Emerging Technologies: Mobile Development for Android Devices

Android Application Structure

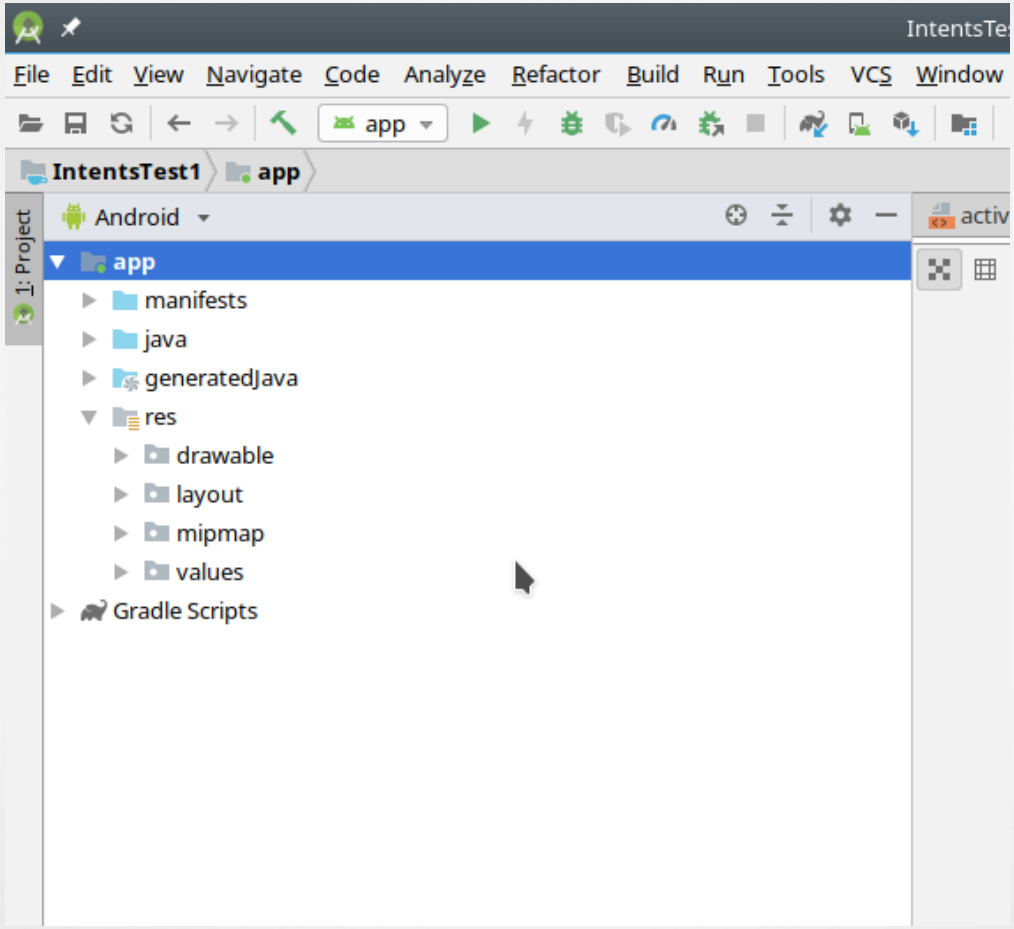
Introduction

- Android application structure from the developer’s point of view.
- Key directories, files and their contents
- Code and non-code resources
- Layouts
- Intents (Explicit intents. Implicit intents)
- The android manifest file (AndroidManifest.xml)
- Conclusion

Application Structure

- Android projects compile to produce an APK which is installed on an android device.
- However, during development, Android projects have a well organised directory structure to contain development assets.
- Development assets include code and non-code resources.
- Code resources are the java source and class files.
- Non-code resources is everything else.
- Non-code resources include, xml files, images, sound, data etc.
- Non-code resources include a lot of resources which are simply text files.
- Text files in particular compress well, leading to storage space efficiency.

Android Development Directories



Android Development Directories

- manifests: Contains the android manifest file.
- java: java code
- res: Non-code resource categories of:
  - drawable
  - layout
  - mipmap
  - values

ART & Dalvik

- Runtime environments for Android
- ART (Android Runtime) replaces the earlier Dalvik.
- Dalvik is essentially a bytecode interpreter.
- ART compiles application code into machine code at install time resulting in improved runtime performance.
- However, both environments treat non-code resources similarly.
- Just static files to be installed/removed from a single location.

The res Directory

- Contains non-code resources (res → resources)
- Subdirectories include layout, mipmap, values
- The layout directory contains xml user interface layouts.
- The mipmap directory contains subdirectories with images in at least five different DPI (resolution) levels.
- mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi
- Having ready images removes the need for expensive image resizing operations on the Android device.

Display Resolution

- DPI → Dots per Inch
- 160 DPI is used as Android reference.
- The higher the DPI number, the more detailed the display.
- Each device has a hard-coded index chosen by the manufacturer to indicate the DPI level that the device can support.
- mdpi (160), hdpi (240), xhdpi (320), xxhdpi (480),
- xxxhdpi (640)

Reducing Image Processing at Runtime

- The strategy of providing various image resolutions allows devices that have low screen resolutions to display images that are suitable. This is better than having to process high-resolution images so that they are displayable.
- Processing capability is likely to be limited on such devices anyway.
- Low-end devices are not bogged down by image processing and are more responsive.
- High-end devices that are capable of doing so, may display high resolution images for improved user interface ‘look and feel’.
- Avoids using low-resolution pixellated images on high-end devices.
- However, there is a cost in the form of a greater storage requirement.

The values Directory

- Contains files that define colours, strings and styles.
- Colours are defined using RGB with Hex using American spelling ‘color’:

```
<color name="colorAccent">#D81B60</color>
```

- A style file can be used to apply a uniform theme to an application, similar to the style sheet concept for html.
- Of particular note is the strings file ‘strings.xml’.
- Used to define all strings used in the application.
- The use of strings.xml to define strings is preferred to the hard-coding of strings.
- Hard-coding of strings makes localization of applications very difficult.

The layout Directory

- Stores layouts for all activities.
- Layout files are created using XML to take full advantage of text compression.
- It is possible to create layouts in java code. However, this is error prone, labour intensive and not as flexible as XML.
- The layout inflator service is fastest at building layouts that are created using XML.
- Layouts can consist of more than one file. For example, they can use included files.
- An activity’s layout files can be shared among other activities.

Application UI Structure

- Due to limited screen space, an activity is required to take up the full screen.
- The on-screen activity gets the largest available screen space while availing of the full processing capability of the CPU.
- However, if additional application functions need to be implemented, another screen is needed. (→ another activity)
- There is then a need for a mechanism to navigate from one activity to the other.
- This is the purpose of intents.

Intents

- Used to move from one activity to another.
- May have data attached to bring to a new activity or be returned from a previous activity.
- Can be implicit or explicit.
- Explicit Intent: Where the fully qualified name of the activity is known. Often an activity within your own application.
- Can be from an external application where the explicit, i.e. fully qualified name of the activity is known.

Implicit Intents

- An implicit intent arises where you have data that you need to display or interact with but do not know how to do this.
- Android will try to find an external application that can display or interact with the data.
- It is possible that such an application may not be available.
- The strategy is that applications do a single job and do it well.
- If another task is required to be done, try to hand off to an application that already implements this.

Implicit Intents

- This strategy results in less duplication and efficient use of storage space.
- Applications are small, bug free and integrate with each other.
- If some common functionality is required, use an existing application for this.
- Improved reliability and smaller coding requirement for new applications.

The Manifest file

- A key file in the application.
- Provides information to the device about the application being installed.
- Name, icon, package, version, api.
- Permissions that the application needs, eg. location data. These are displayed to and authorised by the user. Can also be allowed/disallowed later. (Security)
- For compatability reasons, minimum api.
- An earlier device might not be capable of running the application.

The Manifest file

- A further purpose of the manifest file is to list all of the activities, services and components that make up the application.
- All components must be declared.
- An attempt by the application to start something that has not been declared results in immediate shutdown of the application.

Conclusion

- Development projects include code and non-code resources.
- Try to weight development assets towards non-code resources to take full advantage of compression, localization and android specific efficiencies, eg. fast, efficient rendering of XML layout files.
- Utilize strings.xml rather than hard-coded strings.
- Be aware of the key role of the manifest file and the strict treatment of applications in relation to manifest declarations.