

Maven 聚合项目搭建

作者: [PanCode](#)

原文链接: <https://ld246.com/article/1591345448631>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

Maven聚合项目搭建

Maven聚合项目



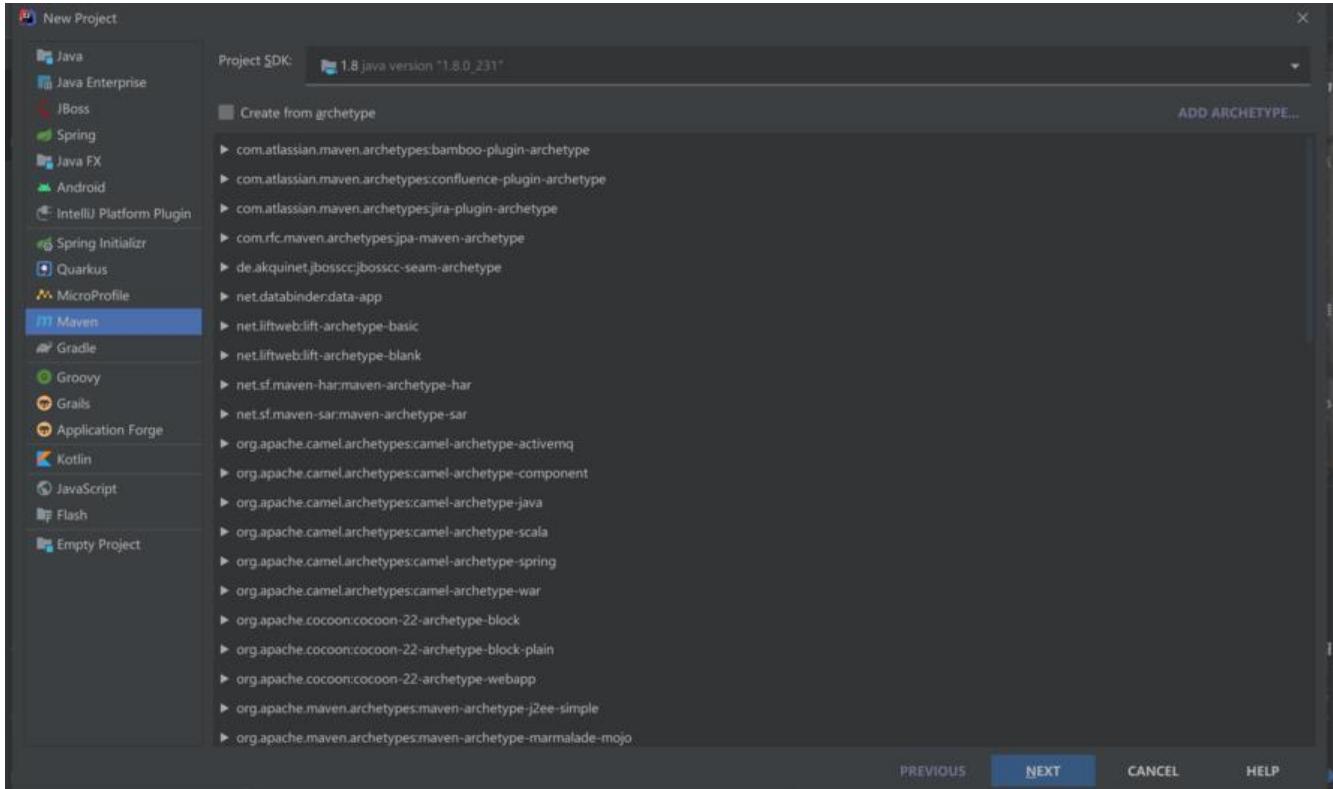
项目准备：

环境：jdk1.8 && maven:latest

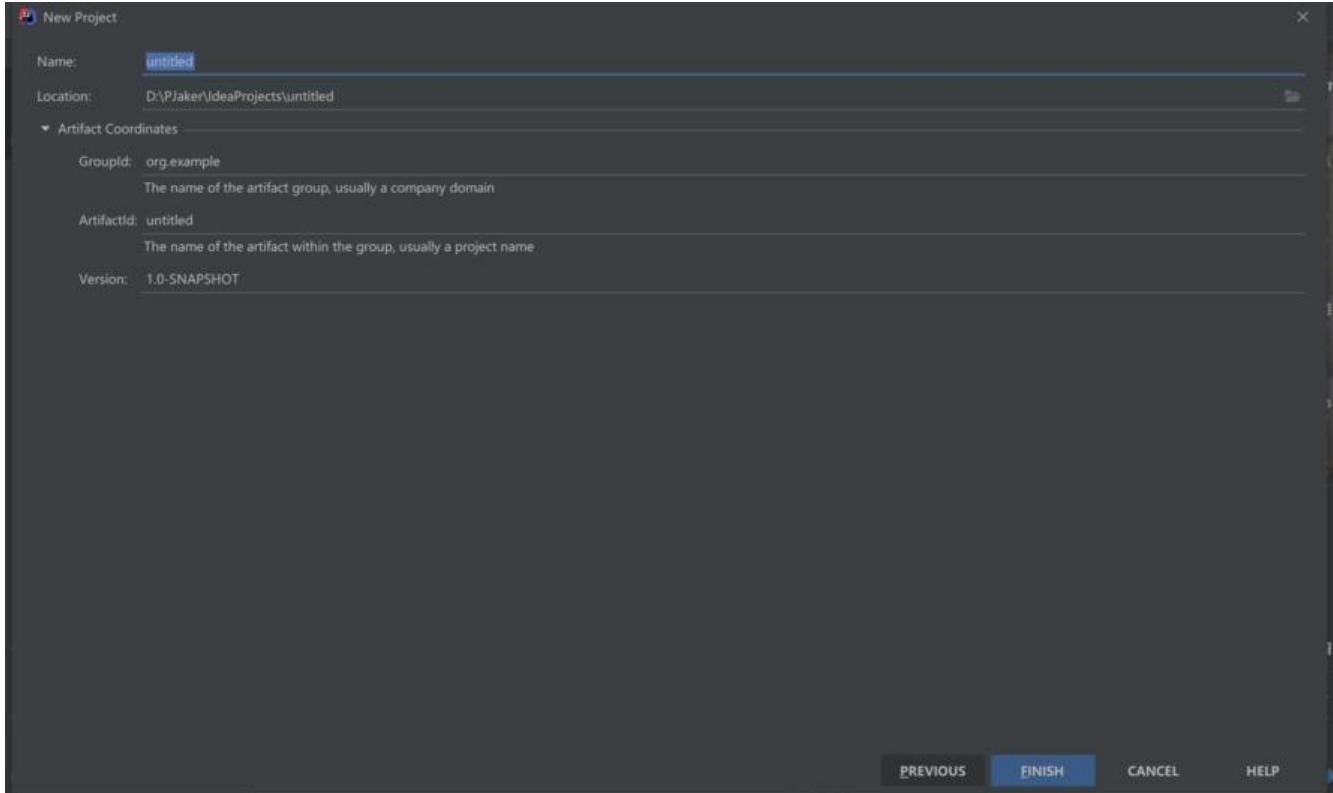
编辑器：# IntelliJ IDEA

具体操作

File->New->Project->Maven



NEXT



写好项目名称和包名等基本设置 -> FINISH

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows a tree view of the project structure under "pan-study". It includes a ".idea" folder, a "src" folder with "main" and "test" subfolders, and "External Libraries".
- pom.xml (pan-study) Content:** The main editor window displays the following XML code:

```
<?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.pancode</groupId>
    <artifactId>pan-study</artifactId>
    <version>1.0-SNAPSHOT</version>
</project>
```
- Toolbars and Status Bar:** The top bar has standard menu items like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, Window, Help, Statistics. The status bar at the bottom shows "39 chars, 1 line break, 2-1 LF, UTF-8, 4 spaces".

此时pan-study是聚合工程最外层也就是主工程。所以打包方式得是pom来告诉maven，这是一个父工程。

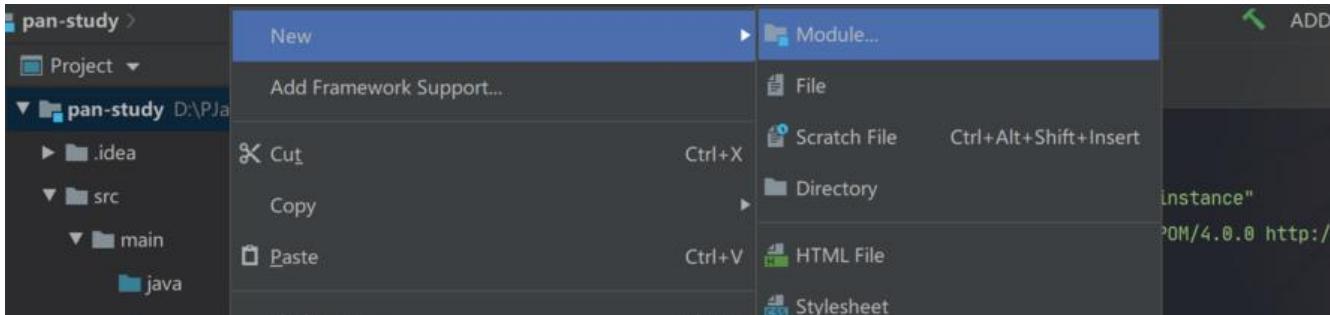
一般来说所有的父级项目的packaging都为pom，packaging默认类型jar类型，如果不做配置，maven会将该项目打成jar包。

The screenshot shows the IntelliJ IDEA interface with the following details:

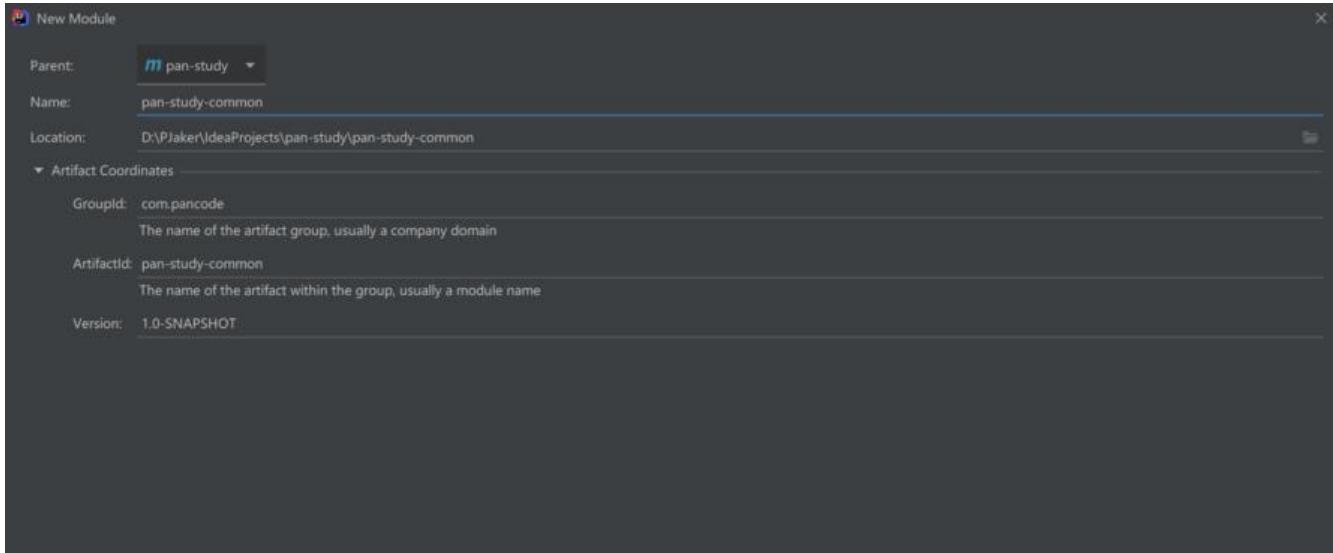
- pom.xml (pan-study) Content:** The main editor window displays the following XML code, with the `<packaging>pom</packaging>` line highlighted:

```
<?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.pancode</groupId>
    <artifactId>pan-study</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>
</project>
```

点击选中项目名->New->Module



直接Next，然后配置好子工程名称



FINISH

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure with a parent module 'pan-study' containing a child module 'pan-study-common'. Both modules have their own 'pom.xml' files.
- pom.xml (pan-study):** Contains the definition for the parent module, including the groupId 'com.panicode', artifactId 'pan-study', version '1.0-SNAPSHOT', and a child module reference for 'pan-study-common'.
- pom.xml (pan-study-common):** Contains the definition for the child module, including the groupId 'com.panicode', artifactId 'pan-study-common', version '1.0-SNAPSHOT', and a packaging type of 'pom'.

项目模块结果和pom文件对比

父子工程依赖关系

1. 聚合工程里可以分为顶级项目（顶级工程、父工程）与子工程，这两者的关系其实就是父子继承的系

子工程在maven里称之为模块（module），模块之间是平级，是可以相互依赖的。

2. 子模块可以使用顶级工程里所有的资源（依赖），子模块之间如果要使用资源，必须构建依赖（构关系）

3. 一个顶级工程是可以由多个不同的子工程共同组合而成。

接下来同理且相同操作构建api, pojo, mapper, service四个子工程。

```
<?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.pancode</groupId>
    <artifactId>pan-study</artifactId>
    <version>1.0-SNAPSHOT</version>
    <modules>
        <module>pan-study-common</module>
        <module>pan-study-api</module>
        <module>pan-study-pojo</module>
        <module>pan-study-mapper</module>
        <module>pan-study-service</module>
        <module>pan-study-poj</module>
    </modules>
    <packaging>pom</packaging>
</project>
```

构建子模块依赖关系

例如pojo依赖common

```
<?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">
    <parent>
        <artifactId>pan-study</artifactId>
        <groupId>com.pancode</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>pan-study-pojo</artifactId>
    <dependencies>
        <dependency>
            <groupId>com.pancode</groupId>
            <artifactId>pan-study-common</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
</project>
```

当pojo依赖了common，此时mapper再依赖pojo，那么mapper就相当于同时依赖pojo和commo

The screenshot shows two tabs in a code editor: 'pom.xml (pan-study-pojo)' and 'pom.xml (pan-study-mapper)'. The 'pan-study-mapper' tab is active, displaying the following Maven POM configuration:

```
7     <groupId>com.pancode</groupId>
8     <version>1.0-SNAPSHOT</version>
9   </parent>
10    <modelVersion>4.0.0</modelVersion>
11
12    <artifactId>pan-study-mapper</artifactId>
13
14    <!--
15        mapper -> pojo -> common
16        mapper通过pojo是可以使用common中相应的方法的
17        -->
18    <dependencies>
19      <dependency>
20        <groupId>com.pancode</groupId>
21        <artifactId>pan-study-pojo</artifactId>
22        <version>1.0-SNAPSHOT</version>
23      </dependency>
24    </dependencies>
25  </project>
```

A yellow circular icon is positioned next to the artifactId 'pan-study-pojo' in the dependency section.

最终达到这样的依赖结构

api -> service -> mapper -> pojo -> common

IDEA 工具栏显示了两个 pom.xml 文件：pan-study-pojo 和 pan-study-api。

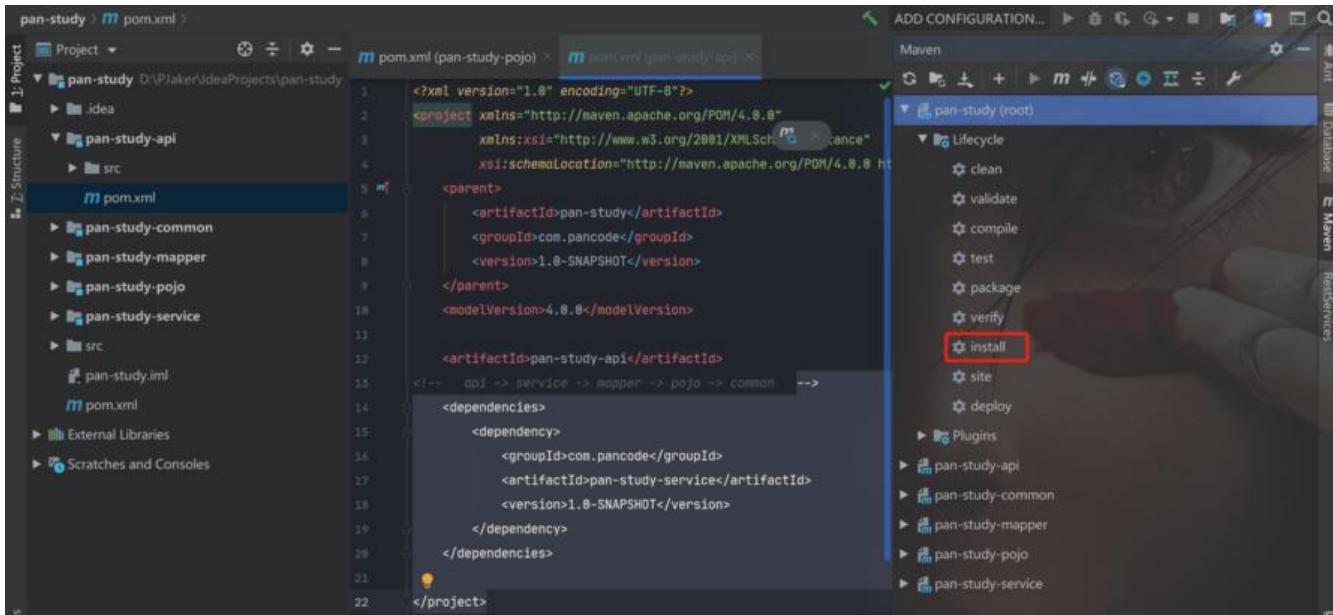
```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org
5   m<parent>
6     <artifactId>pan-study</artifactId>
7     <groupId>com.pancode</groupId>
8     <version>1.0-SNAPSHOT</version>
9   </parent>
10  <modelVersion>4.0.0</modelVersion>
11
12  <artifactId>pan-study-api</artifactId>
13  <!-- api -> service -> mapper -> pojo -> common -->
14  <dependencies>
15    <dependency>
16      <groupId>com.pancode</groupId>
17      <artifactId>pan-study-service</artifactId>
18      <version>1.0-SNAPSHOT</version>
19    </dependency>
20  </dependencies>
21
22 </project>

```

最后安装聚合工程（没安装前，子模块不能配合父模块使用）

IDEA 右侧选择 Maven->父工程->Lifecycle->install



提示 BUILD SUCCESS 即安装成功

```
[INFO] Reactor Summary for pan-study 1.0-SNAPSHOT:  
[INFO]  
[INFO] pan-study ..... SUCCESS [ 0.227 s]  
[INFO] pan-study-common ..... SUCCESS [ 0.810 s]  
[INFO] pan-study-pojo ..... SUCCESS [ 0.055 s]  
[INFO] pan-study-mapper ..... SUCCESS [ 0.051 s]  
[INFO] pan-study-service ..... SUCCESS [ 0.050 s]  
[INFO] pan-study-api ..... SUCCESS [ 0.050 s]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 1.329 s  
[INFO] Finished at: 2020-06-05T15:32:43+08:00  
[INFO] -----
```

聚合工程整合springboot

父工程pom文件里添加对应依赖

1.引入依赖 parent

```
<parent>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-parent</artifactId>  
  <version>2.1.5.RELEASE</version>  
  <relativePath />  
</parent>
```

2.设置资源属性

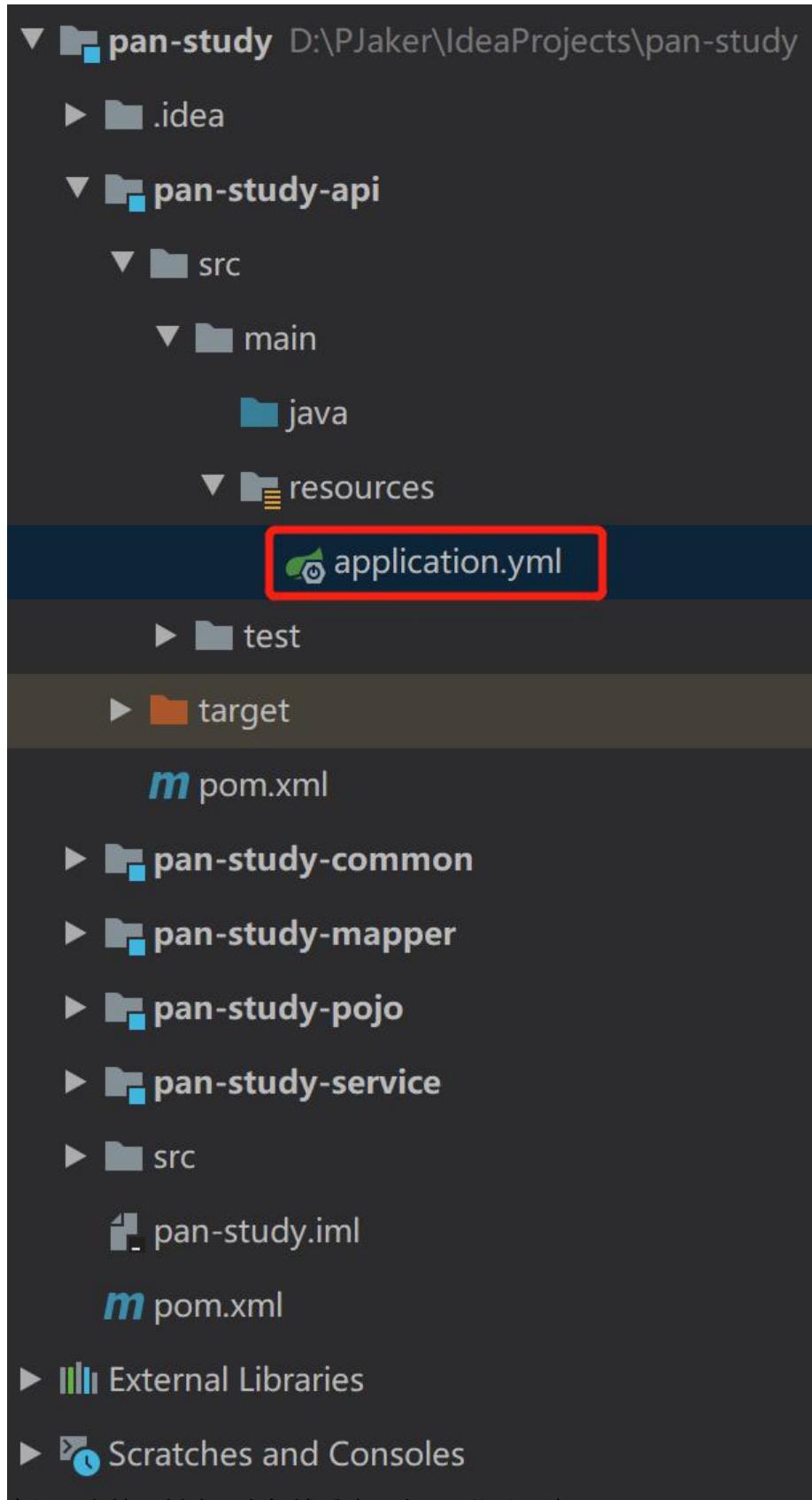
```
<properties>  
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
  <java.version>1.8</java.version>  
</properties>
```

3.引入依赖 dependency

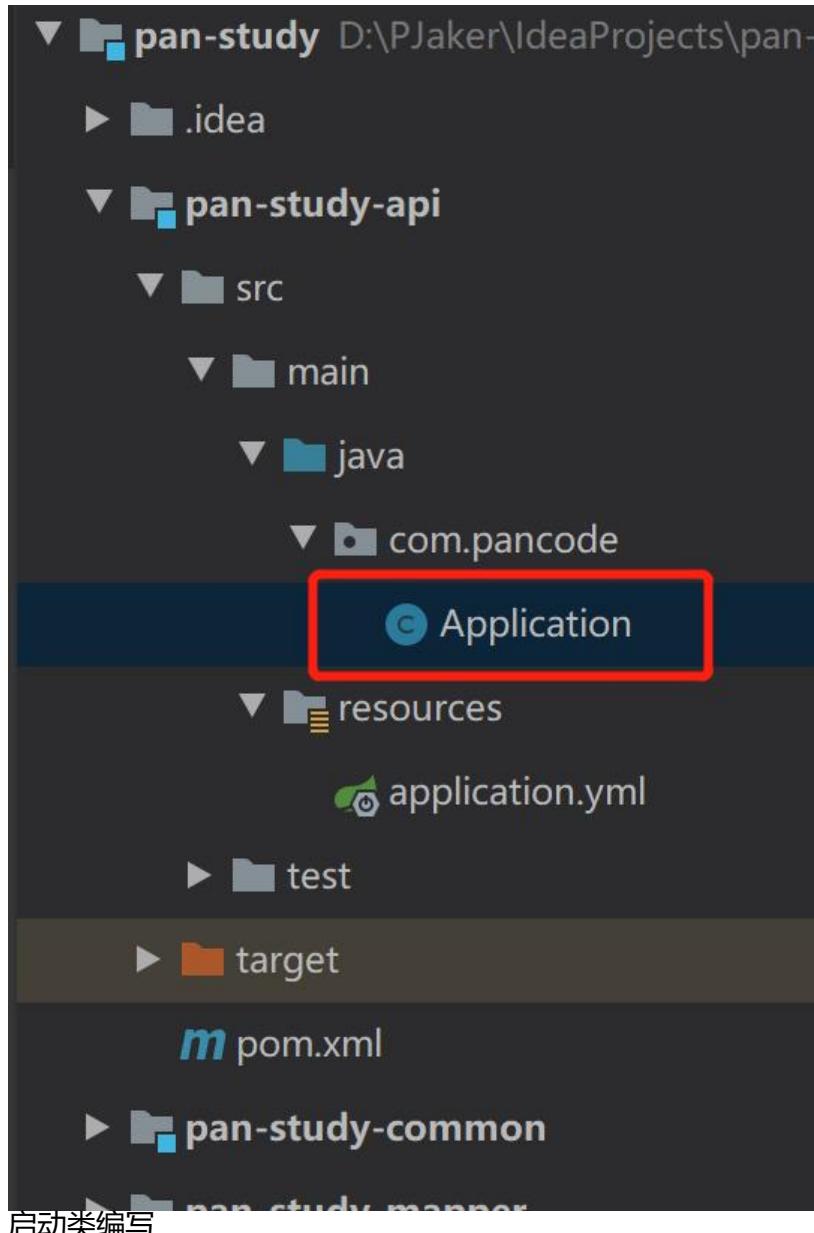
```
<dependencies>  
  <dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter</artifactId>  
    <exclusions>  
      <exclusion>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-logging</artifactId>  
      </exclusion>  
    </exclusions>  
  </dependency>  
  <dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-web</artifactId>
```

```
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-configuration-processor</artifactId>
    <optional>true</optional>
</dependency>
</dependencies>
```

在api子工程里创建yml文件



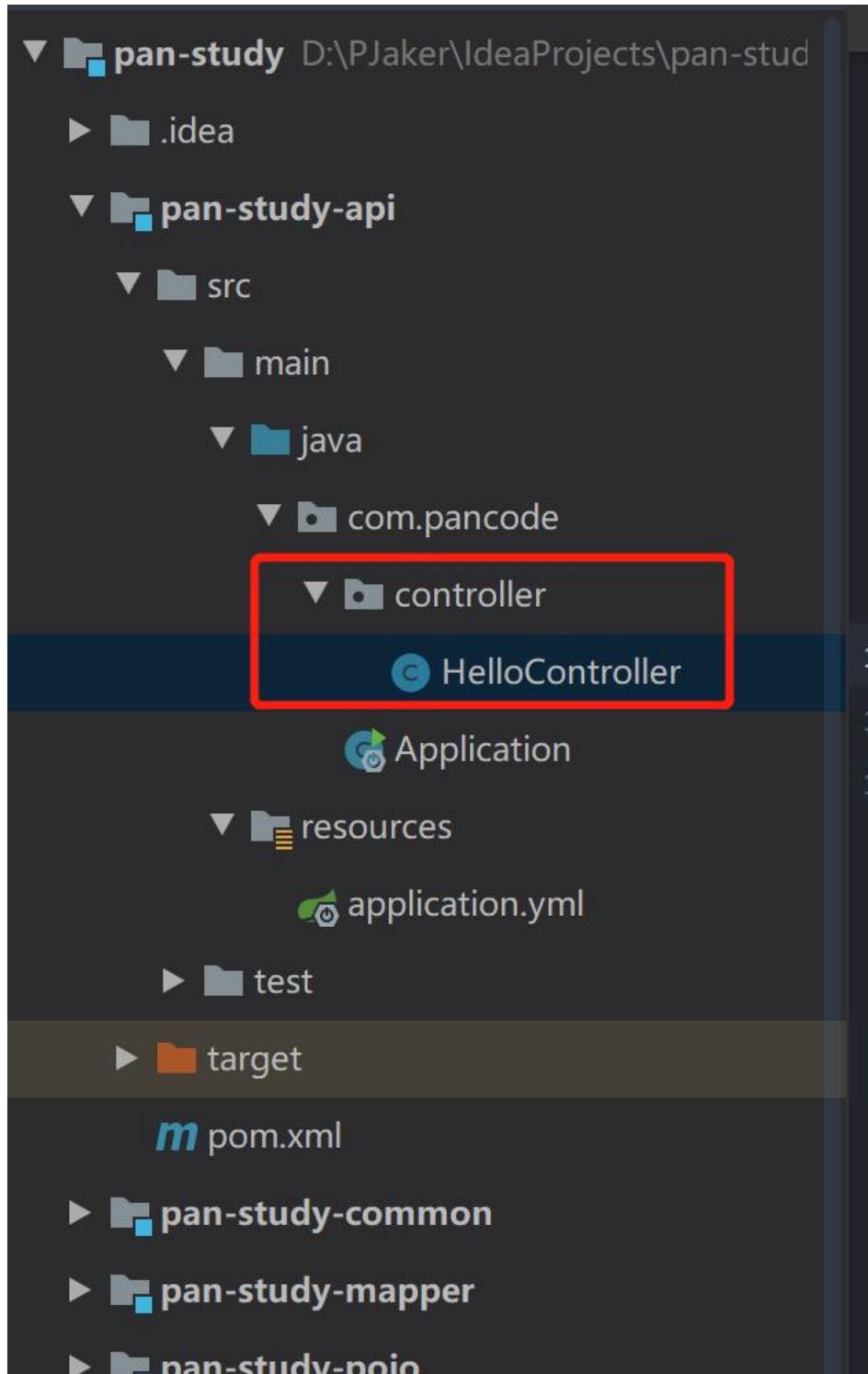
在java文件下创建一个包然后建一个Applictian类



启动类编写

```
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class,args);
    }
}
```

创建controller层的HelloController做测试



@RestController//默认返回出去的是json对象
public class HelloController {

 @GetMapping("/hello")
 public Object hello() {
 return "Hello World! ";
 }

}

最后去maven里install一下，再在启动类里启动

```
1 package com.pancode;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 /**
7 * @PACKAGE_NAME: com.pancode
8 * @Author: PJaker
9 * @DATE: 2020/6/5
10 * @TIME: 15:51
11 * @Description:
12 */
13
14 @SpringBootApplication
15 public class Application {
16     public static void main(String[] args) {
17         SpringApplication.run(Application.class, args);
18     }
19 }
```

在控制台可以看到启动成功后项目的端口（默认为8080）

```
Run: Application
G Console Endpoints
信息: Starting service [Tomcat]
六月 05, 2020 4:00:26 下午 org.apache.catalina.core.StandardEngine startInternal
信息: Starting Servlet engine: [Apache Tomcat/9.0.19]
六月 05, 2020 4:00:26 下午 org.apache.catalina.core.ApplicationContext log
信息: Initializing Spring embedded WebApplicationContext
六月 05, 2020 4:00:26 下午 org.springframework.boot.web.servlet.context.ServletWebServerApplicationContext prepareWebApplicationContext
信息: Root WebApplicationContext: initialization completed in 898 ms
六月 05, 2020 4:00:26 下午 org.springframework.scheduling.concurrent.ExecutorConfigurationSupport initialize
信息: Initializing ExecutorService 'applicationTaskExecutor'
六月 05, 2020 4:00:26 下午 org.springframework.boot.web.embedded.tomcat.TomcatWebServer start
信息: Tomcat started on port(s): 8080 (http) with context path ''
六月 05, 2020 4:00:26 下午 org.springframework.boot.StartupInfoLogger logStarted
信息: Started Application in 1.493 seconds (JVM running for 2.809)
六月 05, 2020 4:00:36 下午 org.apache.catalina.core.ApplicationContext log
信息: Initializing Spring DispatcherServlet 'dispatcherServlet'
六月 05, 2020 4:00:36 下午 org.springframework.web.servlet.FrameworkServlet initServletBean
信息: Initializing Servlet 'dispatcherServlet'
六月 05, 2020 4:00:36 下午 org.springframework.web.servlet.FrameworkServlet initServletBean
信息: Completed initialization in 5 ms
```

在浏览器输入对应url测试是否整合成功

http://localhost:8080/hello



Hello World !