Sharing Code Between an Intellij Server Project and an Android Studio Client Project

If you created your server in Intellij you will need to make some changes to your Intellij server project and your Android Studio client project to share the code that is common to both projects. There are two approaches you can take to sharing the code.

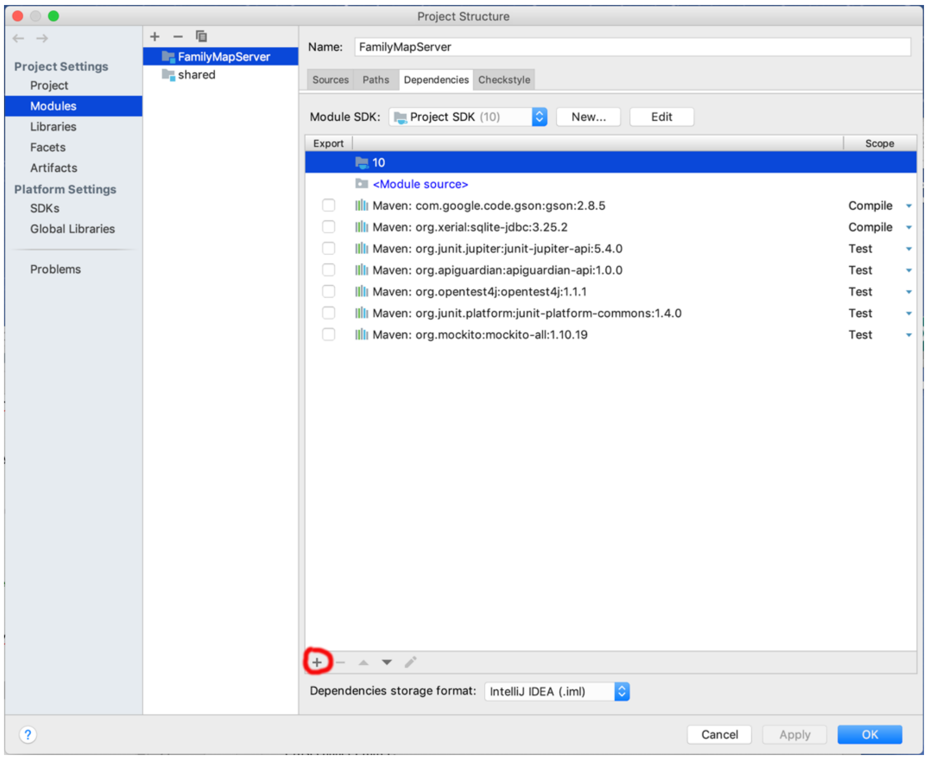
The first approach is to create a shared module in your server project, copy the code to be shared into the new module, build a .jar file from the new module, and include the .jar file as a dependency in your Android client project. The disadvantage of this approach is that you will need to rebuild and re-copy your .jar file if you make changes to the shared code in the server project.

The second approach is to create two new modules in your Android Studio Project, one for the Family Map Server and the other for the code to be shared by both your client and your server. You will then need to copy or move the code to the appropriate modules and move any other required files (web files, json files, database, etc) into the Android project. This approach will take more work up-front, but once your module dependencies are setup properly, you will be able to make changes to the code in the shared module and have those changes be immediately and automatically visible to both the client and the server modules.

These instructions describe the first approach. However, the steps will help you see how to take the second approach if you choose to do that instead.

Create a new shared module

1. *File -> New -> Module…*
2. Move the files that should be shared into the source directory of the new module
3. Make the original project dependent on the new shared module:
    a. Open project settings: *File -> Project Structure…*
    b. Go to Modules, select the project (not the new shared module) and click '+' (the plus symbol near the bottom of the dialog, not the one near the upper right corner)



Name: FamilyMapServer

Module SDK: Project SDK (10)

Export		Scope
	10	
	<Module source>	
☐	Maven: com.google.code.gson:gson:2.8.5	Compile
☐	Maven: org.xerial:sqlite-jdbc:3.25.2	Compile
☐	Maven: org.junit.jupiter:junit-jupiter-api:5.4.0	Test
☐	Maven: org.apiguardian:apiguardian-api:1.0.0	Test
☐	Maven: org.opentest4j:opentest4j:1.1.1	Test
☐	Maven: org.junit.platform:junit-platform-commons:1.4.0	Test
☐	Maven: org.mockito:mockito-all:1.10.19	Test

Dependencies storage format: IntelliJ IDEA (.iml)

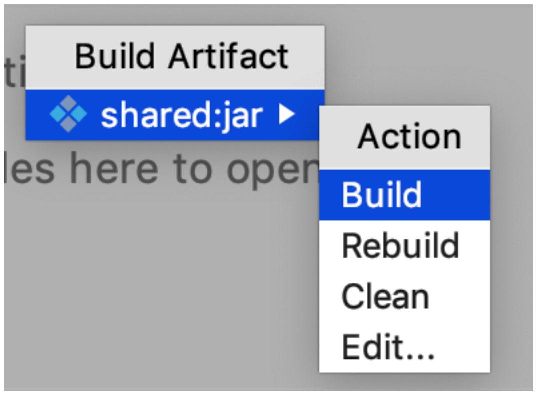
c. Select "Module Dependency…"
d. Select the new module you created in step one and press OK until you return to the main window.

Create a .jar artifact for the new module

1. Open project settings: *File -> Project Structure…*
2. Select "Artifacts" and click '+'
3. Select *Jar -> From modules with dependencies…*
4. Select the new shared module from the Module file of the dialog box and press OK until you return to the main window

Build the .jar file

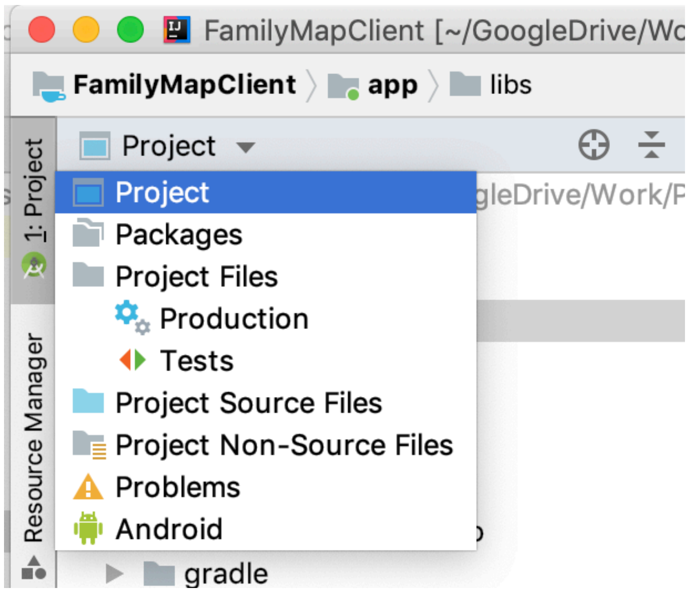
1. *Build -> Build Artifacts…*
2. Select the jar file and click the "Build" action



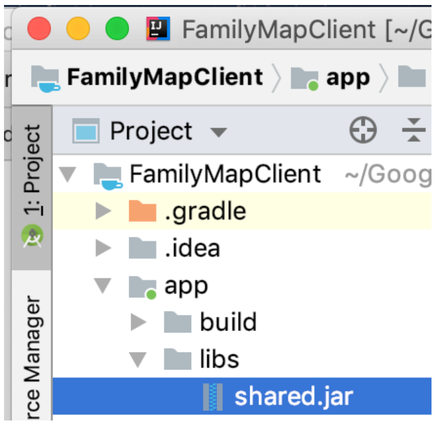
3. Find the .jar file in the “out/artifacts” directory of the project (you will need to search one or more directories down from “artifacts” to find the .jar file).

Add the .jar File as a Dependency in your Android Client Project

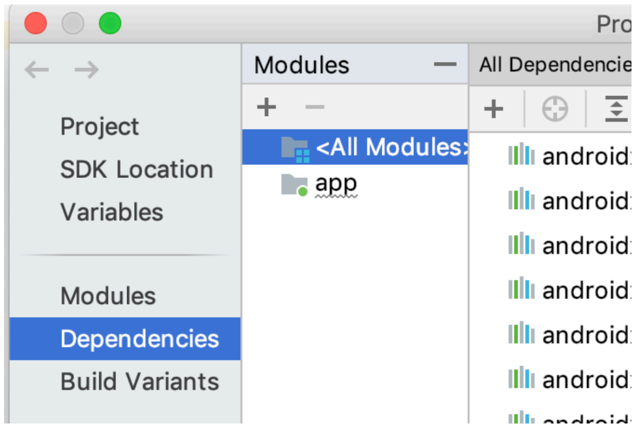
1. Switch to the project view in your Android project



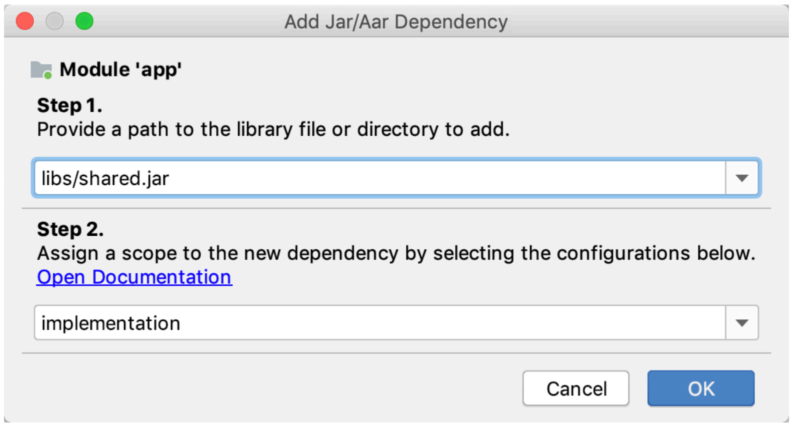
2. Copy the .jar file into the ‘libs’ folder of your project’s ‘app’ module directory (or if you have renamed the ‘app’ module, copy it into the ‘libs’ folder of the renamed module)
    a. You should not need to create a ‘libs’ folder. Your module should already have one. If it does not, create a ‘libs’ folder under the ‘app’ folder.



3. Open project settings: *File -> Project Structure and select 'Dependencies'*



4. Click the '+' sign under "All Dependencies" and then select "Jar Dependency"
5. Enter or select 'libs/shared.jar' (assuming your .jar file is named shared.jar) in the first drop down and select 'implementation' in the second.


Add Jar/Aar Dependency

Module 'app'

Step 1.
Provide a path to the library file or directory to add.

libs/shared.jar

Step 2.
Assign a scope to the new dependency by selecting the configurations below.
Open Documentation

implementation

6. Press OK until you return to the main window.

Troubleshooting

With some versions of Android Studio, you may get an error when trying to build or run your Android project after completing the above steps. The error will say something about a Jetifier transform or conversion not working. If you receive this error, you should be able to correct it by adding the following to your *gradle.properties* file:

```
android.jetifier.blacklist=shared.jar
```

If the .jar you imported has a different name, replace shared.jar with the name of the .jar file you are trying to share. Be careful with case of your jar file name. It has to match exactly.