Enabling Android Support

On this page:

- Preparing for Android development
- Choosing the module type you need
- Creating a project with an Android module from scratch
- Adding an Android module to a project
- Attaching an Android facet to an existing module
- Configuring the code style of Android-specific XML definition files

Preparing for Android development

- Before starting Android development, download and extract the Android SDK.

  - Android SDK is not a substitute for a Java SDK (JDK). You need to download and configure a Java SDK for your project anyway.
  - It is strongly recommended that the path to the Android SDK home directory should not contain spaces.

- Configure the Android SDK in IntelliJ IDEA, see Configuring Global, Project and Module SDKs.
- Make sure, the *Android Support* bundled plugin is active. This plugin provides Android support at the IntelliJ IDEA level and is by default enabled. If not, enable the plugin in the Plugin Configuration Wizard or in the Plugins page of the Settings dialog box.
- Depending on your task, decide on the module type you need for Android development.
- Create a project with an Android module from scratch, or add an Android module to a project, or attach an Android facet to an existing module. Create a module of the type chosen in the previous step.

  In any case, IntelliJ IDEA automatically sets up the correct module structure with `res` and `gen` folders, downloads the necessary libraries, and generates various Android-specific descriptors.

Choosing the module type you need

IntelliJ IDEA distinguishes among the following types of Android modules: *Application Module*, *Empty Module*, *Library Module*, and *Test Module*. You choose the type of the module for Android development either when creating the module from scratch or when adding a module to an existing project.

An *Application Module* is a module that is ready for developing an Android application. IntelliJ IDEA will create an application package and set up the Android-specific module structure:

1. The `src` folder for the sources that implement the application behaviour.
2. The `res` folder with dedicated subfolders to store definitions of visual application resources: images, strings, layouts, etc.
3. The `gen` folder. The folder contains the application package `com.example.<application name>` with the following files:
   - `R.java` file that links the visual resources and the Java source code.
   - `Manifest.java` file stub for referencing custom permissions.
   - `BuildConfig.java` file where the mode of the Android Application package is defined through the `public final static boolean DEBUG` field. If the field value is set to `true`, the APK is generated in the debug mode, which means that the application will not be intended for publishing.

An *Empty Module* contains two empty folders `gen` and `src`.

A *Library Module* is intended for holding shared Android source code and resources. Other Android application projects can reference a library project and include its compiled sources in their `.apk` files at build time. When you choose the *Library Module* type, IntelliJ IDEA sets up almost the same module structure as for an *Application Module* but does not suggest generating a sample application. Another difference is that IntelliJ IDEA updates the Android facet of the module: the *Library project* check box is automatically selected in the module settings.

A *Test Module* is intended for creating and running Android unit tests. Modules of this type can be only added to an existing project.

Creating a project with an Android module from scratch

1. Do one of the following:
   - If no project is currently opened in IntelliJ IDEA, click **Create New Project** on the **Welcome** screen.
   - If you already have an opened project, choose **File | New Project** on the main menu.

   The New Project wizard opens.
2. In the left-hand pane, select **Android**. In the right-hand part of the page, choose the type of the module to create:
   - An **Application Module** is ready for developing an Android application.
   - An **Empty Module** contains two empty folders `gen` and `src`.
   - A **Library Module** is intended for holding shared Android source code and resources. Other Android application projects can reference a library project and include its compiled sources in their `.apk` files at build time.
3. For an *Application* module, on the next page of the wizard, specify:
   - The name of the application that will be implemented in the module and the package to store the application classes in.
   - Optionally, enable creation of a sample "Hello, World!" application and specify the title of the window to be displayed on the sample application start.
4. On the last page of the wizard, specify:
   - The name of the project, its location, and format.
   - The Android SDK to use in the project and in the module.
   - The name of the module, the location of the `.iml` module file, and the content root of the module.

   For an Application module, also specify the target device to run the application on. For a Library module, specify the package name.

Adding an Android module to a project

1. Choose **File | New Module** on the main menu or **New | Module** on the context menu of the **Project** tool window.

   The New Module wizard opens.

2. In the left-hand pane, select **Android**. In the right-hand part of the page, choose the type of the module to create:
   - An **Application Module** is ready for developing an Android application.
   - An **Empty Module** contains two empty folders `gen` and `src`.
   - A **Library Module** is intended for holding shared Android source code and resources. Other Android application projects can reference a library project and include its compiled sources in their `.apk` files at build time.
   - **Test Module** is intended for creating and running Android unit tests.
3. On the next page of the wizard, for an *Application* module, specify:
   - The name of the application that will be implemented in the module and the package to store the application classes in.
   - Optionally, enable creation of a sample "Hello, World!" application and specify the title of the window to be displayed on the sample application start.
4. On the last page of the wizard, specify the name of the module, the location of the `.iml` module file, and the content root of the module. For an Application module, also specify the target device to run the application on. For a Library module, specify the package name. For a Test module specify the target module to run unit tests against.

Attaching an Android facet to an existing module

1. Open the Project Structure dialog box.
2. Under **Project Settings**, select **Modules**.
3. Select the module you want to add an Android facet to, click +, and choose **Android**.
4. On the Facet 'Android' page that opens, specify the location of the key application components in the **Structure** tab: the AndroidManifest.xml file, the application resources, the application assets, and the Android native libraries.

   The controls in the tab shows default settings. You can edit them if necessary. To return to the default Android facet settings, click the **Reset paths to defaults** button.

5. To make the module source code and resources available from other projects, select the **Library project** check box.

Configuring the code style of Android-specific XML definition files

Android development involves working with dedicated XML files, such as layout and resource definition files, manifest files, etc. You can have IntelliJ IDEA apply the standard XML code style to such files or configure custom the code style settings for them.

1. Open the project settings by choosing **File | Settings**, click **Code Style**, then click **XML**.
2. On the **Code Style:XML** page that opens, switch to the **Android** tab.
3. Do one of the following:
   - To define a custom code style for Android-specific XML files, select the **Use custom formatting settings for Android XML files** check box and configure the settings to be applied to various types of Android XML files using the controls of the tab as described in Code Style:XML - Android.
   - To have IntelliJ IDEA format Android-specific XML files according to the standard XML code style settings defined in the other tabs of the page, clear the **Use custom formatting settings for Android XML files** check box.

See Also

Procedures:

- Testing Android Applications
- Sharing Android Source Code and Resources Using Library Projects
- Android

Reference:

- Android Page
- Android Facet Page
- Android Reference

Web Resources:

- Developer Community