76. Adapting an Android Studio Project for Instants Apps

In addition to being able to include Instant Apps support in a new project, it is also important to be able to convert an existing Android Studio project to provide instant app installation and launch capabilities.

In this chapter, the AppLinking project completed in the chapter entitled (*"An Android Studio App Links Tutorial"*) will be modified to add instant apps support.

76.1 Getting Started

As previously outlined, the objective of this chapter is to take the existing AppLinking project and modify it to support instant apps. The completed project will consist of an instant app module, a base feature module containing the main activity, and a second feature module containing the landmark detail activity. The app link already configured within the project will be used to install and launch one of these instant app feature modules. A second app link will be added during this tutorial for the other feature module.

Begin by launching Android Studio and opening the completed AppLinking project. If you have not yet completed this project, refer to the *"An Android Studio App Links Tutorial"* chapter, or load the completed version of the app from the *AppLinking_completed* folder of the code samples download available from the following URL:

http://www.ebookfrenzy.com/retail/androidstudio30/index.php

Creating the Base Feature Module

The project currently contains an application module named *app* which uses the *com.android.application* build plugin. This module will serve as the base feature module for the modified project, so needs to be given a more descriptive name. Within the project tool window, right-click on the *app* entry and select the *Refactor -> Rename...* option from the menu. In the Rename Module dialog, change the module name to *applinkingbase* before clicking on the *OK* button.

Although the module has been renamed, it is still configured as an application. To resolve this, edit the applinking-base *build.gradle* file (*Gradle Scripts -> build.gradle (Module: applinkingbase)* and change the plugin declaration to reference *com.android.feature* instead of *com.android.application*. Since this is no longer an application module, it also no longer makes sense to have an application Id assigned, so also remove this declaration from the build file. Finally, the build file needs to be declared as the base feature module for the project:

```
apply plugin: 'com.android.feature'
.
.
android {
    baseFeature true
    compileSdkVersion 26
    buildToolsVersion "26.0.2"
    defaultConfig {
        applicationId "com.ebookfrenzy.applinking"
        minSdkVersion 25
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
                  "android.support.test.runner.AndroidJUnitRunner"
    }
.
.
.
```

The next step is to add an application module to the project that will allow the app to continue to support the standard APK app installation mechanism in addition to supporting instant app installations.

76.2 Adding the Application APK Module

At this stage we have a base feature module containing all of the code for the project. The project will still need to be able to generate standard application-type APK files during the build process. This can be achieved by adding an app module to the project and configuring it to contain the *applinkingbase* feature module. To add the new module, select the Android Studio *File -> New Module...* menu option and select the *Phone and Tablet* option from the selection panel (Figure 76-1). Once selected, click on the *Next* button to

proceed:

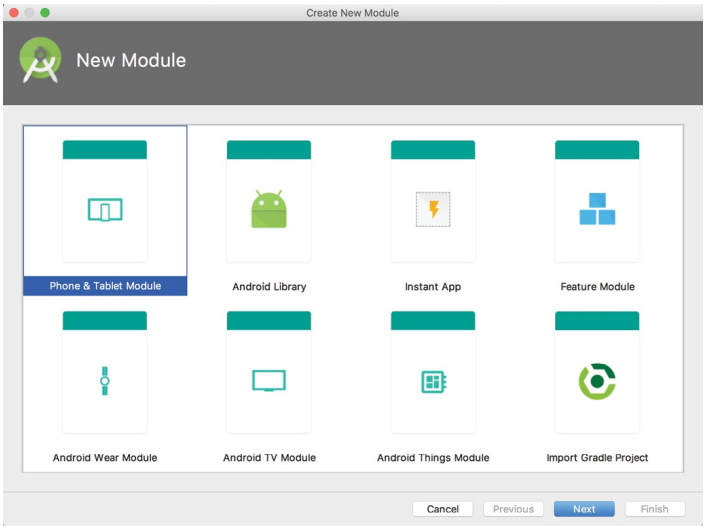

Figure 76-1

On the next screen, set the application/library name to *AppLinking APK* and the module name to *applinkingapk*. Set the minimum SDK to *API 25: Android 7.1.1 Nougat* before clicking on the *Next* button:

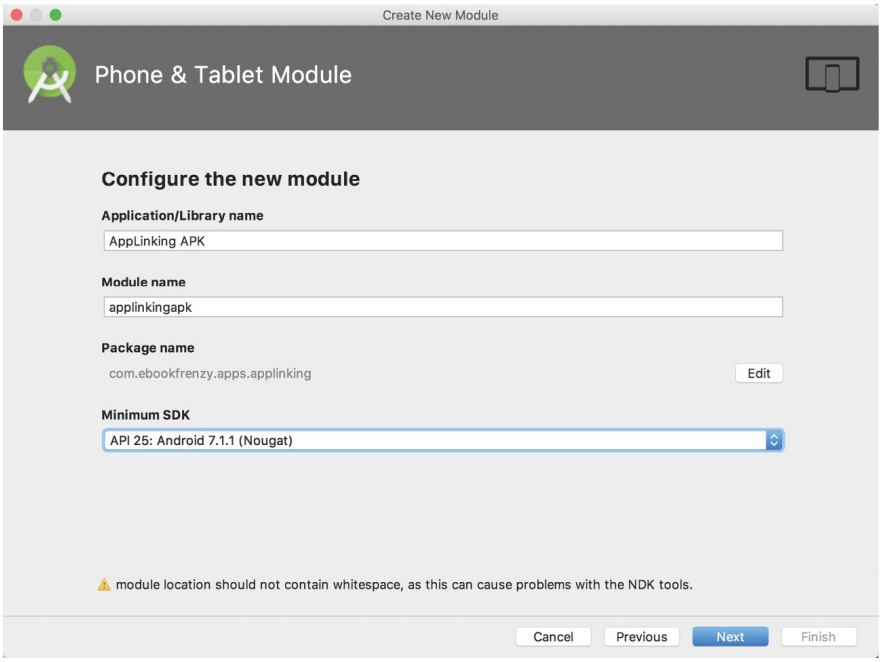

Figure 76-2

Since this module is simply a container within which the base feature module will be referenced, it does not need to have any activities of its own. On the final screen, therefore, select the *Add No Activity* option before clicking on the *Finish* button.

When Android Studio generates the new application module, a number of default dependencies will have been added to the module's *build.gradle* file. Since the only dependency that the module actually has is the base feature module, the default dependencies need to be removed from the *build.gradle (applinkingapk)* file and replaced with a reference to the *applinkingbase* module:

```
apply plugin: 'com.android.application'

android {
.
.
.
dependencies {
    implementation project(':applinkingbase')
}
```

Note that since *applinkingapk* is an application module, the build file correctly applies the *com.android.application* plugin.

At this point in the chapter, the original application module has been converted to a base feature module and a new application module has been added and configured to contain the base module. Check that the applinkingapk application module compiles and runs without any problems by selecting it in the toolbar run target menu and clicking on the run button:

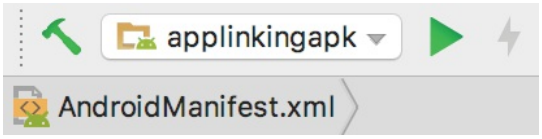

Figure 76-3

Verify that the app launches and functions as expected. Assuming that the app still works, it is time to begin adding instant app support.

76.3 Adding an Instant App Module

The project now has a base feature module and an application module used for creating a standard APK for the project. The next step is to add an instant app module to the project. Begin by selecting the Android Studio *File -> New Module…* menu option and selecting the *Instant App* option in the selection panel:

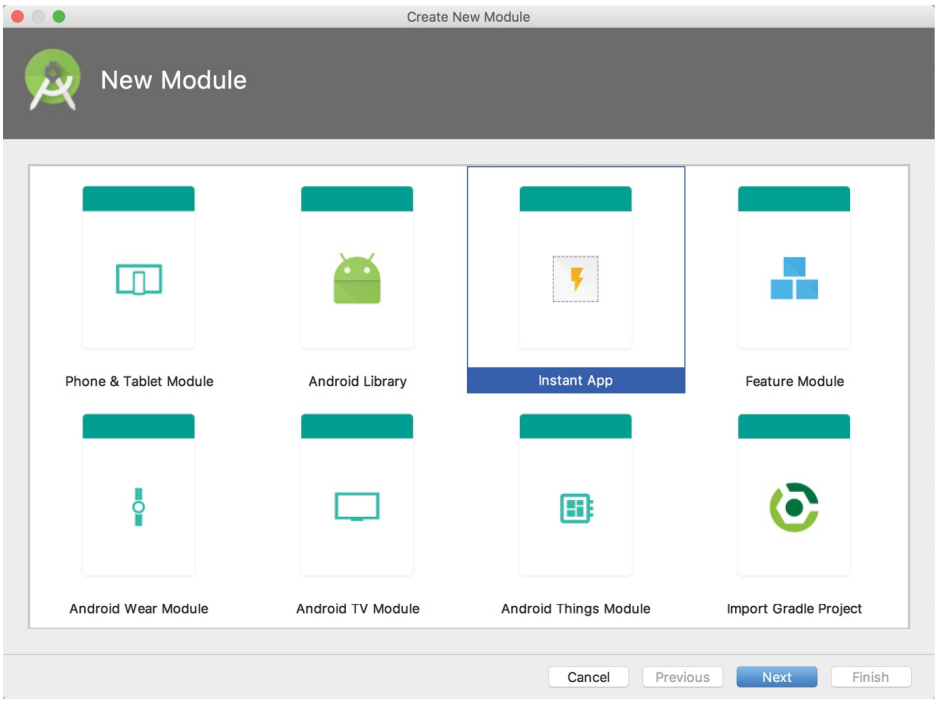

Figure 76-4

Click on the *Next* button, name the module *applinkinginstantapp* and click on the *Finish* button. As with the applinkingapk module, the only dependency for the instant app module is the base feature module. Edit the *build.gradle (applinkinginstantapp)* file and modify it as follows to add this dependency:

```
apply plugin: 'com.android.instantapp'

dependencies {
    implementation project(":applinkingbase")
}
```

Now that the instant app module has been declared, the project is ready to be tested as an instant app. Before proceeding, however, the current standard (i.e. non-instant app APK) for the project must be removed from the device or emulator on which testing is being performed. Launch the Settings app, navigate to the list of installed apps and select and uninstall the *AppLinking APK* app.

76.4 Testing the Instant App

Within the Android Studio toolbar, select *applinkinginstantapp* from the run menu as shown in Figure 76-5:

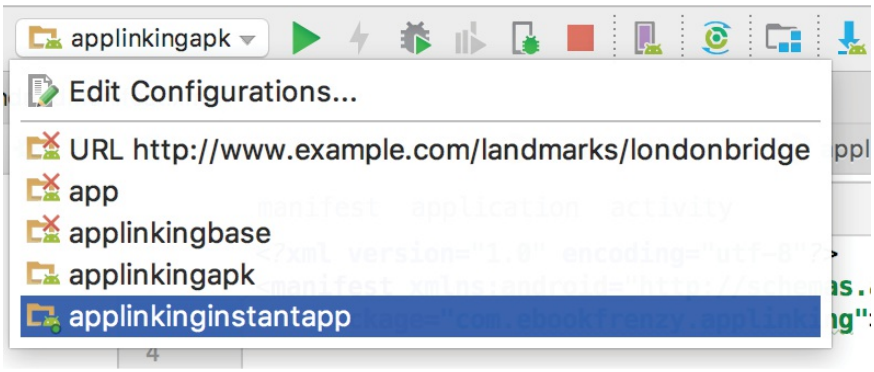

Figure 76-5

Display the menu again, this time selecting the *Edit Configurations…* option. In the launch options section of the configuration dialog, configure a launch URL containing the londonbridge landmark path as illustrated in Figure 76-6:

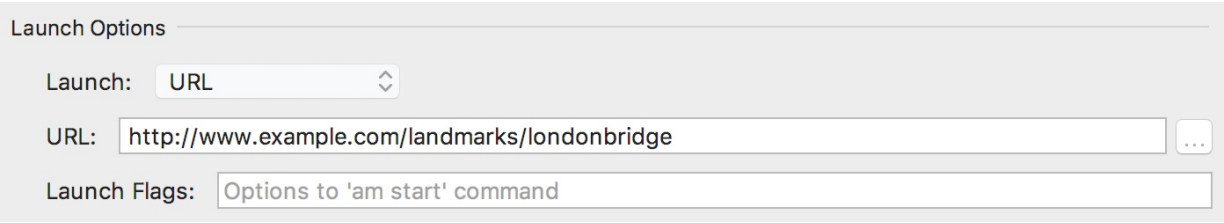

Figure 76-6

Click on the *Apply* button followed by the *OK* button to commit the change, then launch the instant app using the run button. After the build completes, the instant app will be installed and launched using the URL and display the landmark activity populated with London Bridge information.

On the device or emulator, open the Settings app and navigate to the list of installed apps. The AppLinking app icon will now include a lightning bolt indicating that this is an instant app:

Figure 76-7

76.5 Summary

This chapter has outlined how to modify an existing Android Studio app project to add Instant App support. This involved converting the existing app to the base feature module and then creating and configuring both the app

and instant app modules, both of which have the base feature module as a dependency. The app was then tested using the previously configured app link. The next chapter will continue with the same project, this time converting it to a multi-feature project.