77. Creating Multi-Feature Instant Apps

The previous chapter took a project designed only for deployment via app installation and modified it to include instant app support. This chapter will take the same project and separate the landmark detail activity into a second feature module to create a multi-feature app project.

77.1 Adding the Second App Link

As currently configured, only the landmark activity is able to be launched using an app link. An app link now also needs to be set up for the main activity. Display the App Link Assistant (*Tools -> App Links Assistant*) and add a new link for (http://www.example.com/home) configured to launch AppLinkingActivity as illustrated in Figure 77-1 below:

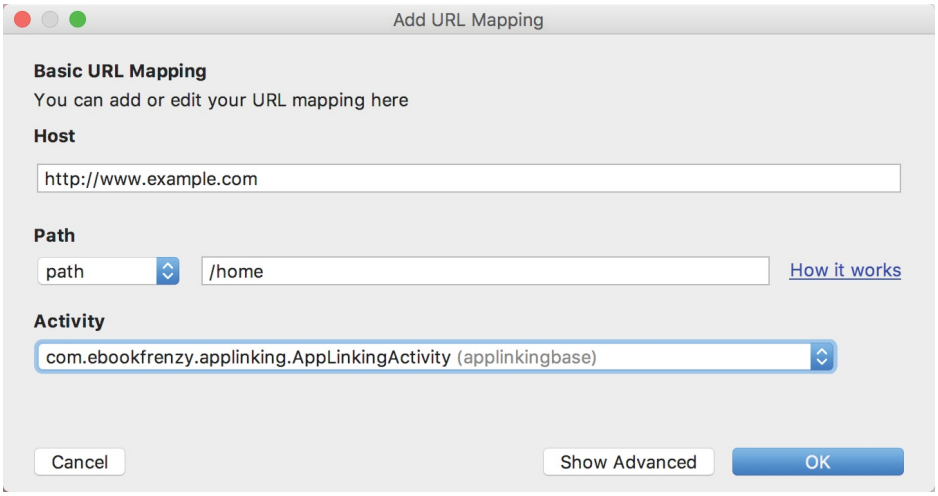

Figure 77-1

Repeat this step to add an entry for the URL using the HTTPS protocol.

Edit the *applinkinginstantapp* run configuration once again, change the launch URL to *http://www.example.com/home* and then run the instant app. When the app launches, the AppLinkingActivity screen should appear.

77.2 Creating a Second Feature Module

The final step in this chapter is to extract the LandmarkActivity class from the base module and place it in a separate feature module.

Begin by selecting the *File -> New Module…* menu option and clicking on the

Feature Module option:

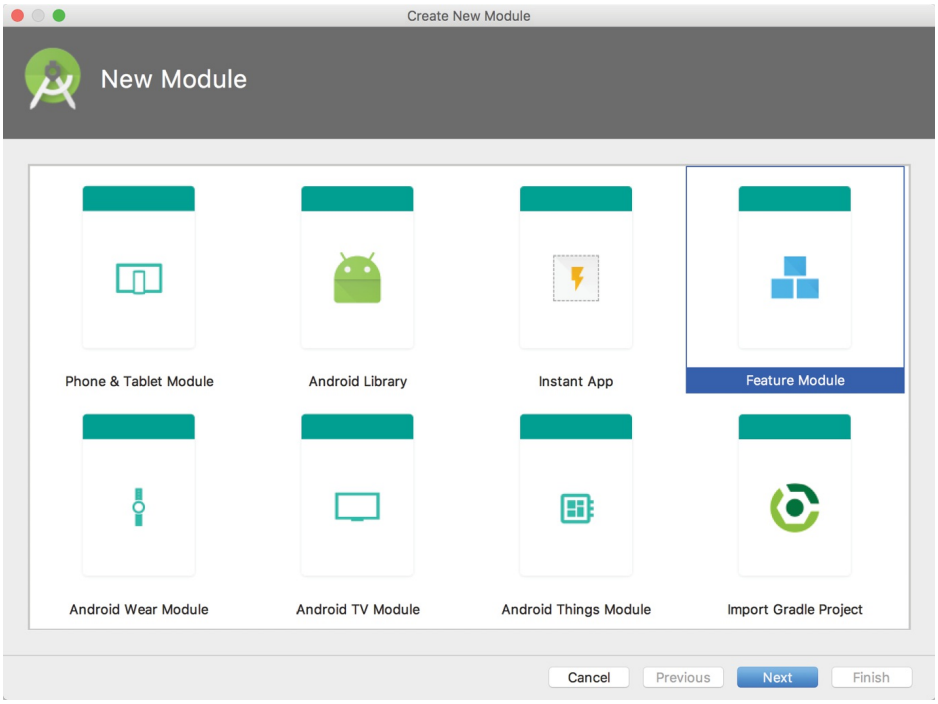

Figure 77-2

Click *Next* and, on the module configuration screen, change the library name to *AppLinking Landmark*, the module name to *applinkinglandmark* and the package name to *com.example.applinkinglandmark*:

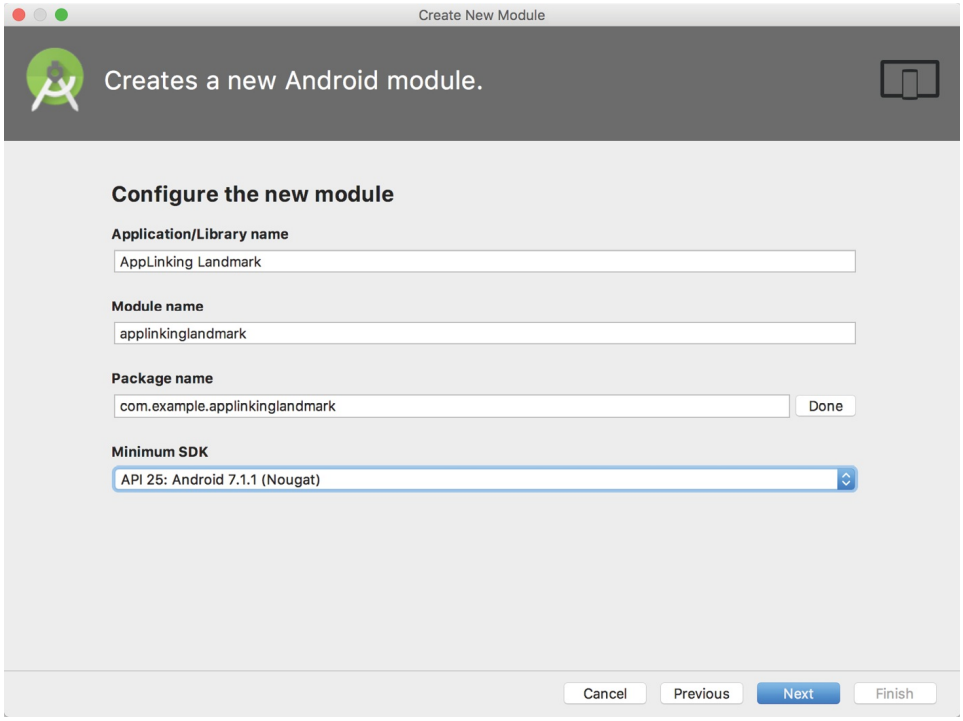

Figure 77-3

Click on the *Next* button, select the *Add No Activity* option and click on *Finish*. After the new module has been created, edit the *build.gradle (Module: applinkinglandmark)* file and replace the default dependencies with a reference to the base feature module:

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

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

Now that the landmark activity is no longer going to be in the base module, the new landmark feature module needs to be added as a dependency to both the instant app and app modules. Begin by editing the *build.gradle (Module: applinkingapk)* file and adding the landmark module dependency:

```
dependencies {
    implementation project(':applinkingbase')
    implementation project(':applinkinglandmark')
}
```

Repeat the above step, this time within the *build.gradle (Module: applinkinginstantapp)* file.

Finally, edit the *build.gradle (Module: applinkingbase)* file and add the new module to the existing list of dependencies if it has not already been added by Android Studio, together with the application project directive:

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
.
.
.
    application project(":applinkingapk")
    feature project(":applinkinglandmark")
}
```

77.3 Moving the Landmark Activity to the New Feature Module

With the preparation work complete, the LandmarkActivity class is ready to

be moved from the base feature module to the new feature module. This will require the creation of an *activity* folder within the landmark module. Locate the *com.example.applinkinglandmark* folder in the Project tool window (located under *applinkinglandmark -> java*), right-click on it and select the *New -> Package…* menu option:

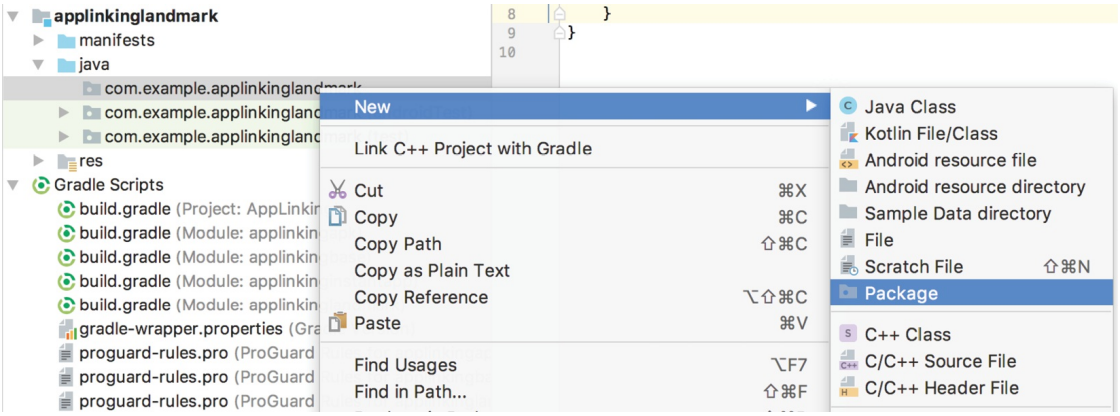

Figure 77-4

In the new package dialog, enter *activity* and click on *OK*.

Locate the *LandmarkActivity.java* file (located under *applinkingbase -> java -> com.ebookfrenzy.applinking*), right-click on it and select the *Cut* menu option. Move back to the newly added *activity* entry within the landmark module and right-click again, this time selecting the *Paste* menu option. When the Move dialog appears, click the *Refactor* button and, in the *Problems Detected* dialog, click on the *Continue* button. Finally, if the *Find Refactoring Preview* panel appears, click on the *Do Refactor* button to relocate the class.

On completion of these steps, the project tree should match that shown in Figure 77-5 below:

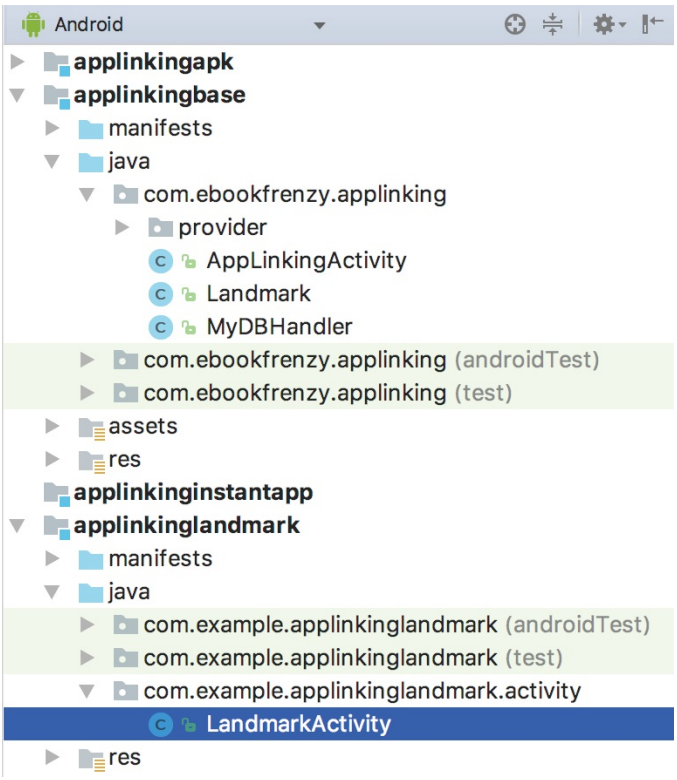

Figure 77-5

The final task to complete the relocation of the activity from the base module to the landmark module is to move the references from the base manifest file to the landmark module manifest.

First, edit the *applinkingbase -> manifests -> AndroidManifest.xml* file and copy the following section of the file:

```
<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
```

Open the *applinkinglandmark -> manifests -> AndroidManifest.xml* file and place the above content into the file so that it reads as follows:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.applinkinglandmark">

        <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
```

```
            android:label="@string/app_name"
            android:roundIcon="@mipmap/ic_launcher_round"
            android:supportsRtl="true"
            android:theme="@style/AppTheme">

        </application>

</manifest>
```

Next, cut the following LandmarkActivity element so that it is removed from the *applinkingbase* manifest file and paste it within the application element of the a*pplinkinglandmark* manifest file:

```
<activity android:name=
"com.ebookfrenzy.applinking.com.example.applinkinglandmark.activity.La

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />

        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data
            android:scheme="http"
            android:host="www.example.com"
            android:pathPrefix="/landmarks" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />

        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

            <data
                android:scheme="https"
                android:host="www.example.com"
                android:pathPrefix="/landmarks" />
        </intent-filter>
</activity>
```

Remaining within the a*pplinkinglandmark* manifest file, edit the activity name declaration so that it refers to the new location:

```
<activity android:name=
    "com.example.applinkinglandmark.activity.LandmarkActivity">
```

Select the *Build -> Rebuild Project* menu and wait for the build to complete. The compilation may fail with the following error:

```
Error:(41, 50) error: cannot find symbol class LandmarkActivity
```

If this error occurs, edit the *AppLinkingActivity.java* file and locate the *findLandmark()* method. If the method reads as follows then an additional change is required:

```
public void findLandmark(View view) {

    if (!idText.getText().equals("")) {
        Landmark landmark =
              dbHandler.findLandmark(idText.getText().toString());

        if (landmark != null) {
            Intent intent = new Intent(this, LandmarkActivity.class);
            String landmarkid = idText.getText().toString();
            intent.putExtra(LANDMARK_ID, landmarkid);
            startActivity(intent);
        } else {
            titleText.setText("No Match");
        }
    }
}
```

The problem here is that the code is specifically launching the LandmarkActivity which now resides in a different feature module. In fact, LandmarkActivity is now contained in an entirely separate instant app APK file. Within the AppLinking project folder, the path to the base feature module APK file containing the AppLinkingActivity class is as follows:

applinking-base/build/outputs/apk/feature/debug/applinking-base-debug.apk

The APK file path for the applinkinglandmark feature module containing the LandmarkActivity class, on the other hand, is as follows:

applinkinglandmark/build/outputs/apk/feature/debug/applinkinglandmark-debug.apk

To resolve this issue, the AppLinkingActivity class will need to launch the LandmarkActivity using an App Link as follows, relying on the Instant Apps system to install and launch the landmark instant app APK:

```
.
.
import android.net.Uri;
```

```
.
.
public void findLandmark(View view) {

    if (!idText.getText().equals("")) {
        Landmark landmark =
                  dbHandler.findLandmark(idText.getText().toString());

        if (landmark != null) {
            Uri uri = Uri.parse("http://example.com/landmarks/" +
                                landmark.getID());
            Intent intent = new Intent(Intent.ACTION_VIEW, uri);
            startActivity(intent);
        } else {
            titleText.setText("No Match");
        }
    }
}
```

After making this change, rebuild the project and check that the project now builds without error.

77.4 Testing the Multi-Feature Project

Edit the run configuration for the applinkinginstantapp module and set the launch URL to the following:

http://www.example.com/landmarks/londonbridge

When the instant app module is launched, the landmark module should load, launch and display the correct landmark information. Repeat this step, this time with the URL set to the following:

http://www.example.com/home

This time, the APK for the base feature module will install and launch, displaying the AppLinkingActivity screen.

Note that entering a landmark ID and clicking on the *Find* button opens the Chrome browser and loads the http://www.example.com web page instead of launching LandmarkActivity. Once the App Links used by the project have been associated with a web site and the app uploaded to the Google Play console, clicking the *Find* button will install and launch the LandmarkActivity correctly.

77.5 Summary

This chapter has outlined the creation of a multi-feature app project by making modifications to an existing app. The process involved the creation of a new feature module, changes to the build and manifest files and the relocation of an activity from the base feature module to the new module. The chapter also demonstrated the problems that arise when an attempt is made to launch an activity that resides in a different feature module using a standard intent. This problem was resolved by launching the activity using the app link URL.