AndroidManifest.xml file

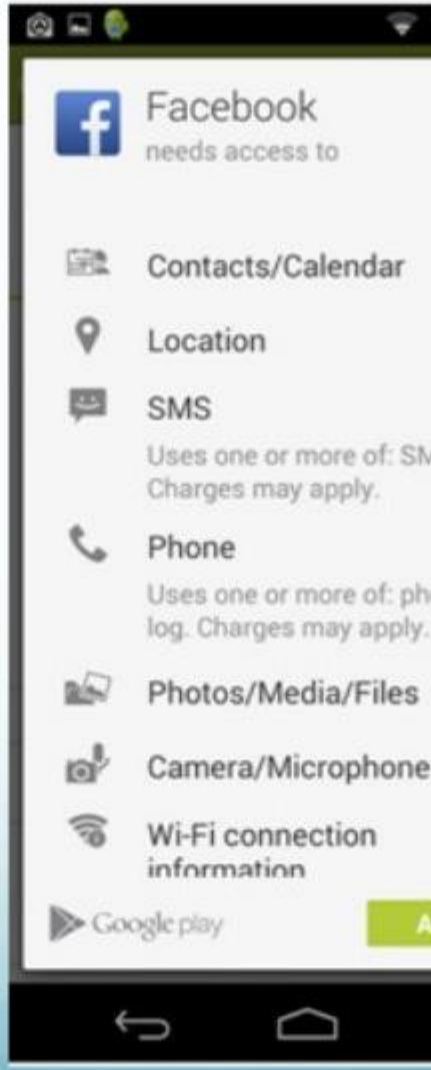
- The **AndroidManifest.xml file** *contains information of your package*, including components of the application such as activities, services, broadcast receivers, content providers etc.
- It performs some other tasks also:
- It is **responsible to protect the application** to access any protected parts by providing the permissions.
- It also **declares the android api** that the application is going to use.
- It **lists the instrumentation classes**. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.
- This is the required xml file for all the android application and located inside the root directory.

AndroidManifest.xml

- XML configuration file
- Every application must have it
- Contains:
  - application's name, icon, labels
  - Linked libraries
  - Application Components:
    - Activities – **<activity>**
    - Services – **<service>**
    - Broadcast receivers – **<receiver>**
    - Content providers – **<provider>**
  - Components specify what kind of Intent they accept with an <intent-filter>
    - Component has a <intent-filter> in AndroidMainfest.xml file, it's exposed by defalut
    - Content provider do not use intents
  - **Permissions**
- Don't export app components unless you want other apps on system to interact with your app

Permissions Manifest F

- Permissions are granted at install time
  - All or nothing.
- API-defined permissions in *AndroidManifest.Permissions* class



Specifying required permissions

- e.g. If your application needs access to the Internet, specify the INTERNET permission

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.testapps.test1">
        ...
        <uses-permission android:name="android.permission.INTERNET" />
        ...
</manifest>
```

Example

- <manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.snazzyapp">

- **<uses-permission android:name="android.permission.SEND_SMS"/>**
- **<application>**
- **</application>**
- **</manifest>**

Uses-sdk

- **API-APPLICATION PROGRAMMING INTERFACE**
- API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.
- **Uses -sdk** is a element is used to specify the API Level, *not* the version number of the SDK (software development kit) or Android platform. The API Level is always a single integer.

- The **Uses -sdk** offers three key attributes:
- android:minSdkVersion — Specifies the minimum API Level on which the application is able to run. The default value is "1".
- android:targetSdkVersion — Specifies the API Level on which the application is designed to run.
- android:maxSdkVersion — Specifies the maximum API Level on which the application is able to run..

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.javatpoint.hello"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Emulator-AVD

- An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, or Android TV device that you want to simulate in the Android Emulator

INTENT

- **Android Intent** is the *message* that is passed between components such as activities, content providers, broadcast receivers, services etc.
- It is generally used with startActivity() method to invoke activity, broadcast receivers etc.
- The **dictionary meaning** of intent is *intention or purpose*. So, it can be described as the intention to do action.

Android intents are mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

Types of Android Intents

1) Implicit Intent

Implicit Intent doesn't specifiy the component. In such case, intent provides information of available components provided by the system that is to be invoked.

For example, you may write the following code to view the webpage.

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

2) Explicit Intent

Explicit Intent specifies the component. In such case, intent provides the external class to be invoked.

```
Intent i = new Intent(getApplicationContext(), ActivityTwo.class);
startActivity(i);
```

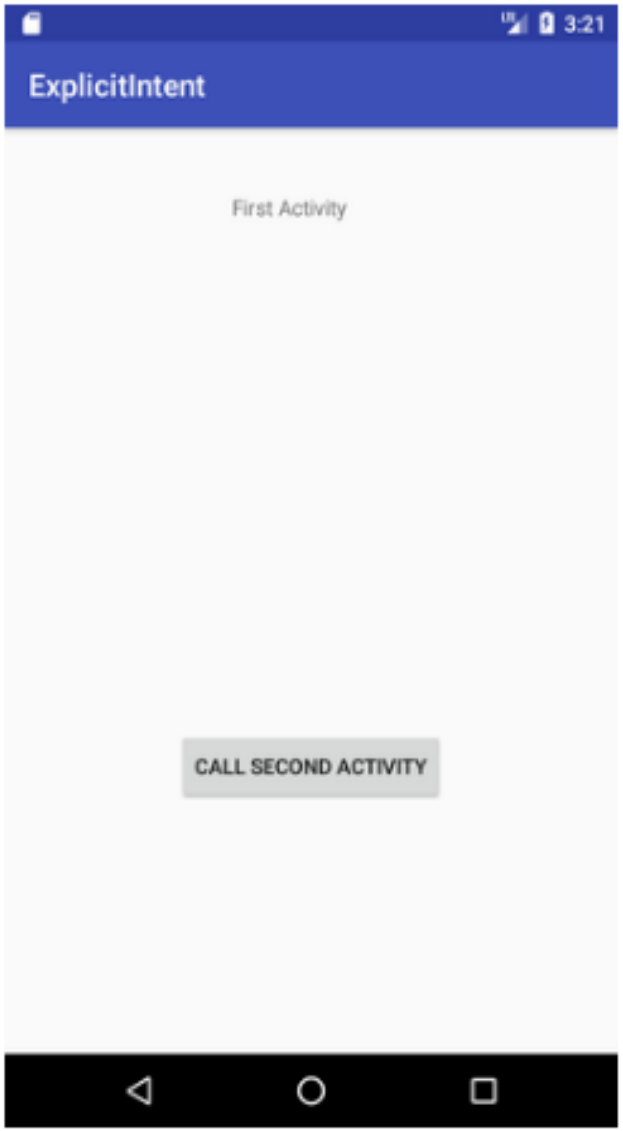
IMIPLICT INTENT

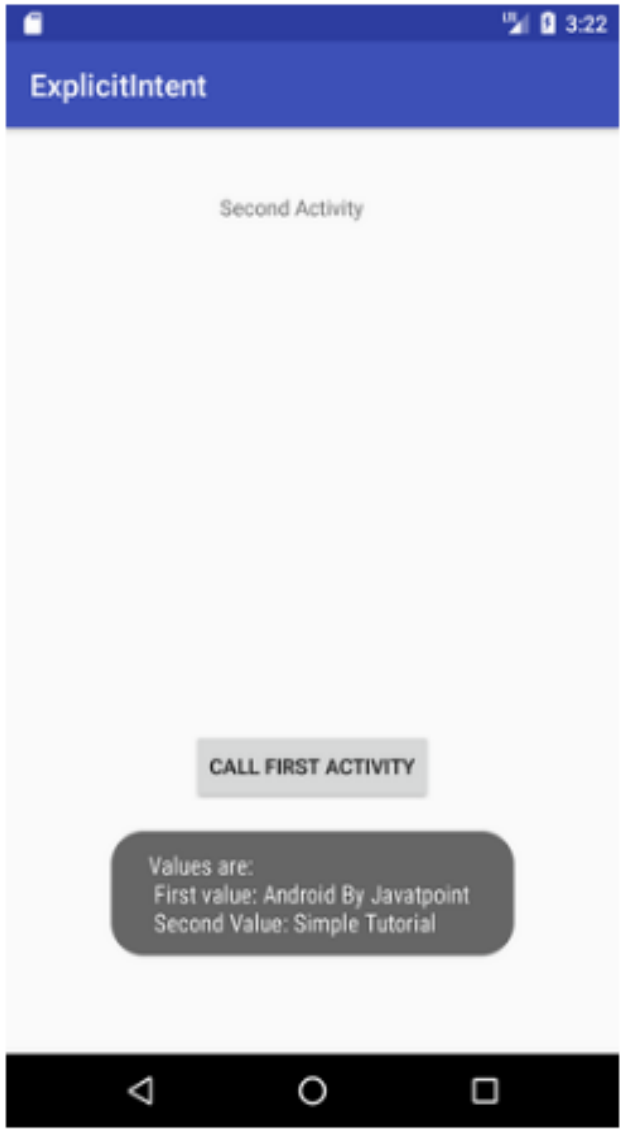
```
public class MainActivity extends AppCompatActivity {

    Button button;
    EditText editText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button = findViewById(R.id.button);
        editText =  findViewById(R.id.editText);

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String url=editText.getText().toString();
                Intent intent=new Intent(Intent.ACTION_VIEW, Uri.parse(url));
                startActivity(intent);
            }
        })
    }
}
```

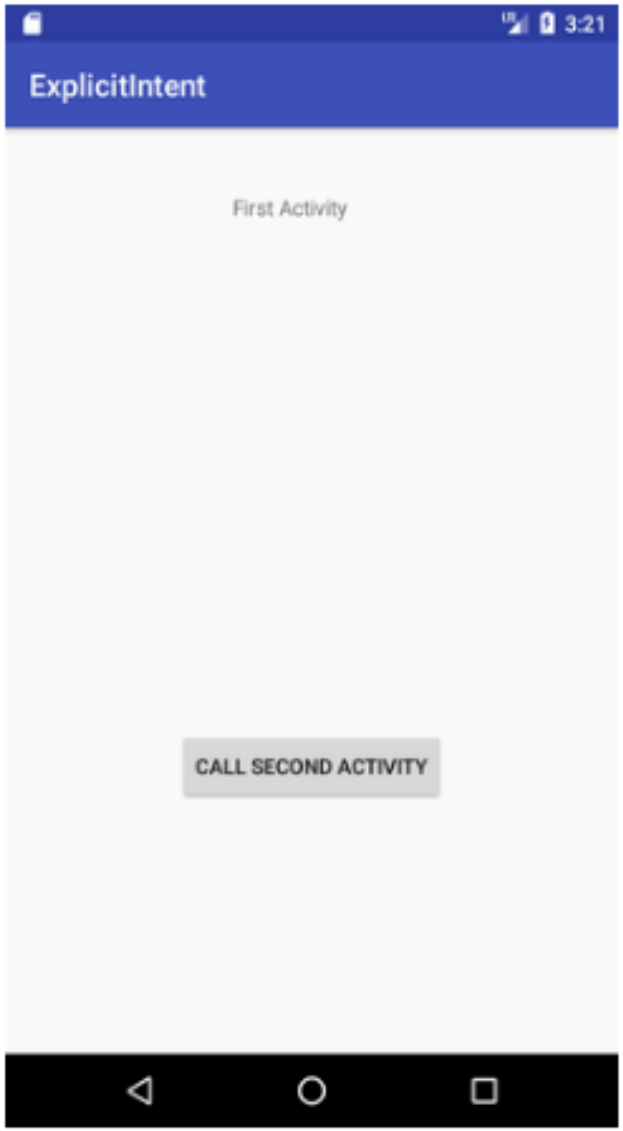
Android Explicit intent

- **Android Explicit intent** specifies the component to be invoked from activity. In other words, we can call another activity in android by explicit intent.
- We can also pass the information from one activity to another using explicit intent.

EXPLICIT INTENT







```
public class FirstActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_first);
    }
    public void callSecondActivity(View view){
        Intent i = new Intent(getApplicationContext(), SecondActivity.class)
     ;
        i.putExtra("Value1", "Android By Javatpoint");
        i.putExtra("Value2", "Simple Tutorial");
        // Set the request code to any code you like, you can identify the
        // callback via this code
        startActivity(i);
    }

}
```

```
public class SecondActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        Bundle extras = getIntent().getExtras();
        String value1 = extras.getString("Value1");
        String value2 = extras.getString("Value2");
        Toast.makeText(getApplicationContext(),"Values are:\n First value: "+val
      ue1+
              "\n Second Value: "+value2, Toast.LENGTH_LONG).show();
    }
    public void callFirstActivity(View view){
        Intent i = new Intent(getApplicationContext(), FirstActivity.class);
        startActivity(i);
    }

}
```