第5章

构建多模块项目

教务辅助管理系统(Teaching Assistant Management System,TAMS)用于学生教师日常教学事务的辅助管理。本章主要介绍了项目的技术架构以及复合构建,并基于多模块方式分别创建了项目的三大部分:公共项目、前端项目、服务端项目。本章项目的顺利完成,为后续章节各模块的成功实现做好了项目准备。

5.1 教务辅助管理系统项目概述

教务辅助管理系统是一个基于 B/S 模式,用于学生、教师一些常规事项管理的项目。为了便于理解,这里对系统进行了功能调整、简化处理,使其业务逻辑简单、更易于理解,但又能满足响应式前后端开发技术的完整展现。

5.1.1 系统功能简介

系统功能主要包括 9 个部分,这里做简要说明。①首页:显示主界面及导航菜单。②用户登录:登录成功后,会生成 JWT 令牌以备其他功能验证使用。③用户注册:根据用户名、密码、E-mail、用户角色进行注册。用户名唯一。④消息推送:后台利用 WebClient、Sinks.Many 等技术向登录的用户推送通知、公告等短消息。⑤学院风采:展示各学院的文字介绍及视频介绍。各学院的视频文件,使用视频流分段加载。⑥学生查询:登录后用户可输入学生姓名或班级关键字模糊查询,并使用分布式内存网格 Hazelcast 进行数据共享。⑦招生一览:可查询指定学院某专业近 5 年招生数据情况,基于 gRPC+ECharts 显示招生图表。⑧资料上传:登录过的用户可上传资料文件,并使用 Spring WebFlux 文件流进行传送。⑨畅论空间:采用 Apache Kafka+WebSocket 技术进行消息传递,提供登录用户的讨论、学习交流场所。

5.1.2 系统技术架构

前端主要使用 RxJS＋Vue 3，后端基于 Spring WebFlux，数据库采用 PostgreSQL，并以 Netty 为服务器引擎。整个系统的技术架构如图 5-1 所示。

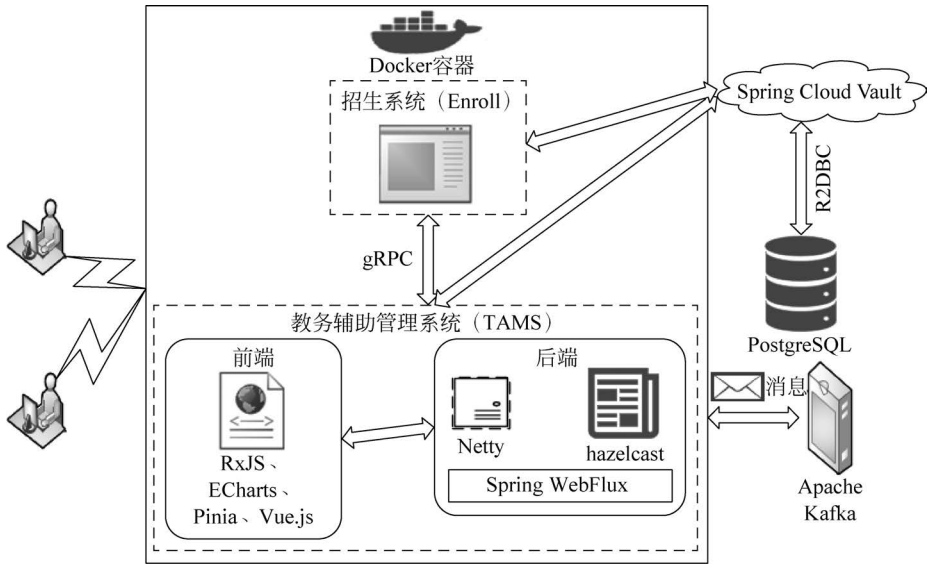


图 5-1 系统技术架构

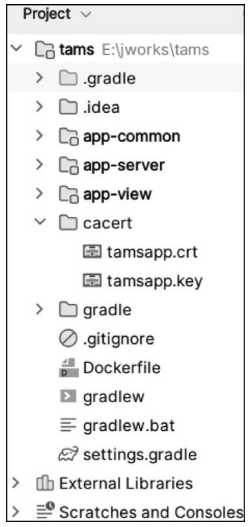


图 5-2 项目整体结构

整个系统包含两个部分：招生系统（Enroll）、教务辅助管理系统（TAMS）。Enroll 对外暴露招生数据接口，以供外部访问。TAMS 则通过 gRPC（google Remote Procedure Call，谷歌远程过程调用），从招生系统获取到招生数据。二者都在 Docker 容器内运行，并通过 Spring Cloud Vault，安全地获取 R2DBC 连接信息，据此访问 Docker 容器外部的数据库服务器 PostgreSQL。

TAMS 利用 Hazelcast 实时数据分发集群平台，在服务器结点之间分布式缓存、共享数据，以提高查询效率。同时，采用外部 Apache Kafka 作为消息处理平台。

5.1.3 系统的复合构建结构

整个系统采用复合构建的形式，根项目 tams 由三大各自独立的子项目构成：公共项目 app-common、前端项目 app-view、服务端项目 app-server，如图 5-2 所示。

Dockerfile 是 Docker 脚本文件，用于将项目发布到 Docker 容器。settings.gradle 配置了项目各组成模块之间的依赖关系。而 cacert 文件夹则存放了发布项目时需要用到的安全证书、私钥文件。这些文件，后续会适时详细介绍。

5.2 创建响应式根项目 TAMS

新建 Spring Reactive Web 项目 tams(请参阅 1.6 节),作为系统的根项目,并定义相关信息如下。

Name(项目名称): tams　　Location(存放位置): E:\rworks\chapter05
Language(语言): Java　　Type(构建类别): Gradle-Groovy
Group(组名): com.tams　　Artifact(打包声明): tams
Package name(包名): com.tams　　JDK: JDK 17
Java: 17　　Packaging(打包方式): jar
Spring Boot 版本: 3.1.4　　依赖项: Lombok、Spring Reactive Web

删除项目中的 build.gradle 文件。作为根项目,这里并不需要该文件,而是在各子项目内配置各自的 build.gradle。

5.3 添加公共项目 app-common

app-common 主要用来配置整个系统的公共配置,包括以下三大部分。

- server.common.gradle: 用来设置服务端模块的公共构建脚本。
- view.common.gradle: 用来设置前端各模块的公共构建脚本。
- reactor.grpc.gradle: 用于设置与远程招生系统(Enroll)有关的构建脚本。

添加公共项目 app-common 的具体步骤如下。

(1) 创建文件夹"app-common"。

在项目名称 tams 上单击鼠标右键,在弹出菜单中选择 New→Directory,输入文件夹名"app-common"即可。

(2) 修改根项目 tams 配置文件 settings.gradle 的内容。

```
rootProject.name = 'tams'
includeBuild 'app-common'                    //引用 app-common 中的构建逻辑、依赖关系
```

再单击 Load Gradle Changes 按钮(请参阅图 1-30)重载更改。

(3) 在 app-common 上新建文件夹 src,在 src 下创建子文件夹 main,再在 main 下创建文件夹 groovy。

(4) 在 app-common 下新建 build.gradle 文件,其内容为

```
plugins {
    id 'groovy-gradle-plugin'
}
```

这里使用的 groovy-gradle-plugin 插件,可为项目 src/main 中的脚本提供构建支持。所以,build.gradle 文件不要错误地放置在 app-common/src 文件夹下,否则,无法对 src/main/groovy 下的构建脚本进行解析构建。

5.3.1 服务端构建脚本 server.common.gradle

服务端构建脚本 server.common.gradle 主要是设置服务模块的全局公共配置信息，例如，各模块打包成 jar 文件时的文件名、版本号、资源仓库地址、Java 编译版本、公共依赖项、编译时的字符编码等。在 app-common/src/main/groovy 文件夹下新建 server.common.gradle，添加构建脚本：

```
plugins {                                          //使用 Gradle 内置的 java、base 插件
    id 'java'
    id 'base'
}

java {
    sourceCompatibility = JavaLanguageVersion.of(17)     //Java 版本
}
group = 'com.tams.server'                          //项目的组名
version = '1.0'                                    //项目版本号
def springVer = '3.1.4'                            // spring-boot-starter 版本号
def lombokVer = '1.18.28'                          // lombok 版本号
base {
    /* 定义所有子模块打包后的 jar 文件名格式:tams.项目名.server
    打包后会附加版本号,例如:tams.chat-room.server-1.0.jar */
    archivesName.set('tams.' + project.name + '.server')
    //设置最终打包后的 jar 文件存放的文件夹为 tams-jars
    libsDirectory.set(layout.buildDirectory.dir('tams-jars') as Provider<? extends Directory>)
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor            //compileOnly 继承自注解解释器 lombok
    }
}
repositories {                                     //配置资源仓库,优先使用阿里云
    maven {
        url 'https://maven.aliyun.com/repository/google'
    }
    mavenLocal()
    mavenCentral()
}

dependencies {                                     //配置项目的依赖项
   implementation "org.springframework.boot:spring-boot-starter-data-r2dbc: ${springVer}"
   implementation "org.springframework.boot:spring-boot-starter-webflux: ${springVer}"
   implementation "org.springframework.cloud:spring-cloud-starter-vault-config:4.0.1"
   implementation "com.hazelcast:hazelcast-spring:5.3.6"       //hazelcast 分布式内存网格
   implementation "io.projectreactor.kafka:reactor-kafka:1.3.22"              //Kafka 消息处理器
   runtimeOnly "org.postgresql:r2dbc-postgresql:1.0.2.RELEASE"//数据库 PostgreSQL
   compileOnly "org.projectlombok:lombok: ${lombokVer}"
   annotationProcessor "org.projectlombok:lombok: ${lombokVer}" //注解处理器
   implementation "org.modelmapper.extensions:modelmapper-spring:3.2.0"

   testImplementation "org.springframework.boot:spring-boot-starter-test: ${springVer}"
   testImplementation "io.projectreactor:reactor-test:3.5.9"
}
```

```
tasks.withType(JavaCompile).configureEach {                    //编译任务
    options.encoding = "UTF-8"                                 //编译时的编码统一设置为 UTF-8
}
tasks.withType(Copy).configureEach {                           //设置复制策略
    duplicatesStrategy = DuplicatesStrategy.EXCLUDE            //若重复则忽略
}
tasks.named('test', Test) {                                    //测试任务
    useJUnitPlatform()
}
```

5.3.2　前端构建脚本 view.common.gradle

前端构建脚本 view.common.gradle 用来配置前端模块的全局公共配置信息，例如，各前端模块打包成 jar 文件时的文件名、版本号、公共依赖项（例如，HTML 页面文件所依赖的 vue.global.prod.min.js、rxjs.umd.min.js 等）。在 app-common/src/main/groovy 文件夹下新建构建文件 view.common.gradle，添加构建脚本：

```
plugins {
    id 'java'
    id 'base'
}
group = 'com.tams.view'
version = '1.0'
base {
    archivesName.set('tams.' + project.name + '.view')
}
dependencies {
    implementation project(':app-view:public')                 //依赖于 app-view 项目的 public 子模块
}
```

由于这时候 app-view 项目的 public 子模块还没来得及创建，因此会提示构建错误，先不用理睬，继续下一步操作。

5.3.3　gRPC 构建脚本 reactor.grpc.gradle

gRPC 构建脚本 reactor.grpc.gradle，用来配置与另外一个招生系统（Enroll）进行远程过程调用的脚本代码。在 app-common/src/main/groovy 文件夹下新建 reactor.grpc.gradle，添加构建脚本：

```
plugins {
    id 'java'
    id 'base'
}
group = 'com.tams.grpc'
version = '1.0'
def ioGrpcVer = '1.57.2'                                       // io.grpc 版本号
def reactorGrpcVer = '1.2.4'                                   // reactor grpc 版本号
base {
    archivesName.set('tams.' + project.name + '.grpc')
}
dependencies {                                                 //gRPC 所需的依赖项
    implementation "io.grpc:grpc-netty-shaded:${ioGrpcVer}"
```

```
    implementation "io.grpc:grpc-protobuf:${ioGrpcVer}"
    implementation "io.grpc:grpc-stub:${ioGrpcVer}"
    implementation "com.salesforce.servicelibs:reactor-grpc:${reactorGrpcVer}"
    implementation "com.salesforce.servicelibs:grpc-spring:0.8.1"
    implementation "com.salesforce.servicelibs:reactor-grpc-stub:${reactorGrpcVer}"
}
```

现在，app-common 项目结构如图 5-3 所示。

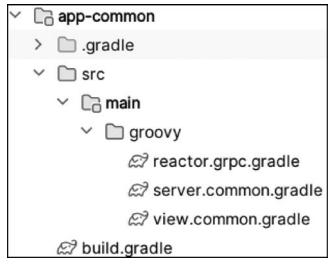


图 5-3　app-common 项目结构

5.4 添加前端项目 app-view

前端项目 app-view，基于 HTML、CSS、RxJS、Vue.js、Pinia、ECharts 等技术，用于构建教务辅助管理系统的人机交互页面。

5.4.1 新建 app-view

在根项目 tams 下创建文件夹“app-view”，修改根项目 tams 的配置文件 settings.gradle 并重载更改：

```
ootProject.name = 'tams'
includeBuild 'app-common'          //引用 app-common 中的构建逻辑、依赖关系
include 'app-view'                 //将 app-view 设置为根项目 tams 构建生成的一部分
```

然后，在 app-view 下新建文件 settings.gradle，通过该文件设置 app-view 的名称，其内容非常简单：

```
rootProject.name = 'app-view'
```

5.4.2 添加子模块 home

现在，可添加 app-view 项目的各子模块了。首先添加子模块 home，该模块定义了 TAMS 项目的前端页面的入口主页。添加子模块 home 的步骤如下。

(1) 新建子模块。在 app-view 上单击鼠标右键，选择 New→Module，弹出 New Module 对话框，如图 5-4 所示。选择左边的 New Module，填写相应信息：

Name(模块名称)：home　　Location(保存位置)：E:\rworks\chapter05\tams\app-view

Parent(父项目)：tams　　GroupId(分组 ID)：com.tams

ArtifactId(打包 ID)：home

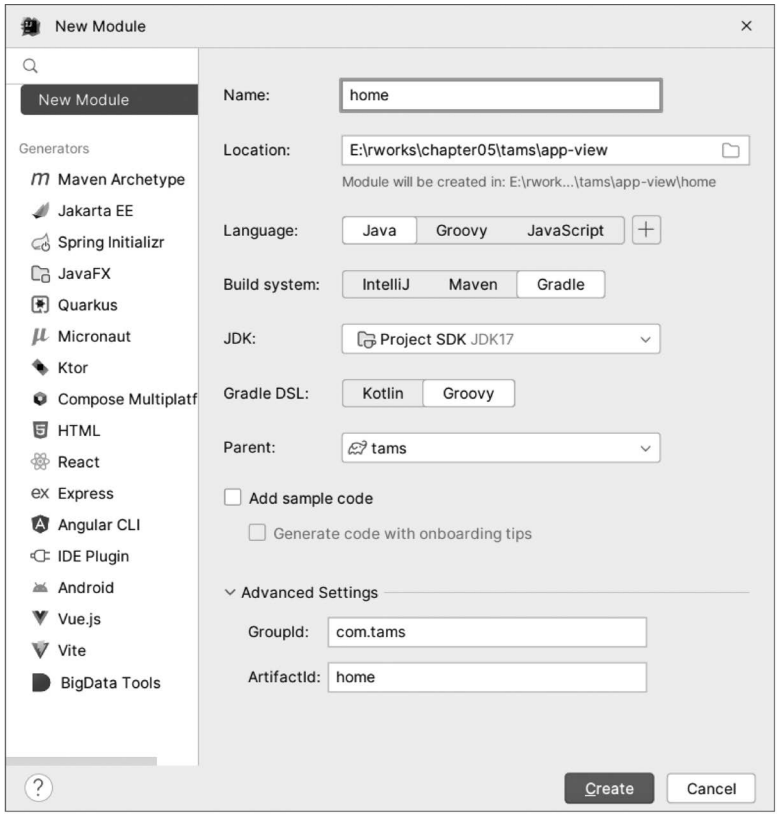


图 5-4　New Module 对话框

单击 Create 按钮，完成创建过程。

（2）配置 home/build.gradle。删除自动生成的内容，修改其内容为

```
plugins {
    id 'view.common'              //引入 app - common 中 view.common.gradle 定义的内容
}
```

（3）删除不再需要的 home/src/main/java 文件夹、home/src/test 模块，并创建好其他后续需要的文件夹及文件，例如 static 文件夹、image 文件夹、home.html 等。最终准备好的项目结构如图 5-5 所示。

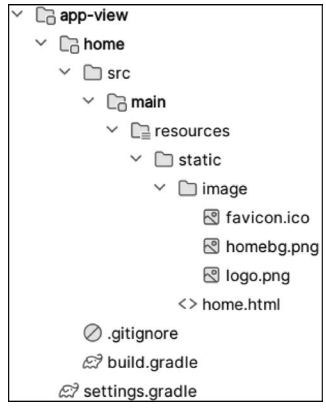


图 5-5　home 子模块结构

（4）在 resources/static 下新建 home.html，先将内容简单设置为

```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>教务辅助管理系统 -- 官方主页</title>
</head>
<body>
教务辅助管理系统，建设中...
</body></html>
```

5.4.3 添加子模块 public

子模块 public 用于配置前端项目的公共资源，例如，JS 支撑文件、插件文件夹、状态管理文件夹等。

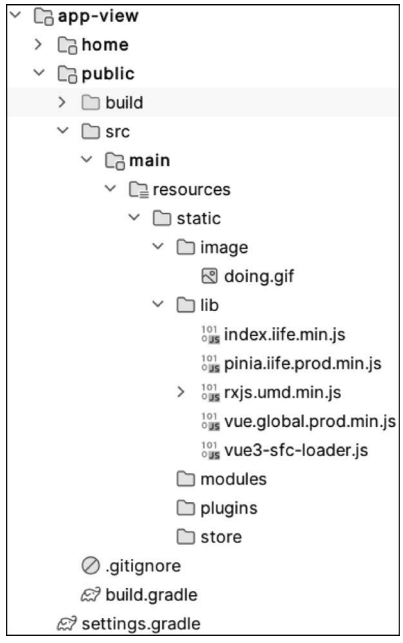


图 5-6 public 子模块结构

（1）创建子模块 public。按照 5.4.1 节的方法，新建子模块 public，同样准备好各文件夹及 JS 支撑文件。子模块 public 的最终结构如图 5-6 所示。

- image 文件夹：存放前端公共图片，例如，公用进度处理图片 doing.gif。
- lib 文件夹：存放 RxJS、Vue.js、Pinia、vue3-sfc-loader 等 JS 支持文件。请添加好这些 JS 支持文件。
- modules 文件夹：存放公共 SFC 组件。
- plugins 文件夹：存放自定义 Vue 插件。
- store 文件夹：存放用于 Pinia 状态管理的 JS 文件。

（2）修改 build.gradle。删除大部分自动生成的内容，只需要保留：

```
plugins {
    id 'java'
}
```

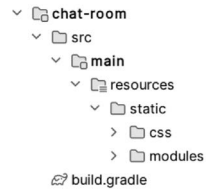
（3）删除不再需要的 public/src/main/java 文件夹、public/src/test 模块。

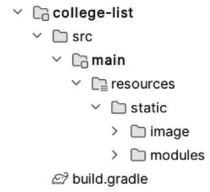
5.4.4 添加其他子模块

按照上述处理思路，添加其他前端子模块。表 5-1 列出了需要添加的前端各子模块的主要配置。

表 5-1 各子模块配置

模 块 名	模块文件夹结构	build.gradle	模 块 说 明

chat-room		plugins {
		id 'view.common'
		}
			畅论空间

college-list		plugins {
		id 'view.common'
		}
			学院风采

续表

模　块　名	模块文件夹结构	build. gradle	模 块 说 明

enroll-chart	enroll-chart
	src
	main
	resources
	static
	modules
	build.gradle
		plugins {
		id 'view.common'
		}
			招生一览

file-service	file-service
	src
	main
	resources
	static
	image
	modules
	build.gradle
		plugins {
		id 'view.common'
		}
			资料上传

student-query	student-query
	src
	main
	resources
	static
	image
	modules
	build.gradle
		plugins {
		id 'view.common'
		}
			学生查询

user-service	user-service
	src
	main
	resources
	static
	image
	modules
	build.gradle
		plugins {
		id 'view.common'
		}
			用户服务(用户登录、用户注册)

最终 app-view 项目结构如图 5-7 所示。

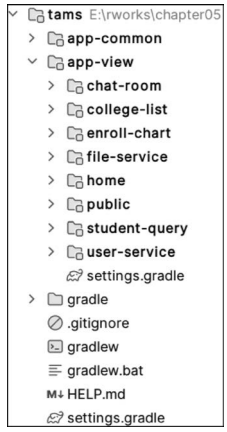


图 5-7　app-view 项目结构

5.5　添加服务端项目 app-server

服务端项目 app-server 基于 Spring WebFlux 技术，实现并提供教务辅助管理系统后端的各种

业务逻辑处理服务。

5.5.1 新建 app-server

与添加前端项目 app-view 类似，同样在根项目 tams 下创建文件夹“app-server”，并在根项目 tams 的 settings.gradle 文件中添加：

```
include 'app-server'        //将 app-server 设置为根项目 tams 构建生成的一部分
```

记得重载更改！然后，在 app-server 下新建文件 settings.gradle 设置服务端项目名称：

```
rootProject.name = 'app-server'
```

5.5.2 添加子模块 app-boot

现在，可添加 app-server 项目的各子模块了。先添加子模块 app-boot，该模块定义了 TAMS 项目的服务端应用入口，以及相应的依赖项。添加子模块 app-boot 的步骤如下。

(1) 新建子模块 app-boot。利用 New Module 对话框(请参阅 5.4.1 节)，定义 app-boot 相应信息如下。

Name(模块名称)：app-boot　Location(保存位置)：E:\rworks\chapter05\tams\app-server

Parent(父项目)：tams　GroupId(分组 ID)：com.tams

ArtifactId(打包 ID)：app-boot

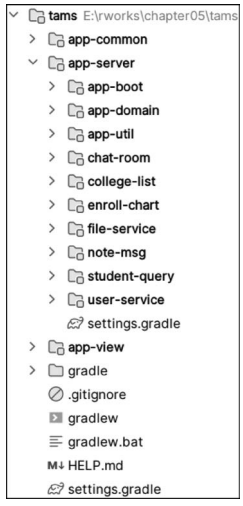


图 5-8　app-server 项目结构

(2) 删除不再需要的根项目 tams 的 src 文件夹。

至于具体的服务端应用入口类以及 build.gradle 构建脚本，将在第 6 章中详细介绍。

5.5.3 添加其他子模块

按照上述方法，继续添加其他子模块，具体如下。

app-domain：实体类子模块　app-util：工具类子模块

chat-room：畅论空间子模块　college-list：学院风采子模块

enroll-chart：招生一览子模块　file-service：资料上传子模块

note-msg：消息推送子模块　student-query：学生查询子模块

user-service：用户服务子模块

各模块的配置，后续章节具体实现时再详细介绍，这里暂时采用默认设置。现在，app-server 项目结构如图 5-8 所示。

5.6 最终的配置文件 settings.gradle

经过优化后的根项目 tams 的配置文件 settings.gradle 内容如下。

```
rootProject.name = 'tams'
includeBuild 'app-common'
```

```
include 'app-view:home'
findProject(':app-view:home')?.name = 'home'
include 'app-view:public'
findProject(':app-view:public')?.name = 'public'
include 'app-view:chat-room'
findProject(':app-view:chat-room')?.name = 'chat-room'
include 'app-view:college-list'
findProject(':app-view:college-list')?.name = 'college-list'
include 'app-view:enroll-chart'
findProject(':app-view:enroll-chart')?.name = 'enroll-chart'
include 'app-view:file-service'
findProject(':app-view:file-service')?.name = 'file-service'
include 'app-view:student-query'
findProject(':app-view:student-query')?.name = 'student-query'
include 'app-view:user-service'
findProject(':app-view:user-service')?.name = 'user-service'

include 'app-server:app-boot'
findProject(':app-server:app-boot')?.name = 'app-boot'
include 'app-server:app-domain'
findProject(':app-server:app-domain')?.name = 'app-domain'
include 'app-server:app-util'
findProject(':app-server:app-util')?.name = 'app-util'
include 'app-server:chat-room'
findProject(':app-server:chat-room')?.name = 'chat-room'
include 'app-server:college-list'
findProject(':app-server:college-list')?.name = 'college-list'
include 'app-server:enroll-chart'
findProject(':app-server:enroll-chart')?.name = 'enroll-chart'
include 'app-server:file-service'
findProject(':app-server:file-service')?.name = 'file-service'
include 'app-server:note-msg'
findProject(':app-server:note-msg')?.name = 'note-msg'
include 'app-server:student-query'
findProject(':app-server:student-query')?.name = 'student-query'
include 'app-server:user-service'
findProject(':app-server:user-service')?.name = 'user-service'
```

5.7 项目打包后的模块结构

至此，整个项目的结构已经建立起来。按照这样的复合结构思想设计后，当项目各模块最终编写完成、打包后，tams-jars 文件夹下会构建生成 tams.app-boot.server-1.0.jar 文件，该文件的 BOOT-INF/lib 文件夹下会包含各前后端模块打包后生成的 jar 文件，例如：

```
tams.user-service.server-1.0.jar
tams.user-service.view-1.0.jar
tams.chat-room.server-1.0.jar
tams.chat-room.view-1.0.jar
```

模块文件的构成结构清晰，非常方便管理，如图 5-9 所示。

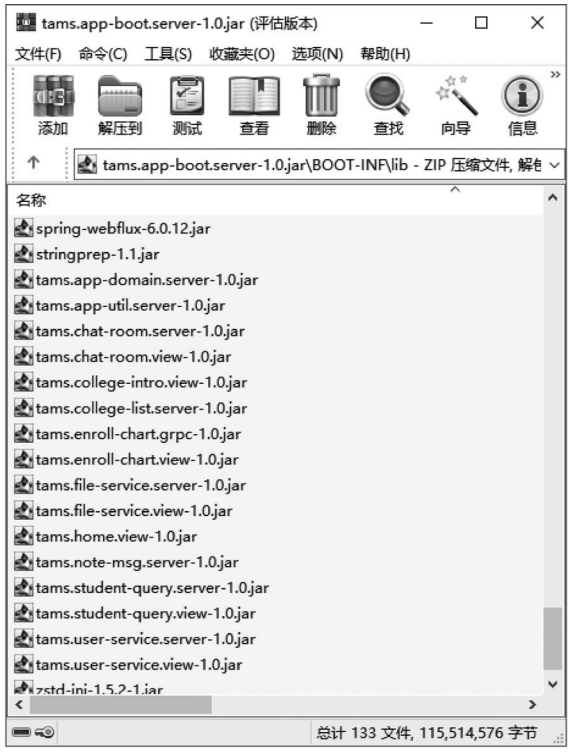


图 5-9　打包后的项目文件