The Anatomy of an Android Application

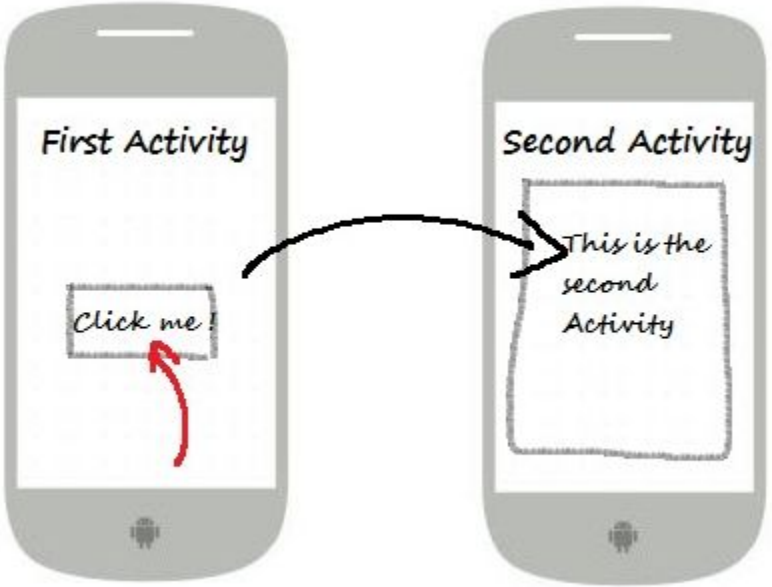
Android Activities

- Since Android applications are written in Java and Kotlin,
  - Encapsulates elements of application functionality into classes that are then instantiated as objects and manipulated to create an application.
- Also takes the concept of re-usable components to a higher level.

Android Activities

- Android applications are created by bringing together one or more components known as ***Activities***.
- An *activity* is a single, standalone module of application functionality that usually correlates directly to a single user interface screen and its corresponding functionality.

Android Activities



Android Activities

- Eg: An appointments application
  - Have an activity screen that displays appointments set up for the current day.
  - Second activity consisting of a screen where new appointments may be entered by the user.
- Activities are intended as fully reusable and interchangeable building blocks that can be shared amongst different applications.

Android Activities

- Eg: An existing email application.
  - Contain an activity specifically for composing and sending an email message.
  - A developer might be writing an application that also has a requirement to send an email message.
  - Rather than develop an email composition activity specifically for the new application, the developer can simply use the activity from the existing email application.

Android Activities

- Activities are created as subclasses of the Android *Activity* class
- Must be implemented so as to be entirely independent of other activities in the application.
- A shared activity cannot rely on being called at a known point in a program flow (since other applications may make use of the activity in unanticipated ways).
- One activity cannot directly call methods or access instance data of another activity.
- This, instead, is achieved using *Intents and Content Providers*.

Android Activities

- By default, an activity cannot return results to the activity from which it was invoked.
- If this functionality is required, the activity must be specifically started as a sub-activity of the originating activity.

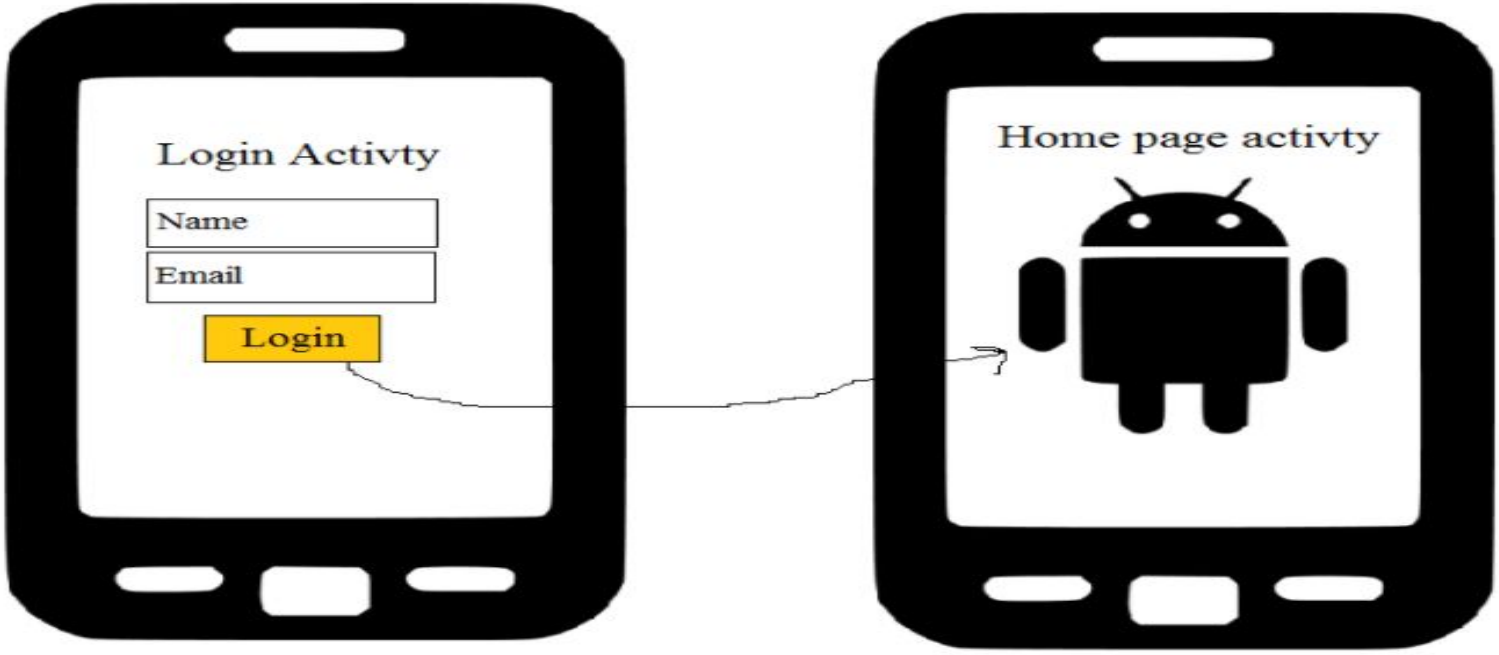
Android Intents

- Mechanism by which one activity is able to launch another and implement the flow through the activities that make up an application.
- Intents consist of a description of the operation to be performed and, optionally, the data on which it is to be performed.

Android Intents

- Intents can be *explicit*
  - They request the launch of a specific activity by referencing the activity by class name.
- *implicit*
  - Stating either the type of action to be performed or providing data of a specific type on which the action is to be performed.
  - Android runtime will select the activity to launch that most closely matches the criteria specified by the Intent using a process referred to as *Intent Resolution*.

Android Intents



Explicit intent

Broadcast Intents

- System wide intent that is sent out to all applications that have registered an “interested” *Broadcast Receiver*.
- The Android system send out Broadcast Intents to indicate
  - changes in device status such as the completion of system start up
  - connection of an external power source to the device
  - screen being turned on or off

Broadcast Intents

- A Broadcast Intent can be *normal* (asynchronous)
  - It is sent to all interested Broadcast Receivers at more or less the same time
- *ordered*
  - It is sent to one receiver at a time where it can be processed and then either aborted or allowed to be passed to the next Broadcast Receiver.

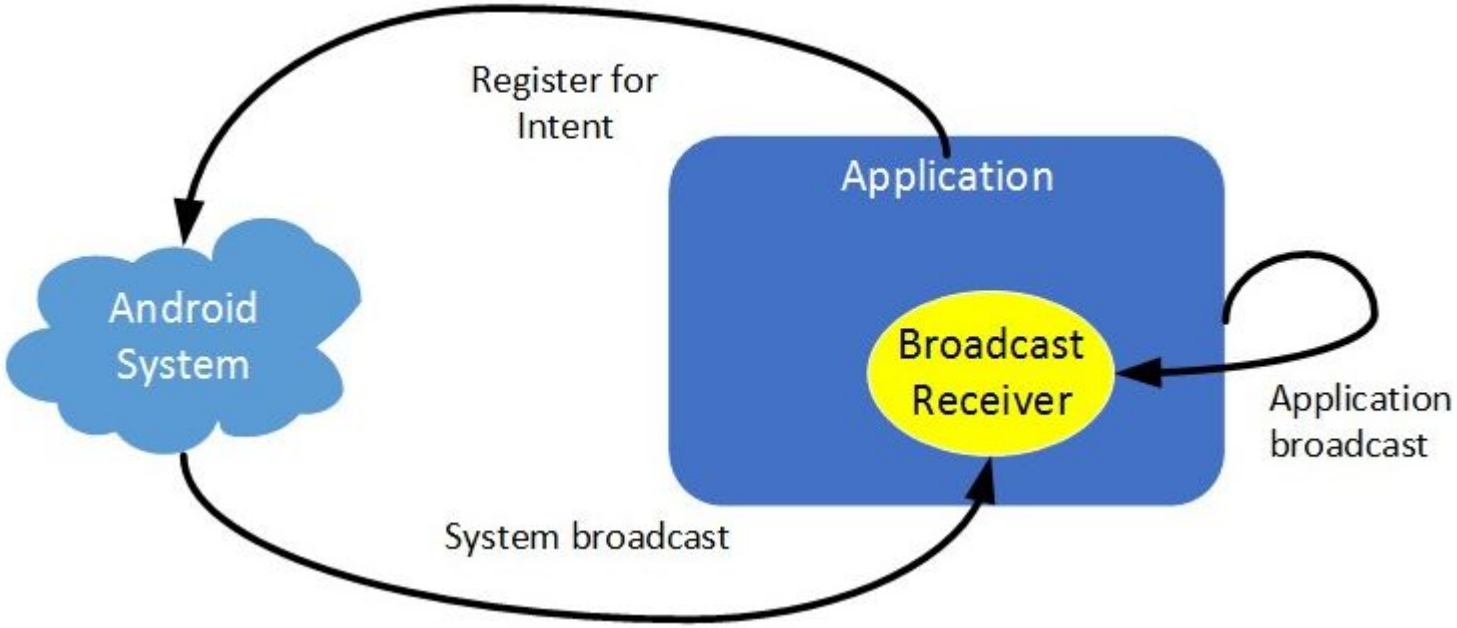
Broadcast Receivers

- Mechanism by which applications are able to respond to Broadcast Intents.
- A Broadcast Receiver must be registered by an application and configured with an *Intent Filter* to indicate the types of broadcast in which it is interested.
- When a matching intent is broadcast, the receiver will be invoked by the Android runtime regardless of whether the application that registered the receiver is currently running.

Broadcast Receivers

- The receiver then has 5 seconds in which to complete any tasks required of it (such as launching a Service, making data updates or issuing a notification to the user) before returning.
- Broadcast Receivers operate in the background and do not have a user interface.

Broadcast Receivers

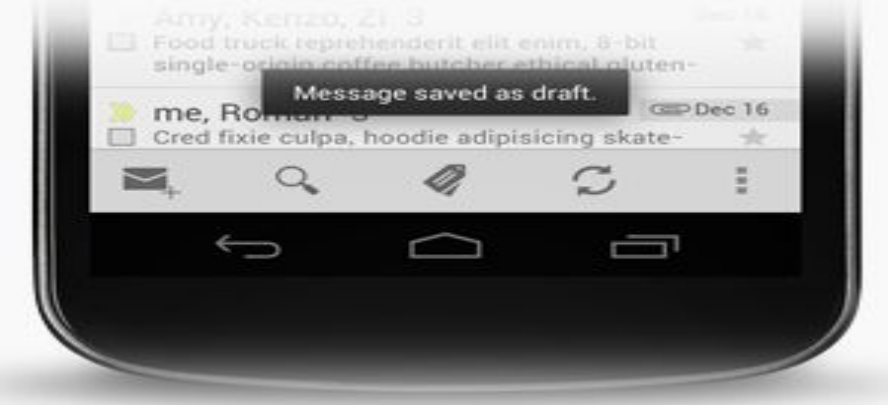


Android Services

- Processes that run in the background and do not have a user interface.
- Can be started and subsequently managed from activities, Broadcast Receivers or other Services.
- Ideal for situations where an application needs to continue performing tasks but does not necessarily need a user interface to be visible to the user.

Android Services

- Can still notify the user of events using notifications and *toasts* (small notification messages that appear on the screen without interrupting the currently visible activity)
- Also able to issue Intents.



Android Services

- Services are given a higher priority by the Android runtime than many other processes.
- Will only be terminated as a last resort by the system in order to free up resources.
  - Automatically restarted as soon as adequate resources once again become available.
- A Service can reduce the risk of termination by declaring itself as needing to run in the *foreground*.
  - Achieved by making a call to *startForeground()*
  - Streaming of audio that should continue when the application is no longer active

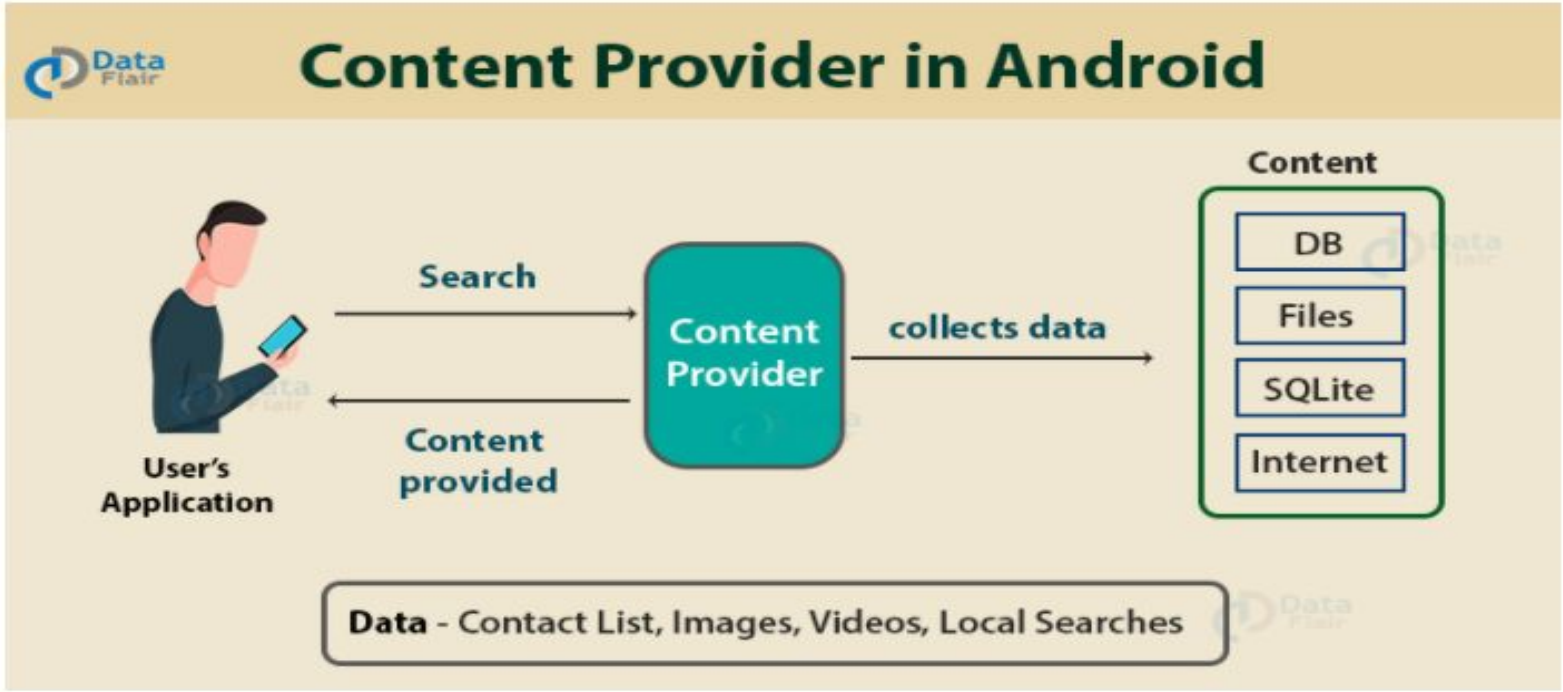
Content Providers

- Implement a mechanism for the sharing of data between applications.
- Any application can provide other applications with access to its underlying data
  - Including the ability to add, remove and query the data (subject to permissions).
- Access to the data is provided via a Universal Resource Identifier (URI) defined by the Content Provider.
- Data can be shared in the form of a file or an entire SQLite database.

Content Providers

- The native Android applications include a number of standard Content Providers allowing applications to access data such as contacts and media files.
- The Content Providers currently available on an Android system may be located using a *Content Resolver.*

Content Providers



The Application Manifest

- The glue that pulls together the various elements that comprise an application is the Application Manifest file.
- It is within this XML based file that the application outlines the activities, services, broadcast receivers, data providers and permissions that make up the complete application.
- AndroidManifest.xml
- Describes essential information about your app
- Describes the components of app and their properties

Application Resources

- In addition to the manifest file and the Dex files that contain the byte code, an Android application package will also typically contain a collection of *resource files*.
- These files contain resources such as the strings, images, fonts and colors that appear in the user interface together with the XML representation of the user interface layouts.
- By default, these files are stored in the */res* sub-directory of the application project's hierarchy.

Application Context

- When an application is compiled, a class named R is created that contains references to the application resources. (R.java)
- The application manifest file and these resources combine to create what is known as the *Application Context.*
- This context, represented by the Android Context class, may be used in the application code to gain access to the application resources at runtime.
- A wide range of methods may be called on an application's context to gather information and make changes to the application's environment at runtime.

Study the terms

- Uses-permission :System permission that the user must grant for the app to operate correctly.
  - Camera, internet
- Uses-sdk: Filter your app from devices that don't meet its platform version requirements.
  - App’s compatibility with version of android platform
- How to create a project
- Create a device (Tools -> Device Manager)
- Locate Layout and Drawable resources

Thank You