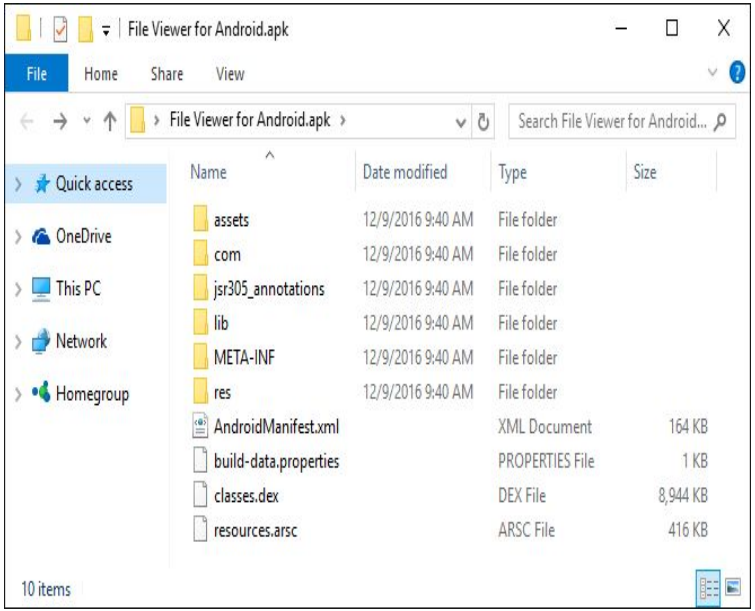
Android Programming

LECTURE 1

Application Fundamentals

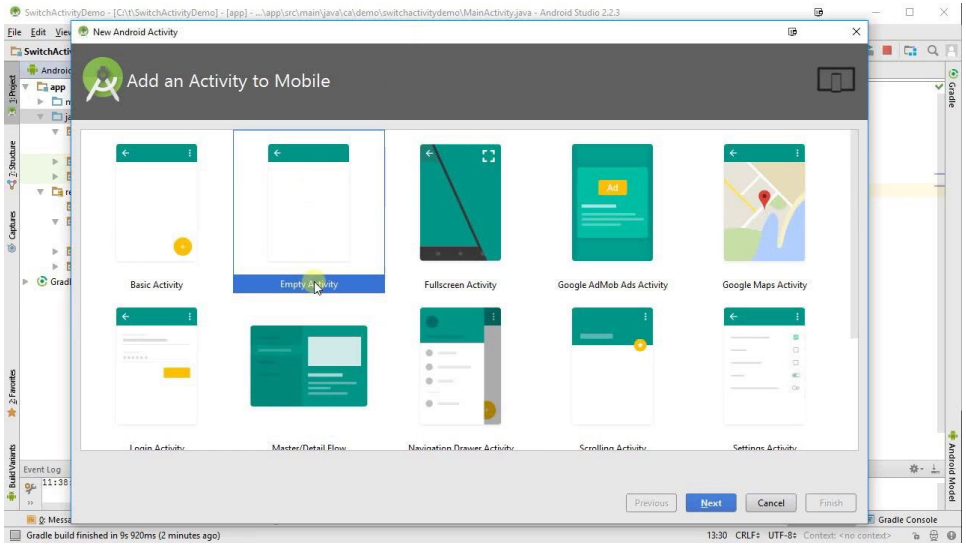
- Android apps can be written using **Kotlin, Java, and C/C++** languages.
- The Android SDK (Software Development Kit) tools compile your code along with any data and resource files into an APK, an *Android package*, **which is an archive file with an `.apk` suffix.**
- Android Studio - Android's IDE (Integrated Development Environment). Contains tools for writing and debugging code, creating UIs for apps, and interfacing with the Android SDK.
- One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.

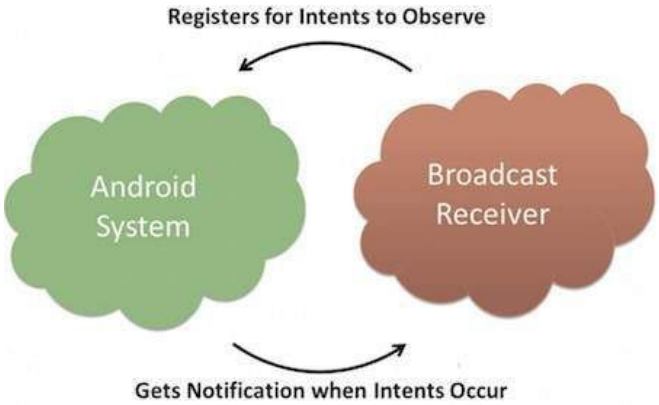


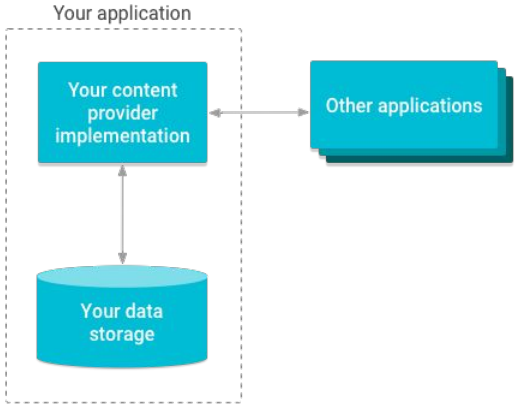
App Components

There are four different types of app components:

- **Activities**
- **Services**
- **Broadcast receivers**
- **Content providers**







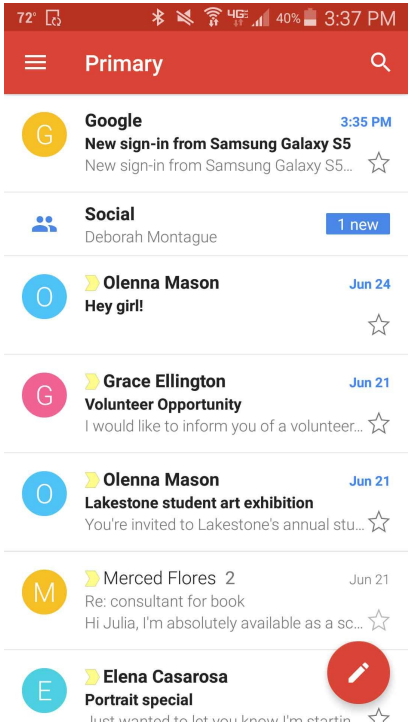
Activities

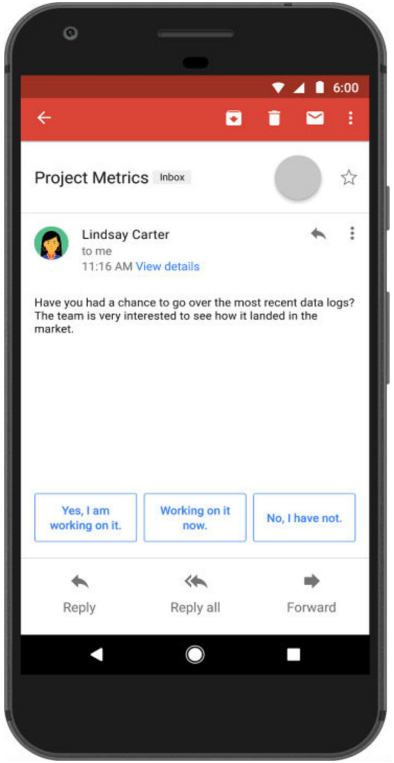
- The entry point for interacting with the user

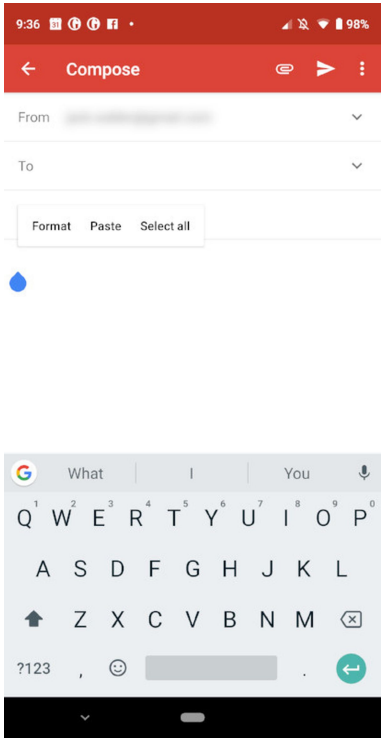
- Represents a single screen with a user interface. Eg. an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails

- The activities work together to form a cohesive user experience in the email app, each one is independent of the others.

Different Activities in gmail App

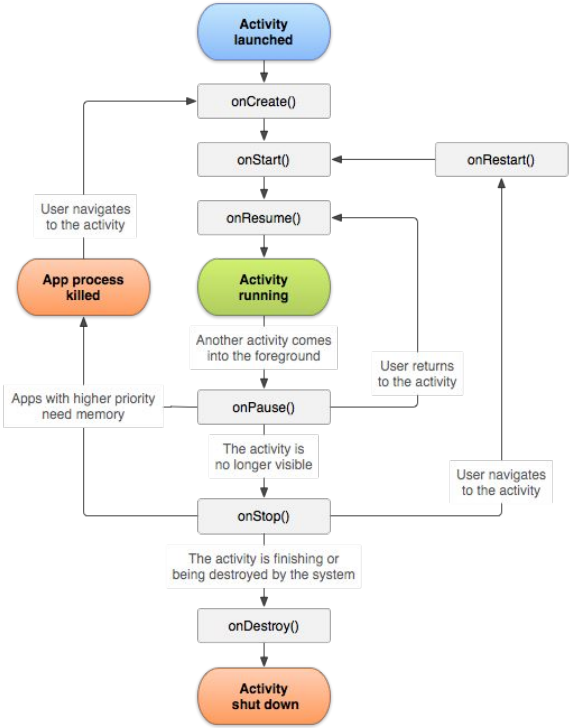






Understand the Activity Lifecycle

As a user navigates through, out of, and back to your app, the Activity instances in your app transition through different states in their lifecycle



Key Interactions between the Android OS and each Activity

- Keeping track of what the user currently cares about (what is on screen) to ensure that the system keeps running the process that is hosting the activity.

- Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.

- Helping the app handle having its process killed so the user can return to activities with their previous state restored.

- Providing a way for apps to implement user flows between each other, and for the system to coordinate these flows. (The most classic example here being share.)

Services

- **A service is a general-purpose entry point for keeping an app running in the background for all kinds of reasons.**

  For Eg: **Live wallpapers, notification listeners, screensavers**

- It is a component that runs in the background to perform long-running operations or to perform work for remote processes.

- A service does not provide a user interface.

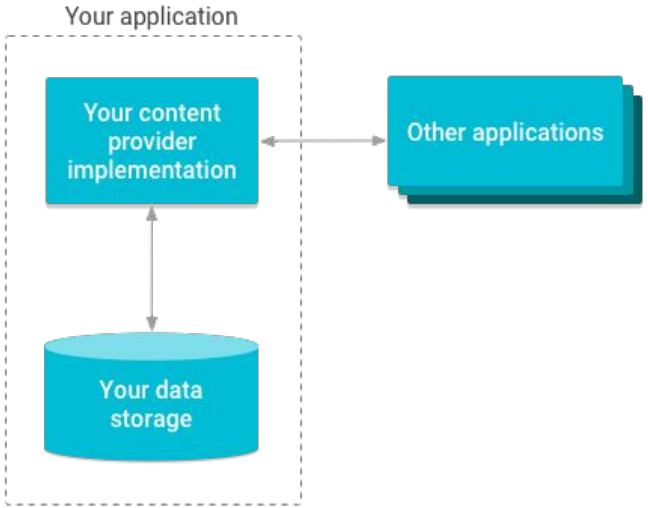
For example, **a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity**

Content providers

- A *content provider* manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access.

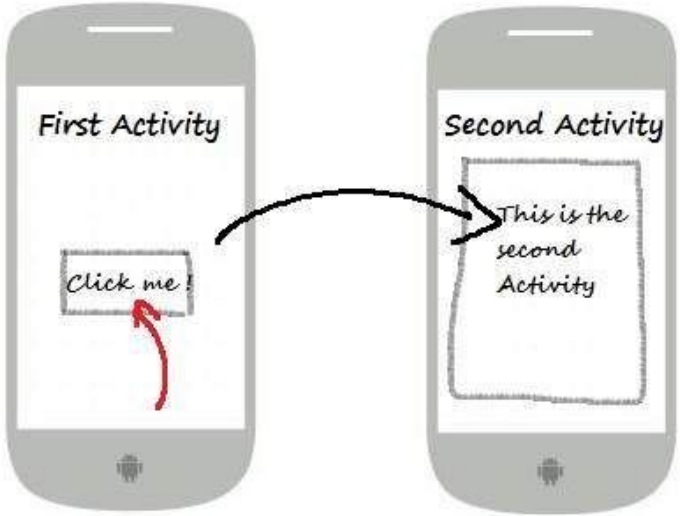
- For example, the Android system provides a content provider that manages the user's contact information. As such, any app with the proper permissions can query the content provider, such as `ContactsContract.Data`, to read and write information about a particular person.

- Content providers are also useful for reading and writing data that is private to your app and not shared.



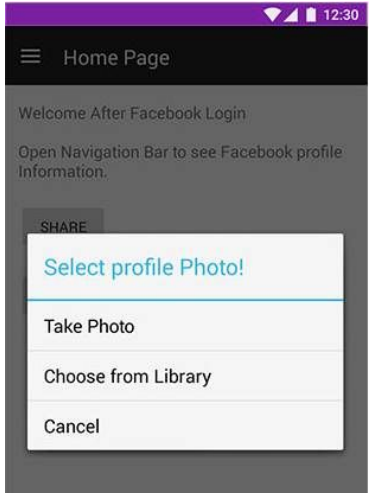
Intents

- Three of the four component types—activities, services, and broadcast receivers—are activated by an asynchronous message called an *intent*.
- Intents bind individual components to each other at runtime.
- Think of them as the messengers that request an action from other components, whether the component belongs to your app or another.



Intents

- For activities and services, an intent defines the action to perform (for example, to *view* or *send* something) For example, an intent might convey a request for an activity to show an image or to open a web page.



- You can start an activity to receive a result, in which case the activity also returns the result in an Intent. For example you can issue an intent to let the user pick a personal contact and have it returned to you.

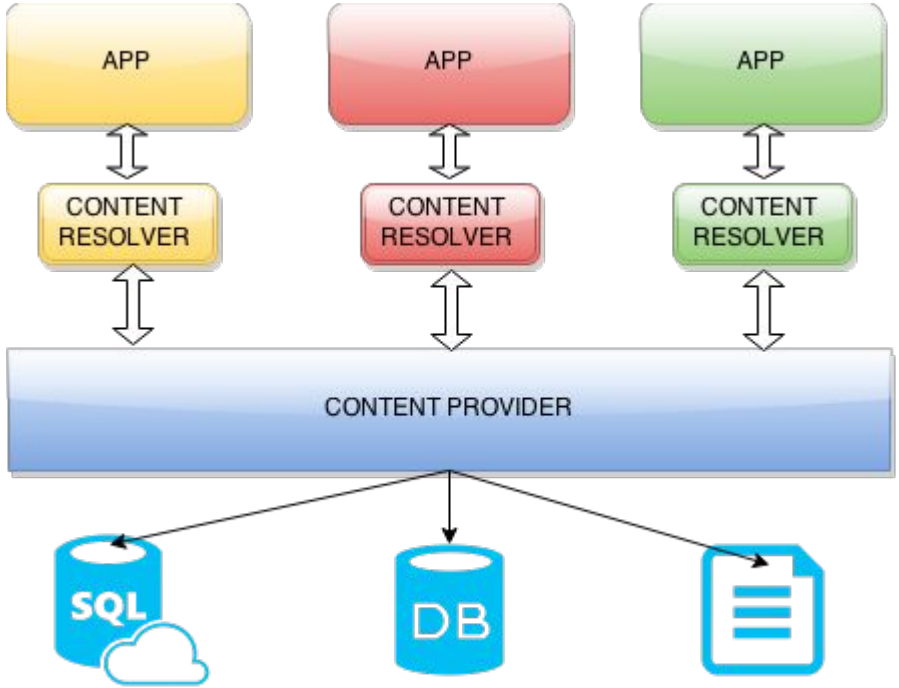
- For broadcast receivers, the intent simply defines the announcement being broadcast. For example, a broadcast to indicate the device battery is low includes only a known action string that indicates *battery is low*.

Content Resolvers

- Unlike activities, services, and broadcast receivers, content providers are not activated by intents. They are activated when targeted by a request from a Content Resolvers.

- The content resolver handles all direct transactions with the content provider so that the component that's performing transactions with the provider doesn't need to and instead calls methods on the Content Resolver object.

- This leaves a layer of abstraction between the content provider and the component requesting information (for security).

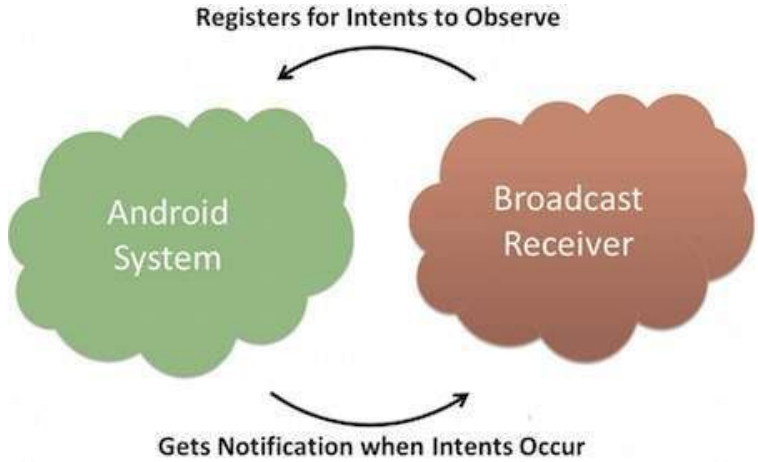


Broadcast Receivers

- Broadcasts are sent when an event of interest occurs. For example, when the system boots up or the device starts charging.

- Apps can register to receive specific broadcasts

- Apps can receive broadcasts with broadcast receivers



Manifest File

- Before the Android system can start an app component, the system must know that the component exists by reading the app's *manifest file*, AndroidManifest.xml.
- App must declare all its components in this file, which must be at the root of the app project directory.
- Identifies any user permissions the app requires, such as Internet access or read-access to the user's contacts.
- Declares the minimum API Level required by the app, based on which APIs the app uses.
- Declares hardware and software features used or required by the app, such as a camera, bluetooth services, or a multitouch screen.
- Declares API libraries the app needs to be linked against (other than the Android framework APIs), such as the Google Maps library.

Sample Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="payspace.ssidit.pp.ua.payspacemagazine">

    <uses-permission android:title="android.permission.INTERNET" />
    <uses-permission android:title="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.Holo.Light">
        <activity
            android:label="@string/app_name"
            android:title=".MainActivity">
            <intent-filter>
                <action android:title="android.intent.action.MAIN" />

                <category android:title="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:label="@string/title_activity_settings"
            android:title=".SettingsActivity"></activity>
    </application>

</manifest>
```

App Resources

An Android app is composed of more than just code—it requires resources that are separate from the source code, such as images, audio files, and anything relating to the visual presentation of the app.

Types of Resources

The following are the most common types of resources within Android apps:

Name	Folder	Description
Property Animations	animator	XML files that define property animations.
Tween Animations	anim	XML files that define tween animations.
Drawables	drawable	Bitmap files or XML files that act as graphics
Layout	layout	XML files that define a user interface layout
Menu	menu	XML files that define menus or action bar items
Values	values	XML files with values such as strings, integers, and colors.

In addition, note the following key files stored within the `values` folder mentioned above:

Name	File	Description
Colors	`res/values/colors.xml`	For color definitions such as text color
Dimensions	`res/values/dimens.xml`	For dimension values such as padding
Strings	`res/values/strings.xml`	For string values such as the text for a title
Styles	`res/values/styles.xml`	For style values such as color of the AppBar

Handling of Resources

- For every resource that you include in your Android project, the SDK build tools define a unique integer ID, which you can use to reference the resource from your app code or from other resources defined in XML.

For example, if your app contains an image file named `logo.png` (saved in the `res/drawable/` directory), the SDK tools generate a resource ID named `R.drawable.logo`.

- This ID maps to an app-specific integer, which you can use to reference the image and insert it in your user interface.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello!</string>
    <string name="submit_label">Submit</string>
</resources>
```

```xml
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/submit_label" />
```

```
String submitText = getResources().getString(R.string.submit_label)
```

Qualifiers for Resources

- The qualifier is a short string that you include in the name of your resource directories in order to define the device configuration for which those resources should be used

- One of the most important aspects of providing resources separate from your source code is the ability to provide alternative resources for different device configurations. For example, by defining UI strings in XML, you can translate the strings into other languages and save those strings in separate files.

- Then Android applies the appropriate language strings to your UI based on a language *qualifier* that you append to the resource directory's name (such as `res/values-fr/` for French string values) and the user's language setting.

- Android supports many different *qualifiers* for your alternative resources.

Different Qualifiers depending upon the screen resolution

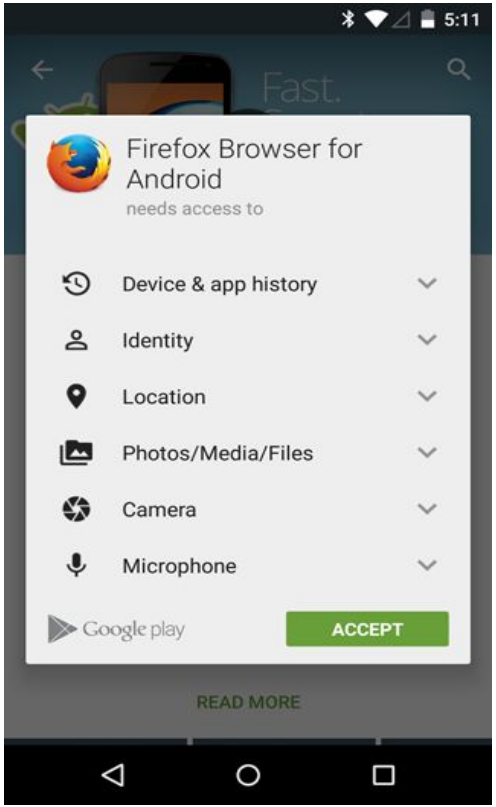
Permissions

- Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet).

- Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.

- Design point of the Android security architecture is that no app, by default, has permission to perform any operations that would adversely impact other apps, the operating system, or the user.

Firefox and messenger android app asking for permissions



Debugging

- Breakpoints
- Logging
  - To write log messages in your code, use the Log class. Log messages help you understand the execution flow by collecting the system debug output while you interact with your app.
  - You can view and filter debug and other system messages in the **Logcat** window