MODULE – 4: ANDROID MESSAGING AND NETWORKING

Syllabus: SMS Messaging – Sending Email – Networking – Downloading Binary Data, Text files – Accessing Web Services – Performing Asynchronous Call – Creating your own services – Communicating between a service and an activity – Binding activities to services

4.1 SMS MESSAGING

SMS messaging is one of the main *killer applications* on a mobile phone today — for some users as necessary as the phone itself. Any mobile phone you buy today should have at least SMS messaging capabilities, and nearly all users of any age know how to send and receive such messages. Android comes with a built-in SMS application that enables you to send and receive SMS messages. However, in some cases you might want to integrate SMS capabilities into your own android application. For example, you might want to write an application that automatically sends a SMS message at regular time intervals. For example, this would be useful if you wanted to track the location of your kids — simply give them an Android device that sends out an SMS message containing its geographical location every 30 minutes.

4.1.1 Sending SMS Messages Programmatically

To create an application that can send SMS, following are the steps to be followed:

- Create a new android application.
- Add the following statements in to the main.xml file:

```
<Button
android:id="@+id/btnSendSMS"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Send SMS" />
```

- In the AndroidManifest.xml file, add the following statements:

```
<uses-sdk android:minSdkVersion="8" />
<uses-permission
     android:name="android.permission.SEND_SMS">
</uses-permission>
```

- Add the following statements to the MainActivity.java file:

```
import android.app.PendingIntent;
import android.content.Intent;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity
{
     Button btnSendSMS;
/** Called when the activity is first created. */
```

```
        @Override
        public void onCreate(Bundle savedInstanceState)
        {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.main);

              btnSendSMS = (Button) findViewById(R.id.btnSendSMS);

              btnSendSMS.setOnClickListener(new View.OnClickListener()
              {
                    public void onClick(View v)
                    {
                          sendSMS("5556", "Hello my friends!");
                    }
              });
        }

        private void sendSMS(String phoneNumber, String message)
        {
          SmsManager sms = SmsManager.getDefault();
          sms.sendTextMessage(phoneNumber, null, message, null, null);
        }
}
```

- Open two AVDs using AVD Manager. Run your application on one AVD to send SMS to another AVD. It looks as shown below –

Observe the `sendSMS()` method, which contains `sendTextMessage()` method. Following are the five arguments to the `sendTextMessage()` method:

- `destinationAddress` — Phone number of the recipient
- `scAddress` — Service center address; use null for default SMSC
- `text` — Content of the SMS message
- `sentIntent` — Pending intent to invoke when the message is sent
- `deliveryIntent` — Pending intent to invoke when the message has been delivered

4.2 SENDING E-MAIL

One can set email through Android program. Following are the steps involved:

- Create a new android application.
- Add the following statements in to the main.xml file:

```
<Button
android:id="@+id/btnSendEmail"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Send Email" />
```

- Add the following statements in bold to the `MainActivity.java` file:

```
import android.content.Intent;
import android.net.Uri;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity
{
     Button btnSendEmail;
     /** Called when the activity is first created. */
     @Override

     public void onCreate(Bundle savedInstanceState)
     {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          btnSendEmail = (Button) findViewById(R.id.btnSendEmail);

        btnSendEmail.setOnClickListener(new View.OnClickListener()
          {
               public void onClick(View v)
               {
                    String[] to = {"weimenglee@learn2develop.net",
                                        "weimenglee@gmail.com"};
                    String[] cc = {"course@learn2develop.net"};
                    sendEmail(to,cc,"Hello", "Hello my friends!");
               }
          });
```

```
        }

        //---sends an SMS message to another device---
        private void sendEmail(String[] emailAddresses, String[]
                  carbonCopies, String subject, String message)
        {
             Intent emailIntent = new Intent(Intent.ACTION_SEND);
             emailIntent.setData(Uri.parse("mailto:"));
             String[] to = emailAddresses;
             String[] cc = carbonCopies;
             emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
             emailIntent.putExtra(Intent.EXTRA_CC, cc);
             emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
             emailIntent.putExtra(Intent.EXTRA_TEXT, message);
             emailIntent.setType("message/rfc822");
             startActivity(Intent.createChooser(emailIntent,"Email"));
        }
}
```

- Test the application by running it. When you click on *Send Mail* button you can see Email application launched on device as shown below –

4.3 NETWORKING

One can communicate with the external world via SMS, email or using HTTP protocol. Using the HTTP protocol, you can perform a wide variety of tasks, such as downloading web pages from a web server, downloading binary data, and so on. The following project creates an Android project so that you can use the HTTP protocol to connect to the Web to download all sorts of data.

- Create a new android project and name it as Networking
- Add the following line in AndroidManifest.xml file:

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

- Import the following namespaces in the MainActivity.java file:

```
package net.learn2develop.Networking;
import android.app.Activity;
import android.os.Bundle;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.widget.ImageView;
import android.widget.Toast;
import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
```

- Define the OpenHttpConnection() method in the MainActivity.java file:

```
public class MainActivity extends Activity
{
 private InputStream OpenHttpConnection(String urlString)
                                      throws IOException
 {
     InputStream in = null;
     int response = -1;
     URL url = new URL(urlString);
     URLConnection conn = url.openConnection();

     if (!(conn instanceof HttpURLConnection))
          throw new IOException("Not an HTTP connection");
     try
     {
       HttpURLConnection httpConn = (HttpURLConnection) conn;
       httpConn.setAllowUserInteraction(false);
       httpConn.setInstanceFollowRedirects(true);
       httpConn.setRequestMethod("GET");
       httpConn.connect();
       response = httpConn.getResponseCode();
```

```
                    if (response == HttpURLConnection.HTTP_OK)
                    {
                         in = httpConn.getInputStream();
                    }
               }
               catch (Exception ex)
               {
                    throw new IOException("Error connecting");
               }
               return in;
          }

          /** Called when the activity is first created. */
          @Override

          public void onCreate(Bundle savedInstanceState)
          {
               super.onCreate(savedInstanceState);
               setContentView(R.layout.main);
          }
}
```

- Run the application to establish a connection.
- **Working Procedure:** Because you are using the HTTP protocol to connect to the Web, your application needs the INTERNET permission; hence, the first thing you do is add the permission in the `AndroidManifest.xml` file. You then define the `OpenHttpConnection()` method, which takes a URL string and returns an `InputStream` object. Using an InputStream object, you can download the data by reading bytes from the stream object. In this method, you made use of the `HttpURLConnection` object to open an HTTP connection with a remote URL. You set all the various properties of the connection, such as the request method, and so on. After you try to establish a connection with the server, you get the HTTP response code from it. If the connection is established (via the response code `HTTP_OK`), then you proceed to get an `InputStream` object from the connection. Using the `InputStream` object, you can then start to download the data from the server.

4.3.1 Downloading the Binary Data

In the previous section, we have seen how to establish network connection for android application. Once we establish connection, we can download the data from the web. The data may be binary data (an image file, for example), or text data etc. Consider the following example project which downloads an image file from the web.

- Use the same application that has been created in the previous section (as network connection has already been established).

- Add an ImageView within <LinearLayout> inside activity_main.xml file as follows:

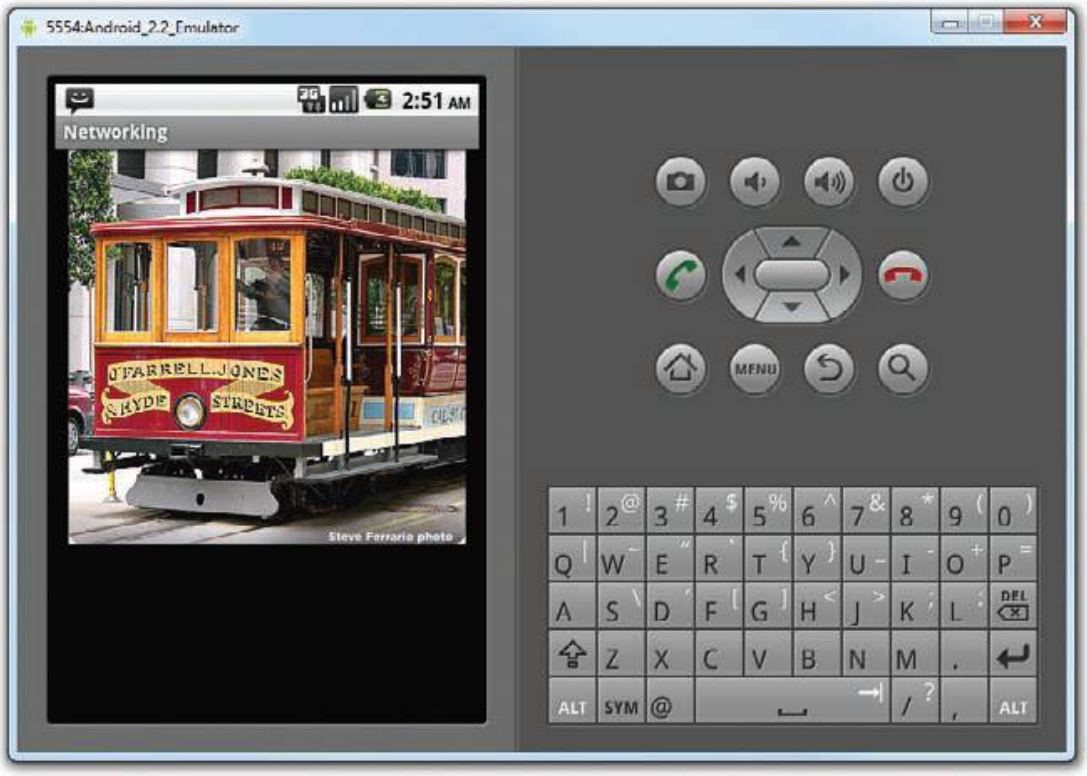
```
<ImageView
android:id="@+id/img"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center" />
```

- Add the following statements to the MainActivity.java file:

```
public class MainActivity extends Activity
{
     ImageView img;
     private InputStream OpenHttpConnection(String urlString)
                          throws IOException
     {
          //the code used in previous section
     }

     private Bitmap DownloadImage(String URL)
     {
          Bitmap bitmap = null;
          InputStream in = null;
          try
          {
               in = OpenHttpConnection(URL);
               bitmap = BitmapFactory.decodeStream(in);
               in.close();
          } catch (IOException e1)
          {
               Toast.makeText(this, e1.getLocalizedMessage(),
                              Toast.LENGTH_LONG).show();
               e1.printStackTrace();
          }
          return bitmap;
}
/** Called when the activity is first created. */
@Override

public void onCreate(Bundle savedInstanceState)
{
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);

     //---download an image---
     Bitmap bitmap = DownloadImage(
    "http://www.streetcar.org/mim/cable/images/cable-01.jpg");

     img = (ImageView) findViewById(R.id.img);
```

```
            img.setImageBitmap(bitmap);
        }
    }
```

- **Working procedure:** The `DownloadImage()` method takes the URL of the image to download and then opens the connection to the server using the `OpenHttpConnection()` method that you have defined earlier. Using the InputStream object returned by the connection, the `decodeStream()` method from the `BitmapFactory` class is used to download and decode the data into a Bitmap object. The `DownloadImage()` method returns a Bitmap object.
- Run the application to see the following output:



4.3.2 Downloading Text Files

One can download plain-text file from web. For example, you might be writing an RSS Reader application and hence need to download RSS XML feeds for processing. The following program shows how you can download a plain-text file in your application.

- Use the project created before (which has network connectivity).
- Write the following code in MainActivity.java file:

```
public class MainActivity extends Activity
{
      ImageView img;
      private InputStream OpenHttpConnection(String urlString)
                                    throws IOException
      {
```

```
        //code used for network connectivity
}
private Bitmap DownloadImage(String URL)
{
        //code used for downloading image/binary file
}
private String DownloadText(String URL)
{
        int BUFFER_SIZE = 2000;
        InputStream in = null;
        try
        {
                in = OpenHttpConnection(URL);
        } catch (IOException e1)
        {
                Toast.makeText(this, e1.getLocalizedMessage(),
                                Toast.LENGTH_LONG).show();
                e1.printStackTrace();
                return "";
        }
        InputStreamReader isr = new InputStreamReader(in);
        int charRead;
        String str = "";
        char[] inputBuffer = new char[BUFFER_SIZE];
        try
        {
                while ((charRead = isr.read(inputBuffer))>0)
                {
                        //---convert the chars to a String---
                  String readString =
                    String.copyValueOf(inputBuffer, 0, charRead);
                  str += readString;
                  inputBuffer = new char[BUFFER_SIZE];
                }
                in.close();
        } catch (IOException e)
        {
                Toast.makeText(this, e.getLocalizedMessage(),
                           Toast.LENGTH_LONG).show();
                e.printStackTrace();
                return "";
        }
return str;
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```
            //---download an image---
            Bitmap bitmap =
                DownloadImage(
                  "http://www.streetcar.org/mim/cable/images/cable-
                  01.jpg");
            img = (ImageView) findViewById(R.id.img);
            img.setImageBitmap(bitmap);

            //---download an RSS feed---
            String str = DownloadText(
                  "http://www.appleinsider.com/appleinsider.rss");
            Toast.makeText(getBaseContext(), str,
                                    Toast.LENGTH_SHORT).show();
        }
    }
```

- Run the application to get the output as shown below –

4.3.3 Performing Asynchronous Calls

Connections discussed in the previous sections were all *synchronous* – that is, the connection to a server will not return until the data is received. In real life, this presents some problems due to network connections being inherently slow. When you connect to a server to download some data, the user interface of your application remains frozen until a response is obtained. In most cases, this is not acceptable. Hence, you need to ensure that the connection to the server is made in an asynchronous fashion.

The easiest way to connect to the server asynchronously is to use the `AsyncTask` class available in the Android SDK. Using `AsyncTask` enables you to perform background tasks in a separate thread and then return the result in a UI thread. Using this class enables you to perform background operations without needing to handle complex threading issues.

Using the previous example of downloading an image from the server and then displaying the image in an `ImageView`, you could wrap the code in an instance of the `AsyncTask` class, as shown below:

```
public class MainActivity extends Activity
{
     ImageView img;
     private class BackgroundTask extends AsyncTask
                    <String, Void, Bitmap>
     {
          protected Bitmap doInBackground(String... url)
          {
               Bitmap bitmap = DownloadImage(url[0]);
               return bitmap;
          }
          protected void onPostExecute(Bitmap bitmap) {
          ImageView img = (ImageView) findViewById(R.id.img);
          img.setImageBitmap(bitmap);
     }
}

private InputStream OpenHttpConnection(String urlString)
throws IOException
{
...
}
```

Basically, you defined a class that extends the AsyncTask class. In this case, there are two methods within the `BackgroundTask` class — `doInBackground()` and `onPostExecute()`. You put all the code that needs to be run asynchronously in the `doInBackground()` method. When the task is completed, the result is passed back via the `onPostExecute()` method. The `onPostExecute()` method is executed on the UI thread, hence it is thread safe to update the `ImageView` with the bitmap downloaded from the server.

To perform the asynchronous tasks, simply create an instance of the `BackgroundTask` class and call its execute() method:

```
@Override
public void onCreate(Bundle savedInstanceState)
{
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        new BackgroundTask().execute(
          "http://www.streetcar.org/mim/cable/images/cable-01.jpg");
    }
```

4.4 CREATING YOUR OWN SERVICES

A service is an application in Android that runs in the background without needing to interact with the user. For example, while using an application, you may want to play some background music at the same time. In this case, the code that is playing the background music has no need to interact with the user, and hence it can be run as a service. Services are also ideal for situations in which there is no need to present a UI to the user.

Following are the steps involved in creating own service.

- Create a new android application.
- Add a new class file to the project and name it MyService.java. Write the following code in it:

```
package net.learn2develop.Services;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class MyService extends Service
{
      @Override
      public IBinder onBind(Intent arg0)
      {
            return null;
      }

      public int onStartCommand(Intent intent, int flags, int startId)
      {
          Toast.makeText(this,"ServiceStarted",Toast.LENGTH_LONG).show();
          return START_STICKY;
      }

      public void onDestroy()
      {
         super.onDestroy();
         Toast.makeText(this,"ServiceDestroyed",Toast.LENGTH_LONG).show();
      }
}
```

The `onBind()` method enables you to bind an activity to a service. This in turn enables an activity to directly access members and methods inside a service. The `onStartCommand()` method is called when you start the service explicitly using the `startService()` method. This method signifies the start of the service, and you code it

to do the things you need to do for your service. In this method, you returned the constant `START_STICKY` so that the service will continue to run until it is explicitly stopped. The `onDestroy()` method is called when the service is stopped using the `stopService()` method. This is where you clean up the resources used by your service.

- In the AndroidManifest.xml file, add the following statement :
```
<service android:name=".MyService" />
```
- In the activity_main.xml file, add the following statements in bold:
```
<Button android:id="@+id/btnStartService"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Start Service" />
<Button android:id="@+id/btnStopService"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Stop Service" />
```

- Add the following statements in bold to the MainActivity.java file:
```
import android.content.Intent;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity
{
  public void onCreate(Bundle savedInstanceState)
  {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);

   Button btnStart = (Button) findViewById(R.id.btnStartService);

   btnStart.setOnClickListener(new View.OnClickListener()
   {
        public void onClick(View v)
        {
     startService(new Intent(getBaseContext(), MyService.class));
        }
   });

 Button btnStop = (Button) findViewById(R.id.btnStopService);
 btnStop.setOnClickListener(new View.OnClickListener()
 {
   public void onClick(View v)
   {
     stopService(new Intent(getBaseContext(), MyService.class));
   }
 });
}
}
```

- Run the application on the emulator and the output screen will be as shown –

4.5 COMMUNICATING BETWEEN A SERVICE AND AN ACTIVITY

Often a service simply executes in its own thread, independently of the activity that calls it. This doesn't pose any problem if you simply want the service to perform some tasks periodically and the activity does not need to be notified of the status of the service. For example, you may have a service that periodically logs the geographical location of the device to a database. In this case, there is no need for your service to interact with any activities, because its main purpose is to save the coordinates into a database. However, suppose you want to monitor for a particular location. When the service logs an address that is near the location you are monitoring, it might need to communicate that information to the activity. In this case, you would need to devise a way for the service to interact with the activity.

4.6 BINDING ACTIVITIES TO SERVICES

Real-world services are usually more sophisticated, requiring the passing of data so that they can do the job correctly for you. Using the service demonstrated earlier that downloads a set of files, suppose you now want to let the calling activity determine what fi les to download, instead of hardcoding them in the service. Here is what you need to do.

First, in the calling activity, you create an Intent object, specifying the service name:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()
```

```
{
      public void onClick(View v)
      {
            Intent intent = new Intent(getBaseContext(), MyService.class);
      }
});
```

You then create an array of URL objects and assign it to the Intent object through its `putExtra()` method. Finally, you start the service using the Intent object:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()
{
      public void onClick(View v)
      {
        Intent intent = new Intent(getBaseContext(), MyService.class);
        try
        {
            URL[] urls = new URL[]
            {
                  new URL("http://www.amazon.com/somefiles.pdf"),
                  new URL("http://www.wrox.com/somefiles.pdf"),
                  new URL("http://www.google.com/somefiles.pdf"),
                  new URL("http://www.learn2develop.net/somefiles.pdf")
            };
            intent.putExtra("URLs", urls);
        } catch (MalformedURLException e)
        {
            e.printStackTrace();
        }
        startService(intent);
      }
  });
```

Note that the URL array is assigned to the Intent object as an Object array. On the service's end, you need to extract the data passed in through the Intent object in the `onStartCommand()` method:

```
@Override
public int onStartCommand(Intent intent, int flags, int startId)
{
      // We want this service to continue running until it is explicitly
      // stopped, so return sticky.

      Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
      Object[] objUrls = (Object[]) intent.getExtras().get("URLs");
      URL[] urls = new URL[objUrls.length];
      for (int i=0; i<objUrls.length-1; i++)
      {
            urls[i] = (URL) objUrls[i];
```

```
        }
        new DoBackgroundTask().execute(urls);
        return START_STICKY;
}
```

The preceding first extracts the data using the `getExtras`() method to return a Bundle object. It then uses the `get`() method to extract out the URL array as an Object array. Because in Java you cannot directly cast an array from one type to another, you have to create a loop and cast each member of the array individually. Finally, you execute the background task by passing the URL array into the `execute`() method.

This is one way in which your activity can pass values to the service. As you can see, if you have relatively complex data to pass to the service, you have to do some additional work to ensure that the data is passed correctly. A better way to pass data is to bind the activity directly to the service so that the activity can call any public members and methods on the service directly.

Question Bank:

1. Explain the procedure for sending an SMS through Android Application.
2. Name the permissions you need to declare in your AndroidManifest.xml file for sending and receiving SMS messages.
3. How do you send an Email via Android Application? Explain.
4. Briefly discuss the networking concept using HTTP protocol in Android.
5. Name the permissions you need to declare in your AndroidManifest.xml file for an HTTP connection.
6. Write an android application to download an image file from the web.
7. Write an android application to download a text file from the web.
8. Discuss asynchronous calls in networking in view of android application.
9. Define a service. How do you create your own service? Explain with a code snippet.
10. Briefly discuss the concept of binding activities to services.