Lab 9
Working with Phone call and SMS

Android Phone Calls

In android, we can easily make a phone call from our android applications by invoking built-in phone calls app using Intents action (**ACTION_CALL**).

Generally, the Intent object in android with proper action (**ACTION_CALL**) and data will help us to launch a built-in phone calls app to make a phone calls in our application.

In android, Intent is a messaging object which is used to request an action from another app component such as activities, services, broadcast receivers, and content providers. To know more about an Intent object in android check this Android Intents with Examples.

To make a phone call using Intent object in android application, we need to write the code like as shown below.

```
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
startActivity(callIntent);
```

If you observe above code, we are using Intent object **ACTION_CALL** action to make a phone call based on our requirements.

Now we will see how to make a phone call in android application using Intent action (**ACTION_CALL**) with examples.

Example

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
      android:id="@+id/fstTxt"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="150dp"
        android:text="Mobile No"
        />
    <EditText
        android:id="@+id/mblTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10">
    </EditText>
    <Button
        android:id="@+id/btnCall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:text="Call" />
</LinearLayout>
```

MainActivity.java

```
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
    private EditText txtPhone;
    private Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtPhone = (EditText)findViewById(R.id.mblTxt);
        btn = (Button)findViewById(R.id.btnCall);
```

```
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                callPhoneNumber();
            }
        });
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults)
    {
        if(requestCode == 101)
        {
            if(grantResults[0] == PackageManager.PERMISSION_GRANTED)
            {
                callPhoneNumber();
            }
        }
    }

    public void callPhoneNumber()
    {
        try
        {
            if(Build.VERSION.SDK_INT > 22)
            {
                if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {
                    ActivityCompat.requestPermissions(MainActivity.this, new String[]
{Manifest.permission.CALL_PHONE}, 101);
                    return;
                }

                Intent callIntent = new Intent(Intent.ACTION_CALL);
                callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
                startActivity(callIntent);

            }

        }
        catch (Exception ex)
        {
            ex.printStackTrace();
        }
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.phonecallexample">
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Android Send SMS

In android, we can send SMS from our android application in two ways either by using **SMSManager** API or Intents based on our requirements.

If we use **SMSManager** API, it will directly send SMS from our application. In case if we use Intent with proper action (**ACTION_VIEW**), it will invoke a built-in SMS app to send SMS from our application.

Android Send SMS using SMSManager API

In android, to send SMS using SMSManager API we need to write the code like as shown below.

```
SmsManager smgr = SmsManager.getDefault();
smgr.sendTextMessage(MobileNumber,null,Message,null,null);
```

Parameters	
`destinationAddress`	`String`: This value cannot be `null`.

`scAddress`	`String`: This value may be `null`.
`text`	`String`: This value cannot be `null`.
`sentIntent`	`PendingIntent`: This value may be `null`.
`deliveryIntent`	`PendingIntent`: This value may be `null`.
`messageId`	`long`: An id that uniquely identifies the message requested to be sent.

SMSManager API required **SEND_SMS** permission in our android manifest to send SMS. Following is the code snippet to set **SEND_SMS** permissions in manifest file.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Android Send SMS using Intent

In android, Intent is a messaging object which is used to request an action from another app component such as activities, services, broadcast receivers, and content providers. To know more about an Intent object in android check this Android Intents with Examples.

To send SMS using the Intent object, we need to write the code like as shown below.

```
Intent sInt = new Intent(Intent.ACTION_VIEW);
sInt.putExtra("address", new String[]{txtMobile.getText().toString()});
sInt.putExtra("sms_body",txtMessage.getText().toString());
sInt.setType("vnd.android-dir/mms-sms");
```

Even for Intent, it required a **SEND_SMS** permission in our android manifest to send SMS. Following is the code snippet to set **SEND_SMS** permissions in manifest file.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Now we will see how to send SMS in android application using **SMSManager** API with examples.

Example

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/fstTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="150dp"
        android:text="Mobile No" />
    <EditText
        android:id="@+id/mblTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10"/>

    <TextView
        android:id="@+id/secTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Message"
        android:layout_marginLeft="100dp" />
    <EditText
        android:id="@+id/msgTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10" />
    <Button
        android:id="@+id/btnSend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:text="Send SMS" />
</LinearLayout>
```

MainActivity.java

```
package com.example.sendsmsexample;
import android.content.Intent;
import android.net.Uri;
import android.provider.Telephony;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private EditText txtMobile;
    private EditText txtMessage;
    private Button btnSms;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtMobile = (EditText)findViewById(R.id.mblTxt);
        txtMessage = (EditText)findViewById(R.id.msgTxt);
        btnSms = (Button)findViewById(R.id.btnSend);
        btnSms.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try{
                    SmsManager smgr = SmsManager.getDefault();

smgr.sendTextMessage(txtMobile.getText().toString(),null,txtMessage.getText().toString(),null,null);
                    Toast.makeText(MainActivity.this, "SMS Sent Successfully",
Toast.LENGTH_SHORT).show();
                }
                catch (Exception e){
                    Toast.makeText(MainActivity.this, "SMS Failed to Send, Please try again",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sendsmsexample">
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
```

```
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Android Send Email

In android, we can easily send an email from our android application using existing email clients such as **GMAIL**, **Outlook**, etc. instead of building an email client from scratch.

Generally, the Intent object in android with proper action (**ACTION_SEND**) and data will help us to launch the available email clients to send an email in our application.

In android, Intent is a messaging object which is used to request an action from another app component such as activities, services, broadcast receivers, and content providers. To know more about an Intent object in android check this Android Intents with Examples.

To send an email using the Intent object in android application, we need to write the code as shown below.

```
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra(Intent.EXTRA_EMAIL, new String[]{"support@example.com"});
it.putExtra(Intent.EXTRA_SUBJECT, "Welcome to example");
it.putExtra(Intent.EXTRA_TEXT, "Hi Guest, Welcome to example Site");
it.setType("message/rfc822");
```

If you observe above code we used multiple components to send email, those are

it - Our local implicit intent

ACTION_SEND - It’s an activity action that specifies that we are sending some data.

putExtra - we use this **putExtra()** method to add extra information to our Intent. Here we can add the following things.

- EXTRA_EMAIL - It's an array of email addresses
- EXTRA_SUBJECT - The subject of the email that we want to send
- EXTRA_TEXT - The body of the email

The android Intent object is having different options such as EXTRA_CC, EXTRA_BCC, EXTRA_HTML_TEXT, EXTRA_STREAM, etc. to add different options for an email client.

setType - We use this property to set the MIME type of data that we want to send. Here we used "**message/rfc822**" and other MIME types are "**text/plain**" and "**image/jpg**".

Now we will see how to send an email in android application using an Intent object with examples.

Example

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="20dp"
   android:paddingRight="20dp"
   android:orientation="vertical" >
   <EditText
      android:id="@+id/txtTo"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="To"/>
   <EditText
      android:id="@+id/txtSub"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Subject"/>
   <EditText
      android:id="@+id/txtMsg"
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight="1"
      android:gravity="top"
```

```
        android:hint="Message"/>
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Send"
        android:id="@+id/btnSend"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.sendmailexample;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    private EditText eTo;
    private EditText eSubject;
    private EditText eMsg;
    private Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        eTo = (EditText)findViewById(R.id.txtTo);
        eSubject = (EditText)findViewById(R.id.txtSub);
        eMsg = (EditText)findViewById(R.id.txtMsg);
        btn = (Button)findViewById(R.id.btnSend);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent it = new Intent(Intent.ACTION_SEND);
                it.putExtra(Intent.EXTRA_EMAIL, new String[]{eTo.getText().toString()});
                it.putExtra(Intent.EXTRA_SUBJECT,eSubject.getText().toString());
                it.putExtra(Intent.EXTRA_TEXT,eMsg.getText());
                it.setType("message/rfc822");
                startActivity(Intent.createChooser(it,"Choose Mail App"));
            }
        });
```

```
    }
}
```

If you observe above code we used multiple components to send email, those are

it - Our local implicit intent

ACTION_SEND - It's an activity action that specifies that we are sending some data.

putExtra - we use this **putExtra()** method to add extra information to our Intent. Here we can add the following things.

- EXTRA_EMAIL - It's an array of email addresses
- EXTRA_SUBJECT - The subject of the email that we want to send
- EXTRA_TEXT - The body of the email

setType - We use this property to set the MIME type of data that we want to send. Here we used "**message/rfc822**" and other MIME types are "**text/plain**" and "**image/jpg**".

We need to add **MIME type** in our android manifest file for that open android manifest file (**AndroidManifest.xml**) and write the code like as shown below

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.sendmailexamp
le">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
                <action android:name="android.intent.action.SEND"/>
                <category android:name="android.intent.category.DEFAULT"/>
```

```
            <data android:mimeType="message/rfc822"/>
        </intent-filter>
    </activity>
  </application>
</manifest>
```

If you observe above **AndroidManifest.xml** file we added following extra fields of Intent filters.

action - we use this property to define that the activity can perform **SEND** action.

category - we included the **DEFAULT** category for this activity to be able to receive implicit intents.

data - the type of data the activity can send.