Working with AndroidManifest

Every Android application has an `AndroidManifest.xml` file in the root directory. This manifest file contains the information which is required for running the application. For more information, see http://developer.android.com/guide/topics/manifest/manifest-intro.html#.

Renaming an Android application package (application ID)

You may need to have your application built in several versions which means that several Android application packages (`.apk` files) will be generated. If these files have this single name, the user will be unable to deploy them on the same device simultaneously. To overcome this obstacle, you can have IntelliJ IDEA generate several `.apk` files with different names (*application IDs*) from the same source code.

The name of the application package to be generated (*application ID*) is specified in the `package` attribute of the `manifest` element (see http://developer.android.com/guide/topics/manifest/manifest-element.html#package). This name follows the Java naming style and by default is the same as the package to which the class implemented for the application belongs.

The name of the application (the `android:name` attribute of the `application` element, see http://developer.android.com/guide/topics/manifest/application-element.html#nm), and the names of activities (the `android:name` attribute of the `activity` element, see http://developer.android.com/guide/topics/manifest/activity-element.html#nm) are by default specified *relative* to the *application ID* and, accordingly, to the parent Java package of the application implementation class. However, renaming the *application ID* does not cause renaming the parent Java package of the application class.

You can either change the *application ID* through the *Rename* refactoring or automatically during build.

Changing an application id through the rename refactoring

1. Open the `AndroidManifest.xml` file.
2. Position the cursor at the `package` attribute of the `manifest` element and choose **Refactor | Rename** on the context menu.
3. In the **Rename** dialog box that opens, specify the new package name and click **OK**.

The value of the `package` attribute changes to the newly specified. However, because this does not cause renaming the parent Java package of the application class, the relative names of the application and activities are replaced with their fully qualified names.

Renaming an application ID during build

1. Open the **Project Structure** dialog box by choosing **File | Project Structure** . See Accessing Project Structure and Accessing Module Settings for more information.
2. In the left-hand pane, choose **Modules**, then in the central pane click the **Android** facet under the relevant module.
3. In the right-hand pane, switch to the **Compiler** tab, select the **Rename manifest package** check box, and specify the new *application ID*.

Merging Manifest Files

You can have manifest files of library modules automatically merged with the manifest of the project that contains this library module.

The main goal of declaring a module as *library module* and attaching it to other projects is re-using the components from the library module. To be successfully integrated into another application and function inside it, these components must be presented in the application `AndroidManifest.xml` file. You can either add this information manually or have it extracted from `AndroidManifest.xml` of the library module and added to the `AndroidManifest.xml` of the application automatically. The second approach is referred to as *merging manifests*.

Find more about library modules on Sharing Android Source Code and Resources Using Library Projects.

To have the manifest files of library modules merged with the manifest of the containing project automatically

1. Open the **Project Structure** dialog box by choosing **File | Project Structure** . See Accessing Project Structure and Accessing Module Settings for more information.
2. In the left-hand pane, choose **Modules**, then in the central pane click the **Android** facet under the application module with the main launcher activity.
3. In the right-hand pane, that shows the Android Facet page, switch to the **Compiler** tab and select the **Enable manifest merging** check box.

See Also

Procedures:

- Sharing Android Source Code and Resources Using Library Projects
- Generating Signed and Unsigned Android Application Packages

Web Resources:

- Developer Community