Main Components of Android Application

Android Core Building Blocks

- Make it easy to break them down into conceptual units
- Those units are independent
- When you find some errors or bugs if we have divide the application into those units it is easy for us to findout those errors
- After implementing all the conceptual units, you can get them together and build a one application

Activities

- A single screen that the user sees on the device at one time
- Most visible part of your application
- Launching single activity in Android involves:
  - Creating new Linux process.
  - Allocating memory for al the UI objects.
  - Retrieving the XML Layouts.
  - Setting up the whole screen.

How to create an activity?

- When activity created it automatically create an onCreate method
- Below statement used to load the xml file of activity
  - `“setContentView (R.layout.activity_main);”`
  - `This should retrieve particular file id of xml file`
- onCreate method is to create the activity and when launching the single activity first we need to create the particular xml file

Activity in Manifest file

- when we create the activity it automatically builds the manifest file
- when there’s only one activity it there with naming the activity file and mentioning that it is the launcher activity

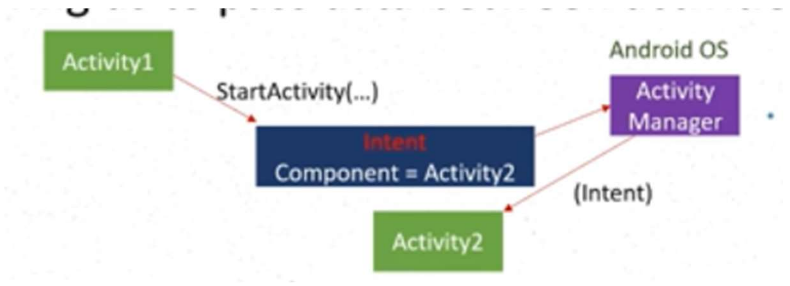
```
<activity android:name=".MainActivity"> //name of the activity
   <intent-filter>// mentioning that it is the launcher activity
      <action android:name="android.intent.action.MAIN" />

      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
```

Activity Manager

- In order to manage activities and all the processes there's an activity manager
- Responsible for creating, destroying and managing activities.
- And also it can manage the speed of our application
- Activity managers coordinates the activities (which activity should show first)
- Activity manager should hold the previous activity and call the next activity
- Stores the previous activities in a list
- Activity manager can decide whether an activity should be there or destroy if it not in use
- This mechanism is designed to help improve the speed of the user interface and thus improve the overall user experience

Intents

- Intent means a message that pass through the activities
- To pass messages and hide the activities in navigations we can use intents



```
Intent intent  = new Intent(MainActivity.this ,activity2.class);
startActivity(intent);
```

- Note: Intent class has a constructor that has two parameters.

  1 – reference to the current activity (this)

  2 – name of the activity we want to open

```
Button btn;//declare variable
btn = findViewById(R.id.button); //pass the id of UI element of xml file


//methods
protected void onResume() {
   super.onResume();
   btn.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
         //text.setText("I am salitha");
         Intent intent  = new Intent(MainActivity.this ,activity2.class);
         startActivity(intent);
      }
   });
}
```

- We need to pass values
- Those extra values are intent extras

```
String myExtra = "salitha";
protected void onResume() {
    super.onResume();
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //text.setText("I am salitha");
            Intent intent  = new Intent(MainActivity.this ,
                    activity2.class);
            intent.putExtra("MIN EXTRA", myExtra);
            startActivity(intent);
        }
    });
}
```

- In order to accept the String value, we have to do some change in activity2

```
String extraText;
TextView txt;
eeeeeeeeeeeeeee
txt = findViewById(R.id.textView);

Intent intent = getIntent();

extraText = intent.getStringExtra("MIN EXTRA");
protected void onResume() {
    super.onResume();
    txt.setText(extraText);
}
```

Intents are divided into 2 types

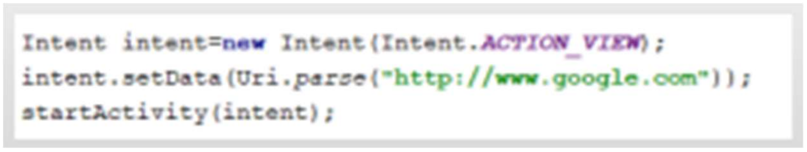
- Explicit Intents
  - This specify the current activity to be invoked by the component

    ```
    (this, MainMenu.class);
    ```

  - As 2nd parameter we specify the exact activity the user need to navigate
  - And we can pass extra values to one activity to other

- Implicit Intents

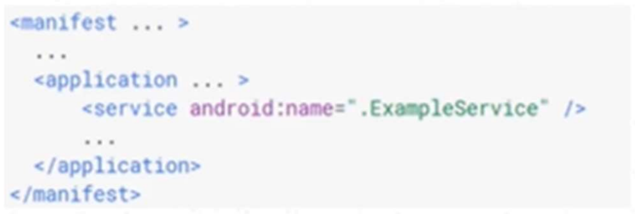
    - This doesn't specify the component and it has only one parameter that specify the type of the receiver
    - If we need to navigate for a website



```
Intent intent=new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.google.com"));
startActivity(intent);
```

Android services

- Services run in the background and don't have any user interface components.
- Those services don't have any user interfaces
- They can perform the same actions as activities but without any user interfaces
    - Playing a music (services) and flipping in other applications
- Bluetooth, wifi
- If the application makes brief sign of services, we have to mention in manifest
- We can take another java file to write the services and that activity name we have to mention in manifest



```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

Android service life cycle (No need to complete cycle in this stage)

- Two form of a service
    - Started
        - Tells the system about something it wants to be doing in the background
    - Bound
        - • Application can be exposed some of its functionality to other applications.

Broadcast receivers

- Those are registered for system announcements and application events
- Example: the signal that getting battery is charging when we plug into the charger
- They inform what happen inside the device

Content Providers

- In a smart phone they cannot directly communicate with each other
- In order to overcome this there are content providers
- They have billion number of databases to store, share, delete data

- There’s a content provider to communicate with those databases and our interfaces
- Example: we need contact details for WhatsApp and in order to get contact there’s a content provider to give those details
- Can be used to manage both structured data (Eg: SQLite relational databases) and unstructured data (Eg: image files)

In build functions

Overridden methods for Content Providers

- onCreate() - Called when the provider is started.
- query() - Receives a request from a client. The result is returned as a Cursor object.
- insert() - Inserts a new record into the content provider.
- delete() - Deletes an existing record from the content provider.
- update() - Updates an existing record from the content provider.
- getType() - Returns the MIME type of the data at the given URI.