UNIT II

CHAPTER 4

Background Tasks

Syllabus

AsyncTask and AsyncTaskLoader, Connecting to the Internet, Broadcast receivers, Services

Syllabus Topic : AsyncTask and AsyncTaskLoader

4.1 AsyncTask and AsyncTaskLoader

- There are two ways to do background processing in Android : using the AsyncTask class, or using the Loader framework, which includes an AsyncTaskLoader class that uses AsyncTask. In most situations you'll choose the Loader framework, but it's important to know how AsyncTask works so you can make a good choice.
- process do some tasks in the background, off the UI thread.

The UI thread

- When an Android app starts, it creates the *main thread*, which is often called the *UI thread*.
- The UI thread dispatches events to the appropriate user interface (UI) widgets, and it's where your app interacts with components from the Android UI toolkit i.e. components from the android.widget and android.viewpackages

Two rules of Android thread

1. **Do not block the UI thread**

- The UI thread needs to give its attention to drawing the UI and keeping the app responsive to user input.
- If everything happened on the UI thread, long operations such as network access or database queries could block the whole UI. From the user's perspective, the application would appear to hang.
- Even worse, if the UI thread were blocked for more than a few seconds (about 5 seconds currently) the user would be presented with the "application not responding" (ANR) dialog. The user might decide to quit your application and uninstall it.

https://E-next.in

- To make sure your app doesn't block the UI thread.
- Complete all work in less than 16 ms for each UI screen.
- Don't run asynchronous tasks and other long-running tasks on the UI thread. Instead, implement tasks on a background thread using AsyncTask (for short or interruptible tasks) or AsyncTaskLoader(for tasks that are high-priority, or tasks that need to report back to the user or UI).

2. Do UI work only on the UI thread

- Don't use a background thread to manipulate your UI, because the Android UI toolkit is not thread-safe.

4.1.1 AsyncTask

- Use the AsyncTask class to implement an asynchronous, long-running task on a worker thread. (A worker thread is any thread which is not the main or UI thread.) AsyncTask allows you to perform background operations and publish results on the UI thread without manipulating threads or handlers.

☞ **Steps involved for AsyncTask**

Following stpes are involved for AsyncTask is executed.

onPreExecute()	is invoked on the UI thread before the task is executed. This step is normally used to set up the task, for instance by showing a progress bar in the UI.
doInBackground(Param...)	is invoked on the background thread immediately after onPreExecute() finishes. This step performs a background computation, returns a result, and passes the result to onPostExecute(). The doInBackground() method can also call publishProgress(Progress...) to publish one or more units of progress.
onProgressUpdate(Progress...)	runs on the UI thread after publishProgress(Progress...) is invoked. Use onProgressUpdate() to report any form of progress to the UI thread while the background computation is executing. For instance, you can use it to pass the data to animate a progress bar or show logs in a text field.
onPostExecute(Result)	runs on the UI thread after the background computation has finished.

- AsyncTask reference Fig. 4.1.1 of their calling order.

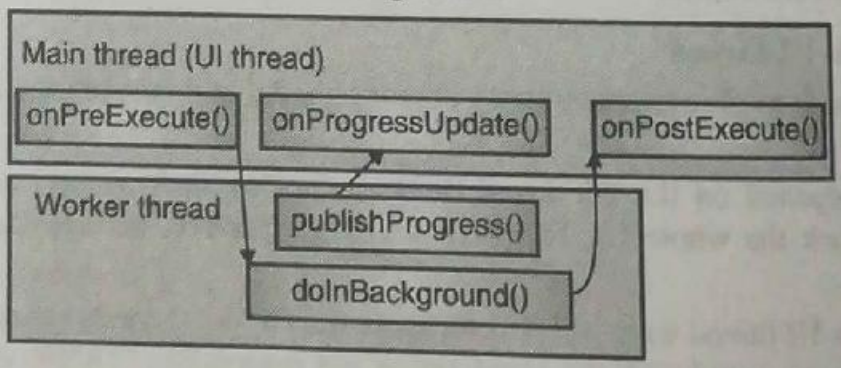


Fig. 4.1.1

https://E-next.in

AsyncTask usage

To use the AsyncTask class, define a subclass of AsyncTask that overrides doInBackground(Params...) method (and usually the onPostExecute(Result) method as well

AsyncTask parameters

In your subclass of AsyncTask, provide the data types for three kinds of parameters:

"Params"	specifies the type of parameters passed to doInBackground() as an array.
"Progress"	specifies the type of parameters passed to publishProgress() on the background thread. These parameters are then passed to the onProgressUpdate() method on the main thread.
"Result"	specifies the type of parameter that doInBackground() returns. This parameter is automatically passed to onPostExecute() on the main thread

Example

```
AsyncTask <"Params","Progress" , "Result" >
```

Here, Specify a data type for each of these parameter types, or use Void if the parameter type will not be used. For example:

```
public class MyAsyncTask extends AsyncTask <String, Void, Bitmap>{}
```

- The "Params" parameter type is String, which means that MyAsyncTask takes one or more strings as

```
parameters in doInBackground(), for example to use in a query.
```

- The "Progress" parameter type is Void, which means that MyAsyncTask won't use the publishProgress() or onProgressUpdate() methods.
- The "Result" parameter type is Bitmap. MyAsyncTask returns a Bitmap in doInbackground(), which is passed into onPostExecute().

4.1.1(A) Examples of an AsyncTask

Here we enter text in EditText View and then copy that text into TextView and after some time show the alert dialog

XML file

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">

<TextView
android:id="@+id/textView2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_marginBottom="145dp" />
```

E-next THE NEXT LEVEL OF EDUCATION

```
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@+id/textView2"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_marginBottom="103dp"
android:layout_marginLeft="134dp"
android:layout_marginStart="134dp"
android:text="Show"
android:textStyle="bold"
android:onClick="click"/>

<EditText
android:id="@+id/editText2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:layout_marginTop="58dp"
android:ems="10"
android:hint="Enter The text here"
android:inputType="textPersonName"
android:textSize="24sp"
android:textStyle="bold" />
</RelativeLayout>
```

E-next
THE NEXT LEVEL OF EDUCATION

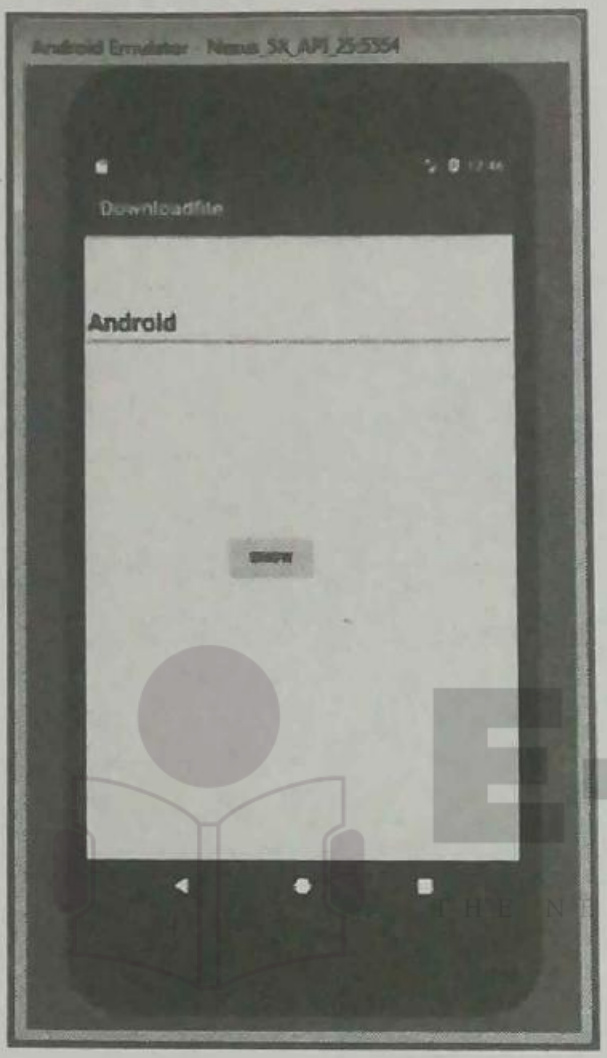
Java File

```
package com.am.mumbai.downloadfile;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
   EditText e1;
```

```
TextView  t1;
ProgressDialog pb;
String s;

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  e1=(EditText)findViewById(R.id.editText2);
  t1=(TextView)findViewById(R.id.textView2);
  t1.setText(" ");
}
public  void click(View v)
{
  new Abc().execute();

}
private class Abc extends AsyncTask<Void,Void,Void>
{

  @Override
  protected Void doInBackground(Void... voids) {


  try {
    Thread.sleep(7000);
  } catch (InterruptedException e) {
    e.printStackTrace();
  }
  return null;
}

  @Override
  protected void onPreExecute() {
  s=e1.getText().toString();
//Toast.makeText(MainActivity.this,"job is started ",Toast.LENGTH_LONG).show();
  super.onPreExecute();
}
  @Override
  protected void onPostExecute(Void aVoid) {
  t1.setText(" "+s);
  e1.setText("");
// Toast.makeText(MainActivity.this,"job is started ",Toast.LENGTH_LONG).show();/

  AlertDialog.Builder a= new AlertDialog.Builder(MainActivity.this);
  a.setTitle(" Message");
  a.setMessage("We learn Android Development  ");
```

E-next THE NEXT LEVEL OF EDUCATION

```
        a.setIcon(R.mipmap.ic_launcher_round);
        a.setPositiveButton("ok",null);
        a.show();
    }
    }
}
```



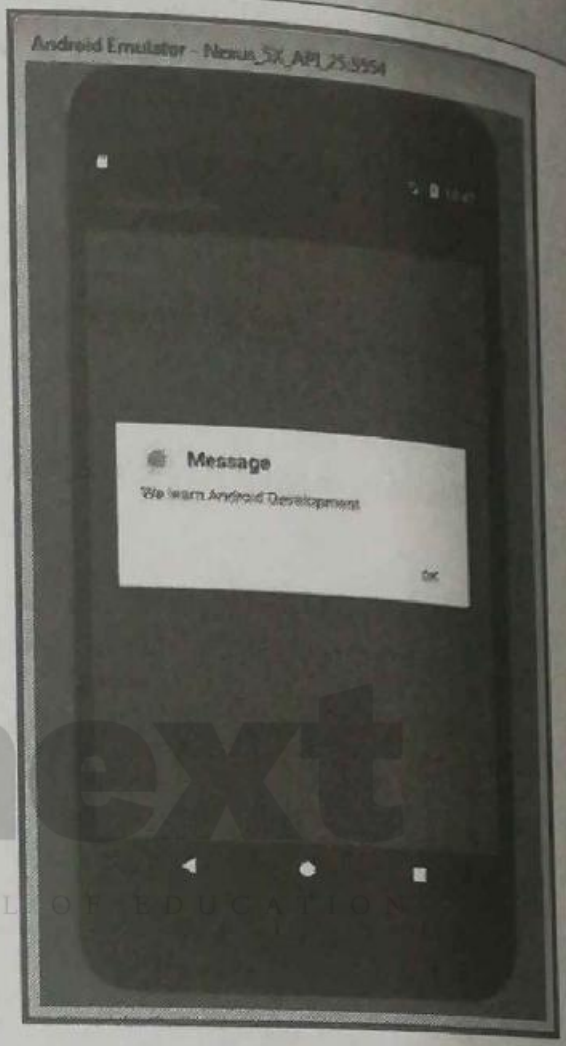


Fig. 4.1.2

☞ **Example**

Download data

xml file

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">

<TextView
android:id="@+id/textView2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
```

E-next
THE NEXT LEVEL OF EDUCATION
https://E-next.in

```
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_marginBottom="145dp" />

<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@+id/textView2"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_marginBottom="103dp"
android:layout_marginLeft="134dp"
android:layout_marginStart="134dp"
android:text="Download"
android:textStyle="bold"
android:onClick="click"/>

<EditText
android:id="@+id/editText2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:layout_marginTop="58dp"
android:ems="10"
android:hint="Enter The text here"
android:inputType="textPersonName"
android:textSize="24sp"
android:textStyle="bold" />
</RelativeLayout>
```

Java file

```
package com.am.mumbai.downloadfile;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
```

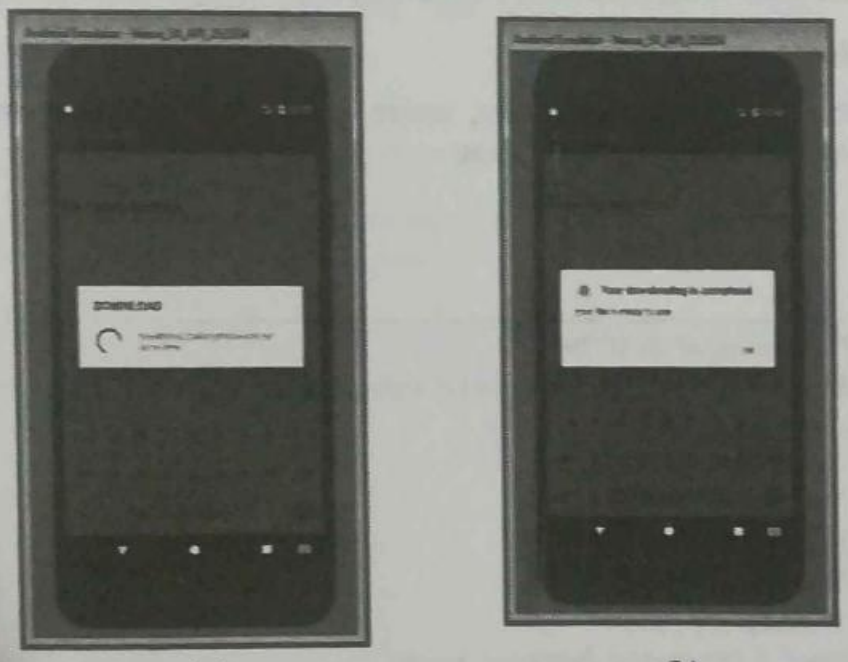
E-next
THE NEXT LEVEL OF EDUCATION

```
public class MainActivity extends AppCompatActivity {
   EditText e1;
   TextView  t1;
   ProgressDialog pb;
   String s;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      e1=(EditText)findViewById(R.id.editText2);
      t1=(TextView)findViewById(R.id.textView2);
      t1.setText(" ");
   }

   public  void click(View v)
   {
      new Abc().execute();
   }
   private class Abc extends AsyncTask<Void,Integer,Void>
   {

      @Override
      protected Void doInBackground(Void... voids) {


// try {
      // Thread.sleep(7000);
      // } catch (InterruptedException e) {
      //   e.printStackTrace();
      // }

      for (int i = 0; i <101; i++) {
         try {
            Thread.sleep(1000);
         } catch (InterruptedException e) {
            e.printStackTrace();

            publishProgress(i);
         }
      }
      return null;
   }
      @Override
      protected void onProgressUpdate(Integer... values) {
      super.onProgressUpdate(values);
      int a;
```

E-next THE NEXT LEVEL OF EDUCATION

```
a=values[0];
pb.setProgress(a);

}

@Override
protected void onPreExecute() {
s=e1.getText().toString();
pb= new ProgressDialog(MainActivity.this);
pb.setMax(100);
pb.setMessage("dowalofing started please wait for some time ");
pb.setTitle("DOWNLOAD");
pb.show();
pb.setProgressStyle(ProgressDialog.STYLE_SPINNER);
super.onPreExecute();

}

@Override
protected void onPostExecute(Void aVoid) {
t1.setText(" "+s);
e1.setText("");
AlertDialog.Builder a= new AlertDialog.Builder(MainActivity.this);
a.setTitle(" Your downloading is completed");
a.setMessage("your file is ready to use ");
a.setIcon(R.mipmap.ic_launcher_round);
a.setPositiveButton("ok",null);
a.show();
pb.dismiss();

}
}
}
```

E-next
THE NEXT LEVEL OF EDUCATION



(a) (b)

Fig. 4.1.3

https://E-next.in

When your data downloaded following alert dialog display(Fig. 4.1.3(b)).

4.1.1(B) Limitations of AsyncTask

- AsyncTask is impractical for some use cases:
- Changes to device configuration cause problems.
- When device configuration changes while an AsyncTask is running, for example if the user changes the screen orientation, the activity that created the AsyncTask is destroyed and re-created. The AsyncTask is unable to access the newly created activity, and the results of the AsyncTask aren't published.
- Old AsyncTask objects stay around, and your app may run out of memory or crash.
- If the activity that created the AsyncTask is destroyed, the AsyncTask is not destroyed along with it. For example, if your user exits the application after the AsyncTask has started, the AsyncTask keeps using resources unless you call cancel().

When to use AsyncTask

- Short or interruptible tasks.
- Tasks that don't need to report back to UI or user.

4.1.2 AsyncTaskLoader

- AsyncTaskLoader is the loader equivalent of AsyncTask. AsyncTaskLoader provides a method, like
- loadInBackground() : Performs actual task in background and returns the result. android.content.AsyncTaskLoader<D> is a loader that uses AsyncTask to perform the task It is bstract class**onCanceled(D data)** : This method is called if task is cancelled before completion.
- It is used to clean up data post cancellation. **cancelLoadInBackground()** : We override this method to cancel the background process. If there is no process in background, it is not going to be called.public static class StringListLoader extends AsyncTaskLoader<List<String>> {}

☞ **AsyncTaskLoader usage**

To define a subclass of AsyncTaskLoader, create a class that extends AsyncTaskLoader<D>, where D is the data type of the data you are loading.

☞ **Example**

Manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.am.mumbai.loader">

  <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">
```

https://E-next.in

```
    <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>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="com.am.mumbai.loader.MainActivity">

<ListView
  android:id="@+id/employees"
  android:layout_height="match_parent"
  android:layout_width="match_parent"/>

</RelativeLayout>
```

E-next THE NEXT LEVEL OF EDUCATION

- Now here we create a new xml file under layout folder name as employeedata.xml ,then right click on layout folder then select New and then select Layout Resourse file and name it as employeedata.xml

employeedata.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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/empid"
    android:textSize="25sp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="#FF8A80"
    android:layout_weight="2"/>
  <TextView
    android:id="@+id/empname"
```

https://E-next.in

```
        android:textSize="25sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FF8A80"
        android:layout_weight="2"/>
</RelativeLayout>
```

Create three java file as Employee.java

```
package com.am.mumbai.loader;
/**
 * Created by kbp on 11/8/2017.
 */
public class Employee {
    public String empid;
    public String name;
    public Employee(String id, String name) {
        this.empid = id;
        this.name = name;
    }
}
```

☞ **EmployeeLoader.java**

```
package com.am.mumbai.loader;

/**
 * Created by kbp on 11/8/2017.
 */
import java.util.ArrayList;
import java.util.List;
import android.content.Context;
import android.support.v4.content.AsyncTaskLoader;
public class EmployeeLoader extends AsyncTaskLoader<List<Employee>> {
    public EmployeeLoader(Context context) {
        super(context);
    }
    @Override
    public List<Employee> loadInBackground() {
        List<Employee> list = new ArrayList<Employee>();
        list.add(new Employee("", "Ashitosh"));
        list.add(new Employee("", "Asha"));
        list.add(new Employee("", "Sampatrao"));
        return list;
    }
}
```

1. To use BaseAdapter ,must need to extend in class and override required methods. Following are some methods,

E-next THE NEXT LEVEL OF EDUCATION

View getView(int position, View view, ViewGroup parent) : Returns a view that displays data at specified position.
Object getItem(int position): Returns the data item for a given position.
long getItemId(int position): Returns the row id associated with the given position.
int getCount() : Count of data items represented by adapter.
Finally to display data, we need to call notifyDataSetChanged().

EmployeeAdapter.java

```
package com.am.mumbai.loader;

/**
 * Created by kbp on 11/8/2017.
 */
import java.util.ArrayList;
import java.util.List;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;
public class EmployeeAdapter extends BaseAdapter {
  private LayoutInflater inflater;
  private List<Employee> employees = new ArrayList<Employee>();
  public EmployeeAdapter(Context context, List<Employee> employees) {
    this.employees = employees;
    inflater = LayoutInflater.from(context);
  }
  @Override
  public View getView(int position, View view, ViewGroup parent) {
    Employee emp = (Employee) getItem(position);
    if (view == null) {
      view = inflater.inflate(R.layout.employeedata, null);
    }
    TextView empid = (TextView) view.findViewById(R.id.empid);
    empid.setText(emp.empid);
    TextView empname = (TextView) view.findViewById(R.id.empname);
    empname.setText(emp.name);
    return view;
  }
  @Override
  public Object getItem(int position) {
    return employees.get(position);
  }
  @Override
  public long getItemId(int position) {
    return position;
```

E-next
THE NEXT LEVEL OF EDUCATION

☞ **Making an HTTP connection**

Most network-connected Android apps use HTTP and HTTPS to send and receive data over the network.

☞ **Building your URI**

- To open an HTTP connection, you need to build a request URI.
- A URI is usually made up of a base URL and a collection of query parameters that specify the resource in question

4.2.1 Example of Connecting to Internet

Xml file

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="com.am.mumbai.connecttointernet.MainActivity">

  <EditText
    android:id="@+id/editText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="22dp"
    android:ems="10"
    android:inputType="textPersonName" />

  <Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Connect To Internet"
    android:onClick="internet"
    android:textSize="24sp"
    android:textStyle="boldlitalic"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_marginRight="38dp"
    android:layout_marginEnd="38dp"
    android:layout_marginBottom="21dp" />
```

E-next

https://E-next.in

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/connecttointernet"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginLeft="16dp"
    android:layout_marginStart="16dp" />


</RelativeLayout>
```

MainActivity.java file

```
package com.am.mumbai.connecttointernet;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
    EditText e1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        e1 = (EditText) findViewById(R.id.editText);
    }
    public void internet(View v)
    {
        String url="https://"+e1.getText().toString();
        Intent i =new Intent(Intent.ACTION_VIEW, Uri.parse(url));
        startActivity(i);
    }
}
```

E-next
https://E-next.in

Fig. 4.2.1

Fig. 4.2.2

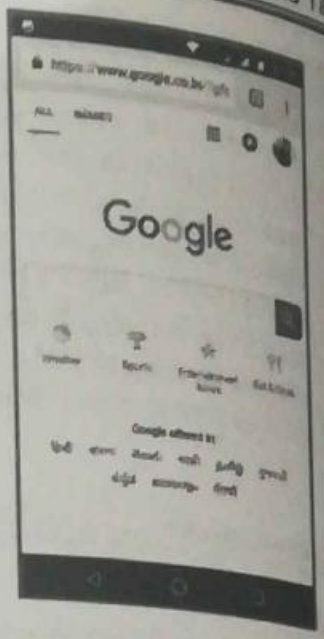


Fig. 4.2.3

– Create app and install it in your mobile and open this app and type given website google.com as shown in Fig. 4.2.2.

☞ **Example :** xml file

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.kbp.intent.MainActivity">
    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="44dp"
        android:ems="10" />
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="54dp"
        android:text="Visit" />
</RelativeLayout>
```

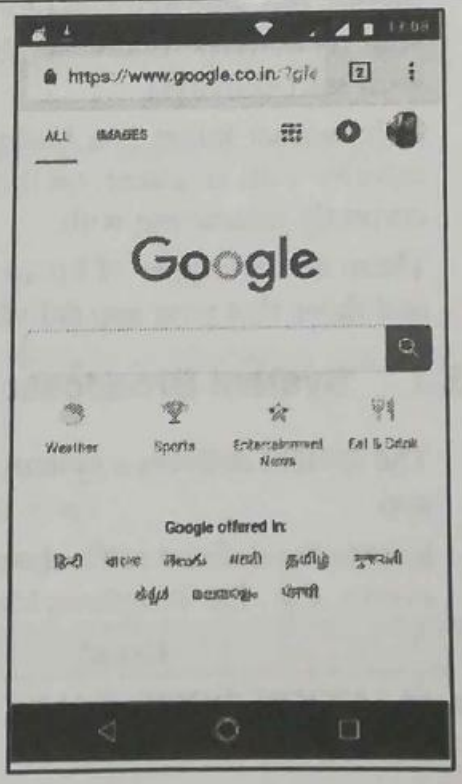
Java file

```
package com.example.kbp.intent;
import android.support.v7.app.AppCompatActivity;
```

E-next THE NEXT LEVEL OF EDUCATION
https://E-next.in

```
 android.os.Bundle;
 android.net.Uri;

 android.app.Activity;
android.content.Intent;
android.view.View;
android.view.View.OnClickListener;
android.widget.Button;
android.widget.EditText;
lass MainActivity extends AppCompatActivity {

erride
ted void onCreate(Bundle savedInstanceState) {
er.onCreate(savedInstanceState);
ContentView(R.layout.activity_main);

 EditText editText1=(EditText)findViewById(R.id.editText1);
on button1=(Button)findViewById(R.id.button1);
n1.setOnClickListener(new OnClickListener() {
Override
blic void onClick(View arg0) {
String url=editText1.getText().toString();
Intent intent=new Intent(Intent.ACTION_VIEW,Uri.parse(url));
startActivity(intent);
```

E-next
THE NEXT LEVEL OF EDUCATION



4.2.4

Fig. 4.2.5

Fig. 4.2.6

https://E-next.in

Syllabus Topic : Broadcast Receivers

4.3 Broadcast Receivers

- We know that intent are of two type as Implicit and Explicit intents

Implicit intents are used to start activities based on registered components that the system is aware of, for example general functionality.	Explicit intents are used to start specific, fully qualified activities, as well as to pass information between activities in your app.

- Here we see about broadcast intents, which don't start activities but instead are delivered to broadcast receivers.

4.3.1 Broadcast Intents

- we know that intents always resulted in an activity being launched, either a specific activity from your application or an activity from a different application that could fulfil the requested action.
- But sometimes an intent doesn't have a specific recipient, and sometimes you don't want an activity to be launched in response to an intent.
- For example, when your app receives a system intent indicating that the network state of a device has changed, you probably don't want to launch an activity, but you may want to disable some functionality of your app.
- Hence we required third type of intent that can be delivered to any interested application know as the broadcast intent.

E-next THE NEXT LEVEL OF EDUCATION

4.3.2 Broadcast Intents Characteristics

- Broadcast intents are delivered using sendBroadcast() or a related method, while other types of intents use startActivity() to start activities. When you broadcast an intent, you never find out to start an activity. Likewise, there's no way for a broadcast receiver to see or capture intents used with startActivity().
- A broadcast intent is a background operation that the user is not normally aware of. Starting a activity with an intent, on the other hand, is a foreground operation that modifies what the user is currently interacting with.
- There are two types of broadcast intents, those delivered by the system (system broadcast intents) and those that your app delivers (custom broadcast intents).

4.3.3 System Broadcast Intents

- The system delivers a system *broadcast intent* when a system event occurs that might interest your app.
- Events are defined as final static fields in the Intent class. Other Android system classes also define events, e.g., the TelephonyManager defines events for the change of the phone state.

Event	Description
Intent.ACTION_BOOT_COMPLETED	Boot completed. Requires the android.permission.RECEIVE_BOOT_COMPLETED permission
Intent.ACTION_POWER_CONNECTED	Power got connected to the device.

https://E-next.in

Event	Description
Intent.ACTION_POWER_DISCONNECTED	Power got disconnected to the device.
Intent.ACTION_BATTERY_LOW	Triggered on low battery. Typically used to reduce activities in your app which consume power.
Intent.ACTION_BATTERY_OKAY	Battery status good again.

4.3.4 Custom Broadcast Intents

- *Custom broadcast intents* are broadcast intents that your application sends out. Use a custom broadcast intent when you want your app to take an action without launching an activity,
- for example when you want to let other apps know that data has been downloaded to the device and is available for them to use.
- To create a custom broadcast intent, create a custom intent action. To deliver a custom broadcast to other apps, pass the intent to sendBroadcast(), sendOrderedBroadcast(), or sendStickyBroadcast().
- For example, the following method creates an intent and broadcasts it to all interested broadcast receivers:

```
public void sendBroadcastIntent() {
  Intent intent = new Intent();
  intent.setAction("com.example.myproject.ACTION_SHOW_TOAST");
  sendBroadcast(intent);
}
```

E-next THE NEXT LEVEL OF EDUCATION

4.3.5 Broadcast Receivers

- Broadcast intents aren't targeted at specific recipients. Instead, interested apps register a component to "listen" for these kind of intents. This listening component is called a *broadcast receiver*.
- Use broadcast receivers to respond to messages that are broadcast from other apps or from the system. To create a broadcast receiver:

1. Define a subclass of the BroadcastReceiver class and implement the onReceive() method.
   - Implement the required onReceive() method.
   - Include any other logic that your broadcast receiver needs.
2. Register the broadcast receiver either dynamically in Java, or statically in your app's manifest file.

4.3.6 Example of Broadcast Receiver

Create a broadcast receiver here consider name as the AlarmReceiver

It is subclass of BroadcastReceiver which shows a Toast message if the incoming broadcast intent has the action ACTION_SHOW_TOAST:

```
private class AlarmReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    if (intent.getAction().equals(ACTION_SHOW_TOAST)) {
```

https://E-next.in

```
        CharSequence text = "Broadcast Received!";
        int duration = Toast.LENGTH_SHORT;
        Toast toast = Toast.makeText(context, text, duration);
        toast.show();
    }
  }
}
```

☞ **Registering your broadcast receiver and setting intent filters**

- To register your broadcast receiver statically, add a <receiver> element to your AndroidManifest.xml file. Within the <receiver> element:
- Use the path to your BroadcastReceiver subclass as the android:name attribute.
- To prevent other applications from sending broadcasts to your receiver, set the optional\ android:exported attribute to false. This is an important security guideline.
- To specify the types of intents the component is listening for, use a nested <intent-filter> element.

☞ **Example**

- Static registration for a broadcast receiver that listens for a custom broadcast intent with "ACTION_SHOW_TOAST" in the name of its action.
- The receiver includes an intent filter that checks whether incoming intents include an action named ACTION_SHOW_TOAST.

```
<receiver
android:name="com.example.myproject.AlarmReceiver"
android:exported="false">
<intent-filter>
  <action android:name="com.example.myproject.intent.action.ACTION_SHOW_TOAST"/>
</intent-filter>
</receiver>
```

☞ **Example**

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="com.am.mumbai.broadcast.MainActivity">
  <Button
    android:id="@+id/button"
    android:layout_width="match_parent"
```

E-next THE NEXT LEVEL OF EDUCATION

```
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_marginTop="88dp"
    android:onClick="click"
    android:text="Broadcast "
    android:textColor="@android:color/holo_red_dark"
    android:textSize="24sp"
    android:textStyle="italic" />
  <ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:srcCompat="@drawable/br"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

</RelativeLayout>
```

MainActivity.java

```
package com.am.mumbai.broadcast;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  public void click(View v)
  {
    Intent i =new Intent(this,MyBroadcast.class);
    i.setAction("COM_I_CUSTUM_ACTION");
    sendBroadcast(i);
    try {
      Thread.sleep(1000);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```

Create a new java file name MyBroadcast and write given code

E-next THE NEXT LEVEL OF EDUCATION

MyBroadcast.java

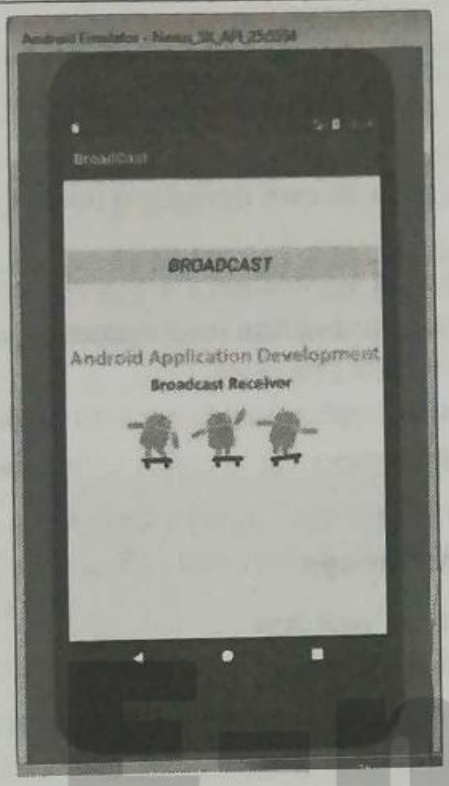
```
package com.am.mumbai.broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

/**
 * Created by kbp on 11/9/2017.
 */
public class MyBroadcast extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent)
   {
      Toast.makeText(context,"Your Broadcast service is started ",Toast.LENGTH_LONG).show();
   }
   }
```

AndroidManifest.xml

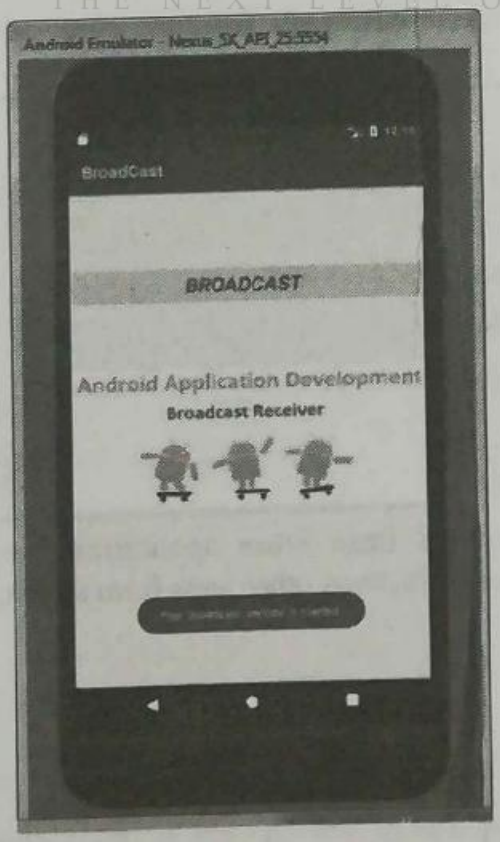
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.am.mumbai.broadcast">
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
<receiver android:name=".MyBroadcast"> </receiver>
      android:theme="@style/AppTheme">
      <activity android:name=".MainActivity">
        <intent-filter>
          <action android:name="android.intent.action.POWER_USAGE_SUMMARY"></action>

          </action>
        </intent-filter>

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

E-next THE NEXT LEVEL OF EDUCATION

```
</application>

</manifest>
```



- When you click on Broadcast Button Broadcast is stated here is output when you click on Broadcast Button



E next
THE NEXT LEVEL OF EDUCATION

Syllabus Topic : Services

4.4 Services

A service is an application component that performs long-running operations, usually in the background. A service doesn't provide a user interface (UI). A service runs in the main thread of its hosting process: the service doesn't create its own thread and doesn't run in a separate process unless you specify that it should.

1. A started service is a service that an application component starts by calling startService(). This method run in the background to perform long-running operations and used to started services for tasks that perform work for remote processes.
2. A bound service is a service that an application component binds to itself by calling bindService(). This service used for another app component interacts with to perform interprocess communication (IPC).

☞ **Steps are implement service in your app**

Following steps are implement service in your app:

1. Declare the service in the manifest.
2. Create implementation code, i.e Started services and Bound services
3. Manage the service lifecycle.

4.4.1 Declaring Services in the Manifest

declare all services in your application's manifest file, <service> element as a child of the <application> element.

```
<manifest ... >
  ...
  <application ... >
    <service android:name="Exampleofservice"
          android:exported="false" />
    ...
  </application>
</manifest>
```

For block access to a service from other applications, declare the service as private. as android:exported attribute to false. This stops other apps from starting your service, even when they use an explicit intent.

4.4.2 Started services

How a service starts

1. An application component such as an activity calls startService()and passes in an Intent. The Intent specifies the service and inclu des any data for the service to use.

E-next THE NEXT LEVEL OF EDUCATION

2. The system calls the service's onCreate() method and any other appropriate callbacks on the main thread
3. The system calls the service's onStartCommand() method, passing in the Intent supplied by the component that calls the service.

- Once started, a service can run in the background indefinitely, even if the component that started it is destroyed. Usually, a started service performs a single operation and does not return a result to the caller.
- For example, when download or upload a file over the network. When the operation is done, the service should stop itself by calling stopSelf().
- For instance, suppose an activity needs to save data to an online database. The activity starts a companion service by passing an Intent to startService(). The service receives the intent in onStartCommand(), connects to the Internet, and performs the database transaction. When the transaction is done, the service uses stopSelf() to stop itself and is destroyed.

4.4.3 IntentService

Most started services don't need to handle multiple requests simultaneously, and if they did it could be a dangerous multi-threading scenario. For this reason, it's probably best if you implement your service using the IntentService class.

IntentService is a useful subclass of Service:

- IntentService automatically provides a worker thread to handle your Intent.
- IntentService handles some of the boilerplate code that regular services need (such as starting and stopping the service).
- IntentService can create a work queue that passes one intent at a time to your onHandleIntent()implementation, so you don't have to worry about multi-threading.

E-next THE NEXT LEVEL OF EDUCATION

To implement IntentService

1. Provide a small constructor for the service.
2. Create an implementation of onHandleIntent() to do the work that the client provides.

Here's an example implementation of IntentService:

```
public class HelloIntentService extends IntentService {
 /**
  * A constructor is required, and must call the super IntentService(String)
  * constructor with a name for the worker thread.
  */
public HelloIntentService() {
   super("HelloIntentService");
}
/**
* The IntentService calls this method from the default worker thread with
* the intent that started the service. When this method returns, IntentService
* stops the service, as appropriate.
```

https://E-next.in

```
 */
@Override
protected void onHandleIntent(Intent intent) {
  // Normally we would do some work here, like download a file.
  // For our sample, we just sleep for 5 seconds.
  try {
    Thread.sleep(5000);
  } catch (InterruptedException e) {
    // Restore interrupt status.
    Thread.currentThread().interrupt();
  }
}
}
```

4.4.4 Bound Services

- A service is "bound" when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, and get results, sometimes using interprocess communication (IPC) to send and receive information across processes.
- A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.
- A bound service generally does not allow components to start it by calling startService().

☞ **Implementing a bound service**

- To implement a bound service, define the interface that specifies how a client can communicate with the service. This interface, which your service returns from the onBind() callback method, must be an implementation of IBinder.
- To retrieve the IBinder interface, a client application component calls bindService().
- Once the client receives the IBinder, the client interacts with the service through that interface.

☞ **Binding to a service**

To bind to a service that is declared in the manifest and implemented by an app component, use bindService() with an explicit Intent.

4.4.5 Service Lifecycle

- The lifecycle of a service is simpler than that of an activity. Because a service has no UI, services can continue to run in the background with no way for the user to know, even if the user switches to another application. This consumes resources and drains battery.
- Like an activity, a service has lifecycle callback methods that you can implement to monitor changes in the service's state and perform work at the appropriate times. The following skeleton service demonstrates each of the lifecycle methods:

https://E-next.in

```
public class ExampleService extends Service {
int mStartMode;       // indicates how to behave if the service is killed
IBinder mBinder;      // interface for clients that bind
boolean mAllowRebind; // indicates whether onRebind should be used
@Override
public void onCreate() {
  // The service is being created
}
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
  // The service is starting, due to a call to startService()
  return mStartMode;
}
@Override
public IBinder onBind(Intent intent) {
  // A client is binding to the service with bindService()
  return mBinder;
}
@Override
public boolean onUnbind(Intent intent) {
  // All clients have unbound with unbindService()
  return mAllowRebind;
}
@Override
public void onRebind(Intent intent) {
  // A client is binding to the service with bindService(),
  // after onUnbind() has already been called
}
@Override
public void onDestroy() {
  // The service is no longer used and is being destroyed
}
}
```

E-next
THE NEXT LEVEL OF EDUCATION

The Fig. 4.4.1 shows a comparison between the started and bound service.

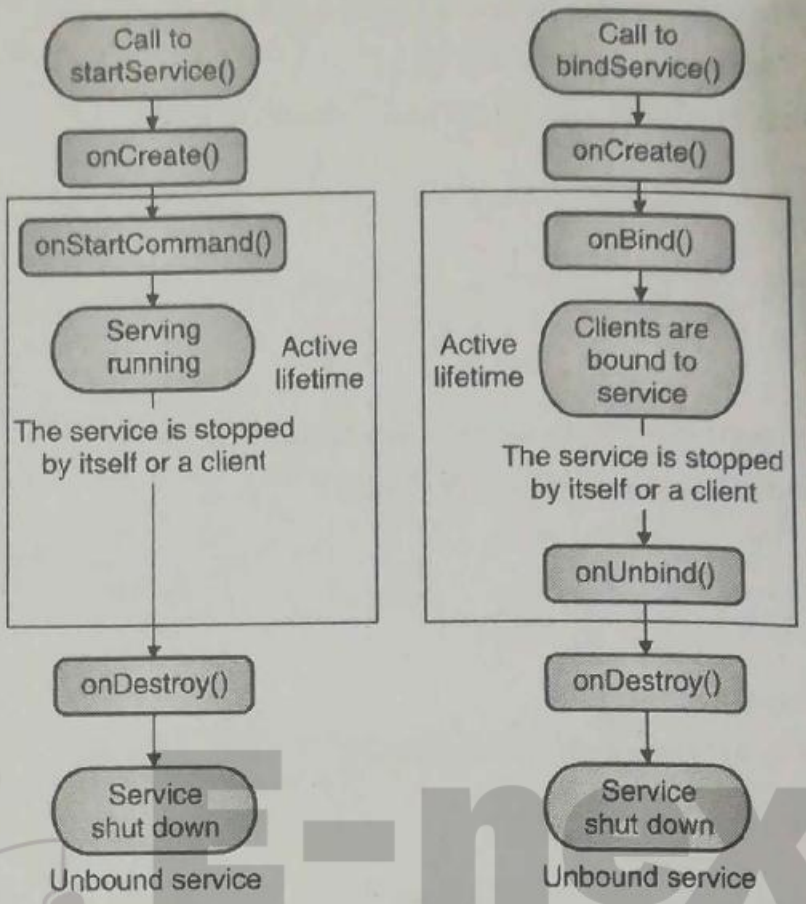


Fig. 4.4.1

E-next THE NEXT LEVEL OF EDUCATION

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    tools:context="com.am.mumbai.musics.MainActivity">
    <Button
        android:id="@+id/button"
        android:layout_width="175sp"
        android:layout_height="100sp"
        android:layout_marginLeft="19dp"
        android:layout_marginStart="19dp"
        android:onClick="start"
        android:text="Start"
        app:layout_constraintEnd_toEndOf="@color/colorPrimaryDark"
        tools:layout_editor_absoluteY="200dp"
        android:layout_marginBottom="38dp"
        android:layout_alignParentBottom="true"
```

https://E-next.in

```
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
  <Button
    android:id="@+id/button2"
    android:layout_width="100dp"
    android:layout_height="50dp"
    android:onClick="stop"
    android:text="Stop"
    app:layout_constraintStart_toStartOf="@mipmap/ic_launcher_round"
    tools:layout_editor_absoluteY="505dp"
    android:layout_above="@+id/button"
    android:layout_alignLeft="@+id/button"
    android:layout_alignStart="@+id/button"
    android:layout_marginBottom="76dp"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
  <ImageView
    android:id="@+id/imageView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/cake3"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true" />
</RelativeLayout>
```

MainActivity.java

```
package com.am.mumbai.musics;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;


public class MainActivity extends AppCompatActivity {

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

E-next THE NEXT LEVEL OF EDUCATION

```
    public  void start(View v)
    {
        Intent i= new Intent (this,myservices.class);

        startService(i);
    }
    public  void stop (View v)
    {
        Intent i= new Intent (this,myservices.class);

        stopService(i);
    }

}
```

- Create a new java file name as myservices.java and write given code

myservices.java

```
package com.am.mumbai.musics;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.provider.MediaStore;
import android.support.annotation.IntDef;
import android.support.annotation.Nullable;

/**
 * Created by kbp on 11/8/2017.
 */

public class myservices extends Service{
    MediaPlayer mp;
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onStart(Intent intent, int startId) {
        mp= MediaPlayer.create(this,R.raw.songname);
        mp.start();
```

E-next THE NEXT LEVEL OF EDUCATION

```
    super.onStart(intent, startId);
  }

  @Override
  public void onDestroy() {
    mp.stop();
    super.onDestroy();
  }
}
```

to start the service you must add service in manifest file as

here we pass <service android:name=".myservices">

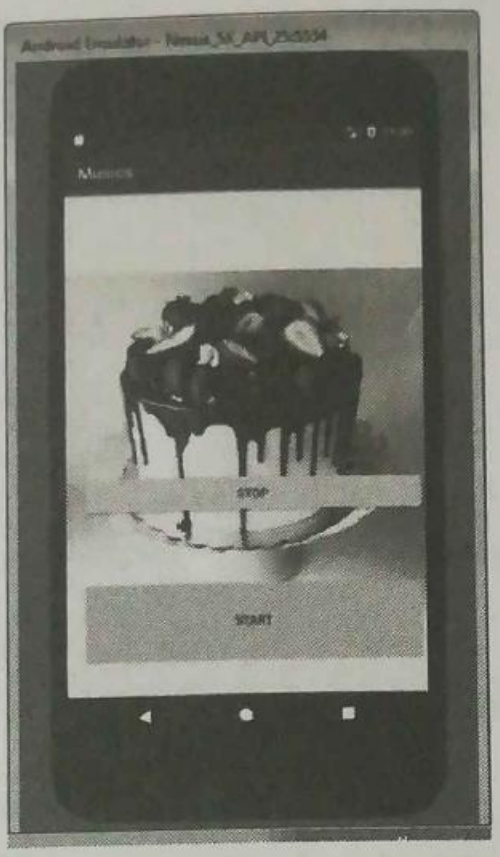
```
        </service>
```

where myservices is the name of java file which is above created

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.am.mumbai.musics">

  <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" />
        <service android:name=".myservices">

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

E-next THE NEXT LEVEL OF EDUCATION



- When you click on start button Music is started and when you click on stop button music is stop
- (Remember that until you click stop button your app does not stop music)

Review Questions

E-next
THE NEXT LEVEL OF EDUCATION

Q. 1 Write short note on AsyncTask and AsyncTaskLoader.**(Refer section 4.1.1)**

Q. 2 List the limitations of AysncTask. **(Refer section 4.1.1(B))**

Q. 3 Write short note AsynceTaskLoader. **(Refer section 4.1.2)**

Q. 4 Explain how to connect to internet ? **(Refer section 4.2)**

Q. 5 Explain broadcast intent. **(Refer section 4.3.1)**

Q. 6 What is service ? **(Refer section 4.4)**

Q. 7 How to start services in android? **(Refer section 4.4.2)**

Q. 8 Describe service lifecycle. **(Refer section 4.4.5)**

https://E-next.in