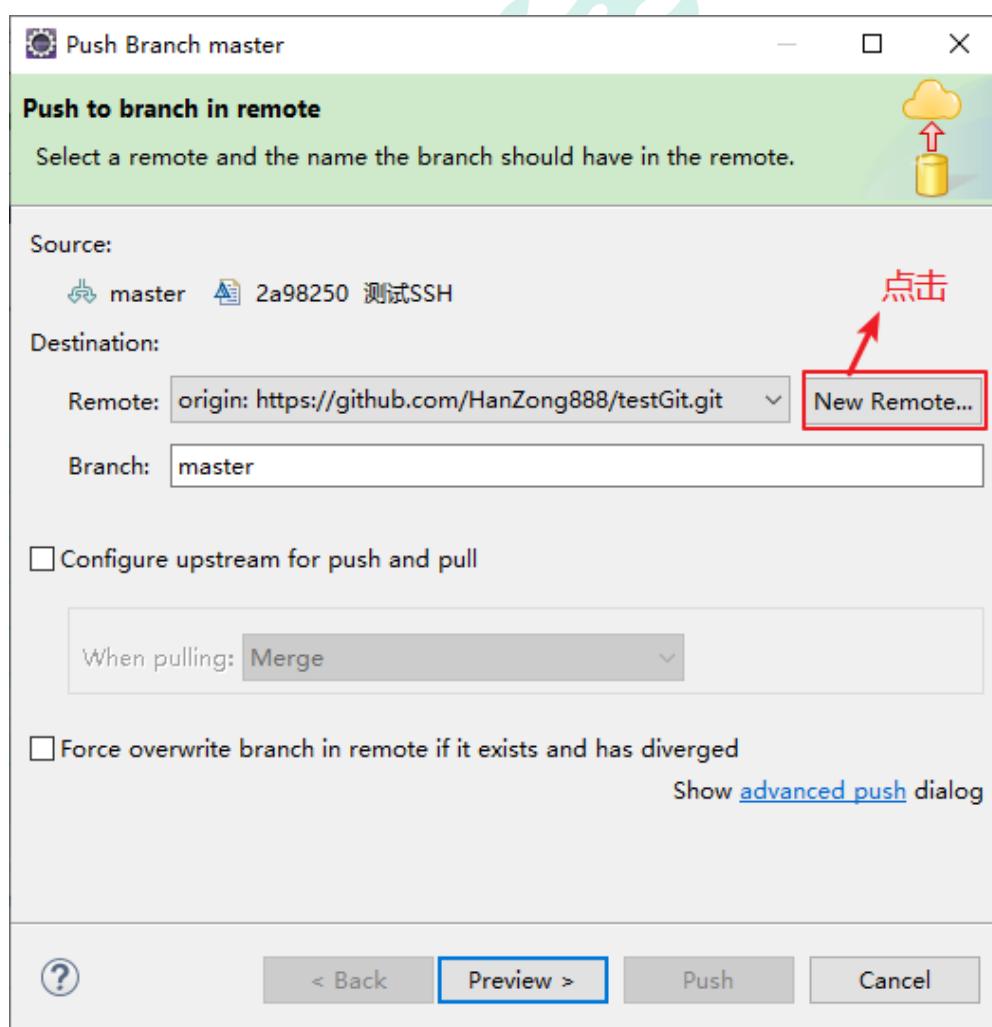


通过 SSH 模式上传项目的步骤：

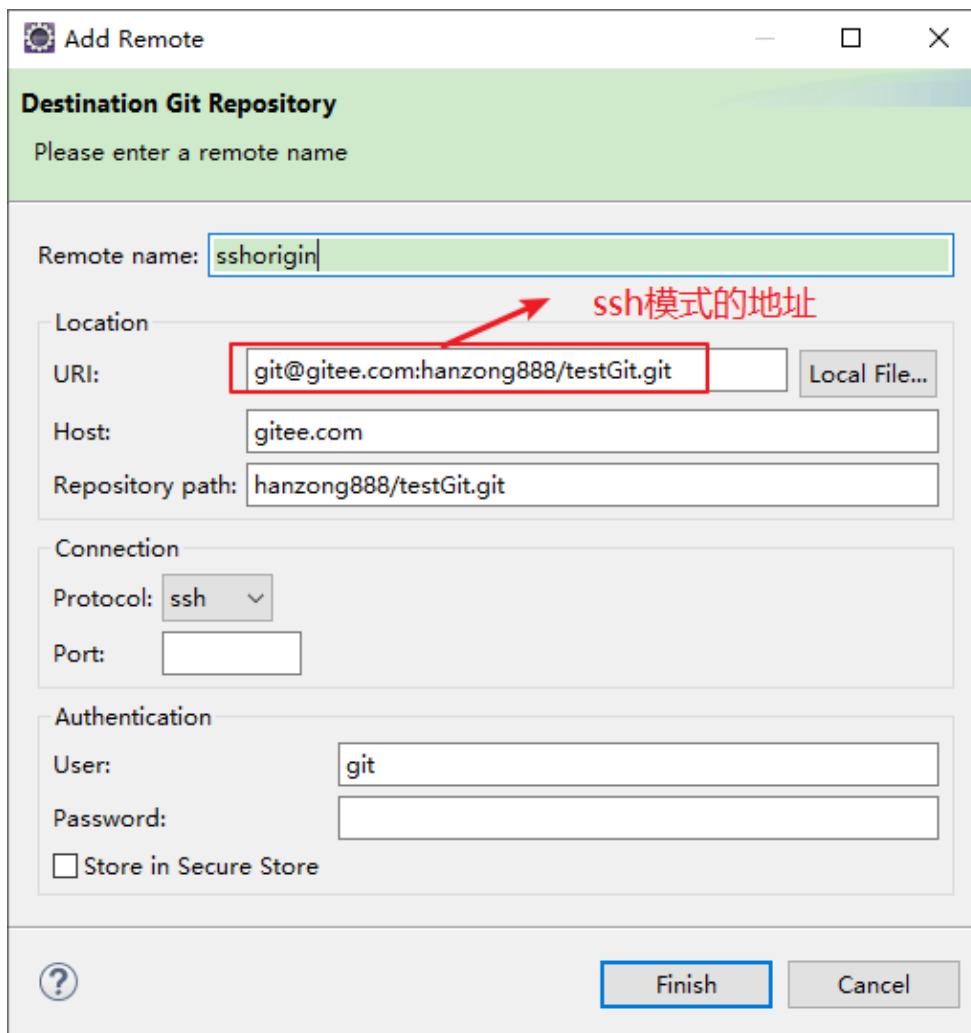
- 1) 在要上传的项目上右键



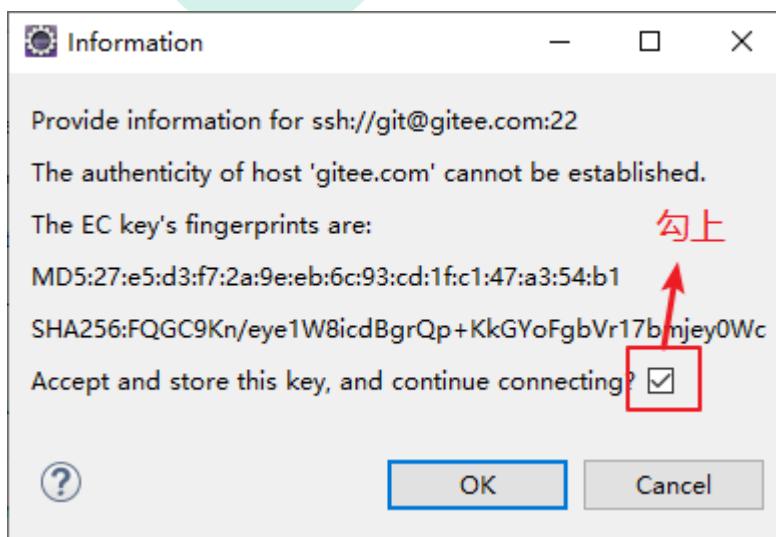
2) 点击 New Remote...

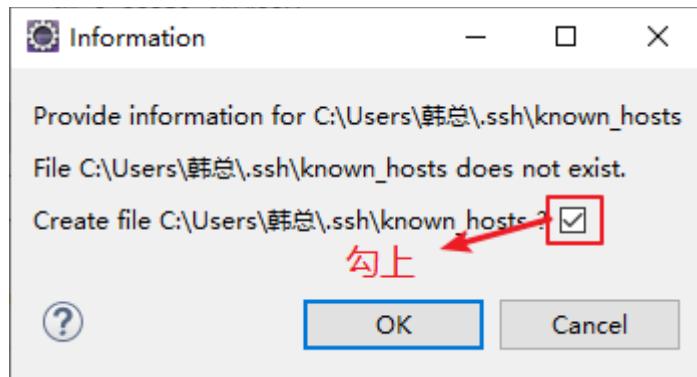


3) 复制 SSH 模式的地址并给远程地址起一个别名

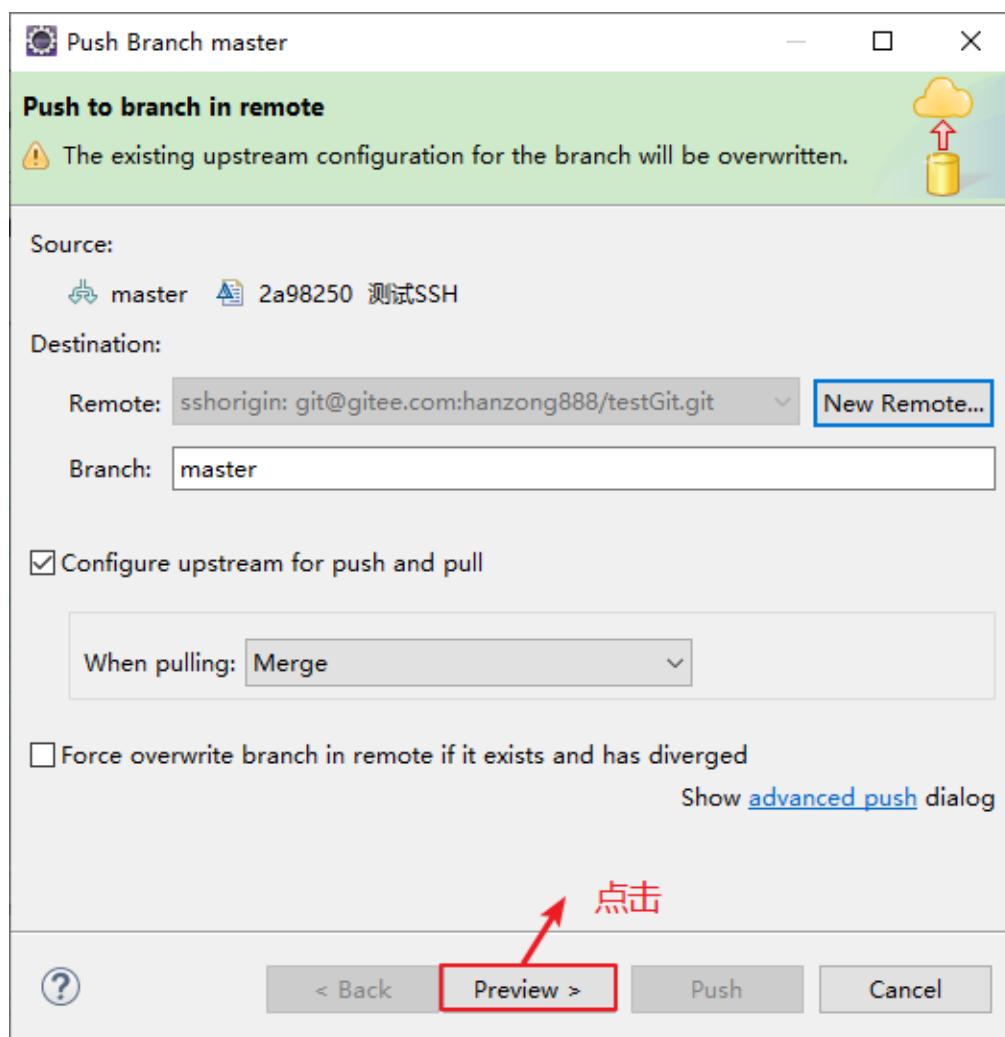


4) 因为是第一次使用 SSH 模式，点击 Finish 之后需要勾选保存 key，勾选创建 known_hosts 文件，以后就不需要这样了

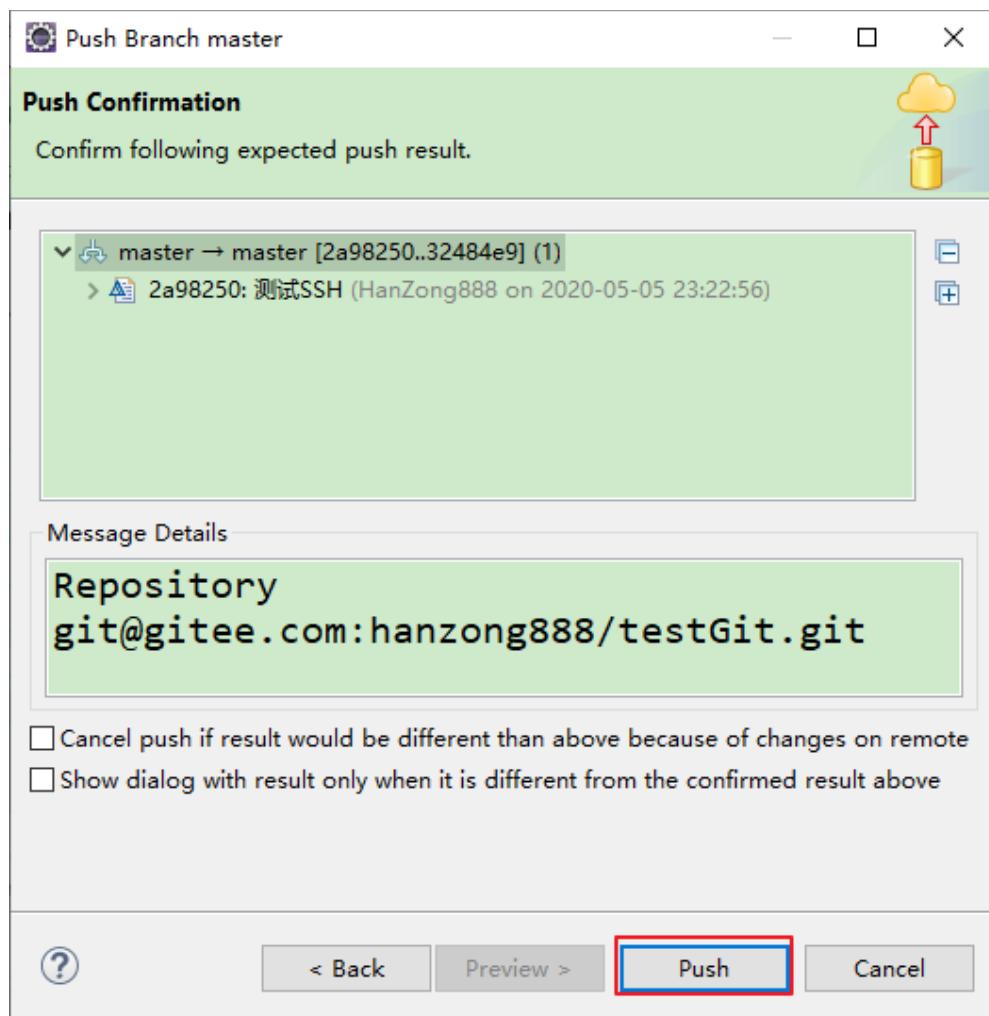




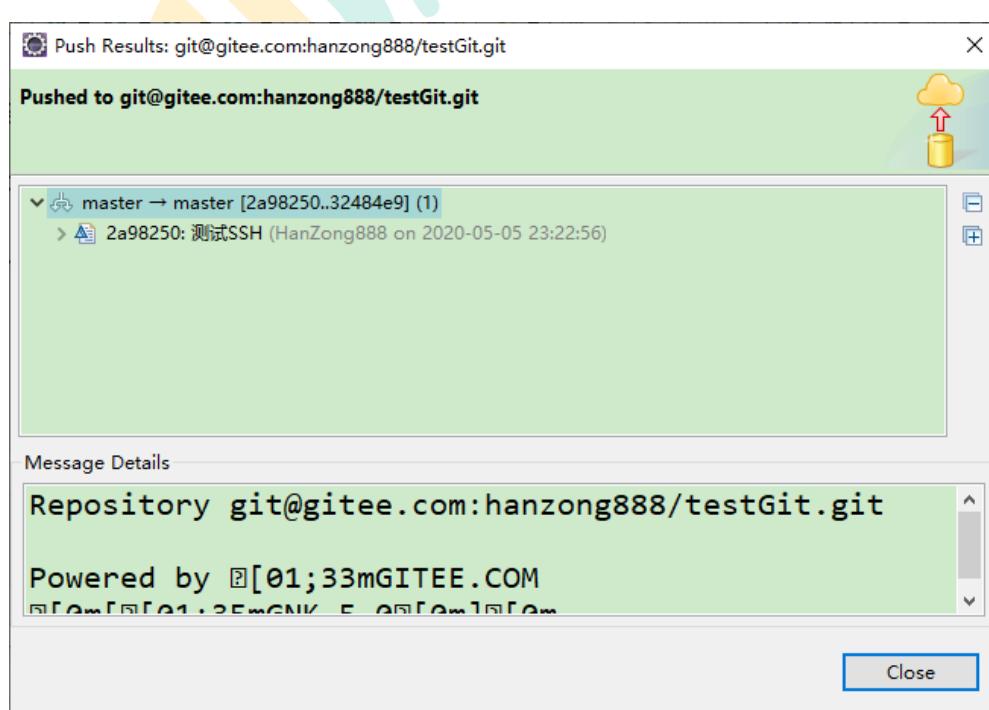
5) 点击 Preview 进入确定上传窗口



6) 点击 Push 开始上传，不再需要输入用户名和密码



7) 上传成功

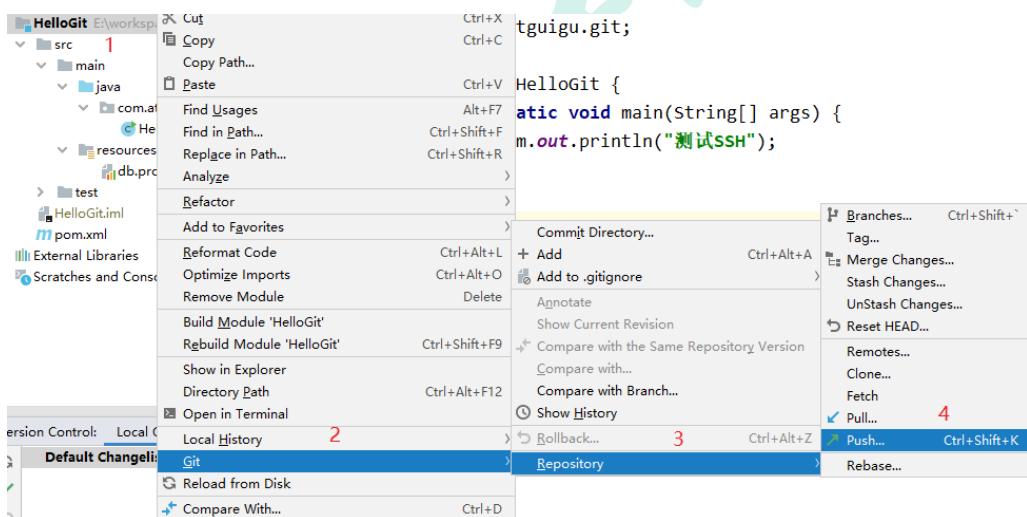


5.4 在 Idea 中通过 SSH 模式上传项目

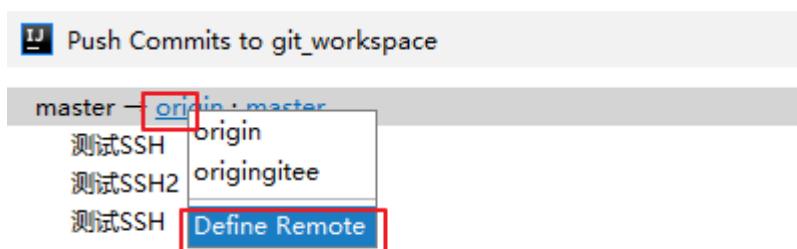
- 在码云账户中创建一个新的仓库，复制 SSH 模式的地址



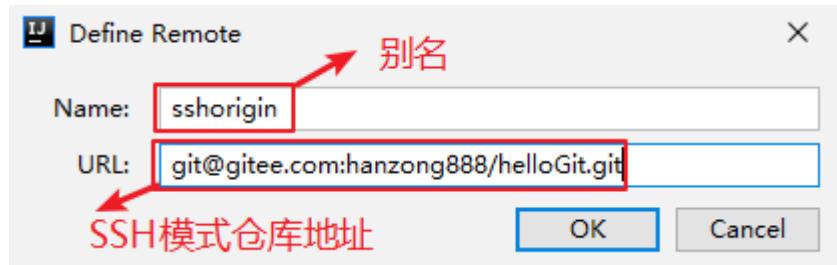
- 在 Idea 中要上传的项目上右键



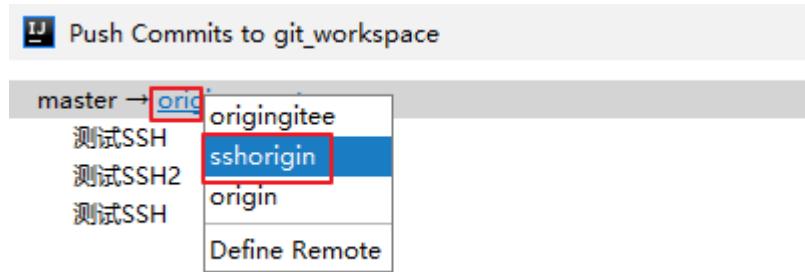
- 在弹出的窗口点击 origin→Define Remote



- 设置 SSH 模式远程地址与别名



5) 选择定义的 sshorigin 开始上传项目



6) 点击 Push 直接上传成功

