Services

Learning Objectives

After studying this module learner should be able to

- Define service
- List the uses of services
- Create, start and stop service
- Understand service life cycle
- Create own service

Introduction

In Android a service is an application that runs in the background without any interaction with the user. For example, while using an application, you may want to download some file at the same time. In this case, the code that is downloading file has no need to interact with the user, and hence it can be run as a service. Services are also perfect for circumstances in which there is no need to present a user interface (UI) to the user. For example, consider an application that continually logs the geographical coordinates of the device. In this case, you can write a service to do that in the background. You can create your own services and use them to perform background tasks asynchronously.

To improve application responsiveness and performance, consider implementing a service to handle the task outside the main application lifecycle. Any Services exposed by an Android application must be registered in the Android Manifest file.

Uses Services

You can use services for different purposes. Generally, you use a service when no input is required from the user. Here are some circumstances in which you might want to implement or use an Android service:

- A weather, email, or social network app might implement a service to routinely check for updates.
- A photo or media app that keeps its data in sync online might implement a service to package and upload new content in the background when the device is idle.
- A video-editing app might offload heavy processing to a queue on its service in order to avoid affecting overall system performance for non-essential tasks.
- A news application might implement a service to “pre-load” content by downloading news stories in advance of when the user launches the application, to improve performance.

Creating a Service

To create service, you must define a class that extends the Service base class. Inside your service class, you have to implement four methods discussed below:

Method	Description
onStartCommand()	• The system calls this method when another component, such as an activity, requests that the service be started, by calling startService(). • Once this method executes, the service is started and can run in the background indefinitely. • It is your responsibility to stop the service when its work is done, by calling stopSelf() or stopService(). • If you only want to provide binding, you don't need to implement this method.
onBind()	• The system calls this method when another component wants to bind with the service by calling bindService(). • In your implementation of this method, you must provide an interface that clients use to communicate with the service, by returning an IBinder.

Method	Description
	• If you don't want to allow binding, then you should return null.
onCreate()	• The system calls this method when the service is first created, to perform one-time setup procedures before it calls either onStartCommand() or onBind(). • If the service is already running, this method is not called.
onDestroy()	• The system calls this method when the service is no longer used and is being destroyed. • This method should be implemented to clean up any resources such as threads, registered listeners, receivers, etc. • This is the last call the service receives.

Table-1

Start and Stop a Service

You can use Intents and Activities to launch services using the startService() and bindService() methods. A service can essentially take two forms. The difference between two is as follows:

startService()	bindService()
A service is "started" when an application component starts it by calling startService()	A service is "bound" when an application component binds to it by calling bindService()
Once started, a service can run in the background indefinitely, even if the component that started it is destroyed	A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service.
Usually, a started service performs a single operation and does not return a result to the caller. For example, it might	A bound service offers a client-server interface that allows components to interact with the service, send requests,

download or upload a file over the network. When the operation is done, the service should stop itself	get results, and even do so across processes with inter process communication (IPC)

Table-2

Service Life Cycle

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 as discussed above. Below Figure illustrates the typical callback methods for a service for that are created by startService() and from those created by bindService.

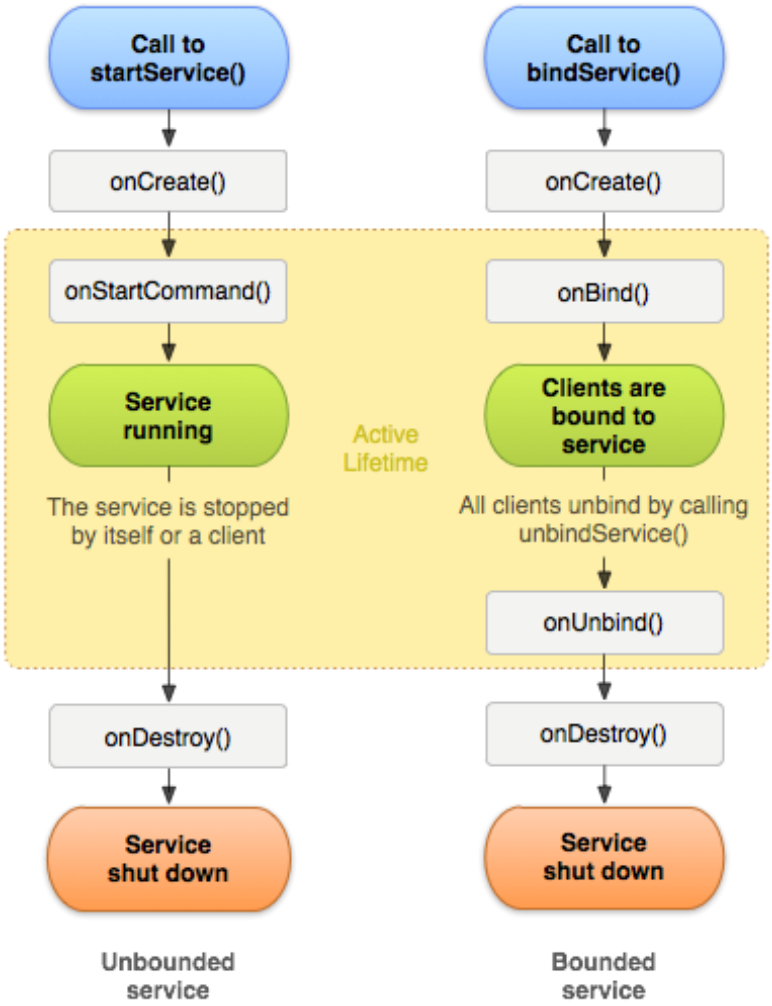

Figure-1

Creating your own service

We will create service to logs counter which starts from 1 and incremented by one at interval of one second. To do so perform following steps:

1. Create a New Android Studio Project with project name ServiceDemo and Main Activity name as ServiceActivity
2. Add new service by right click on package and select New ➔ Service ➔ Service and click.

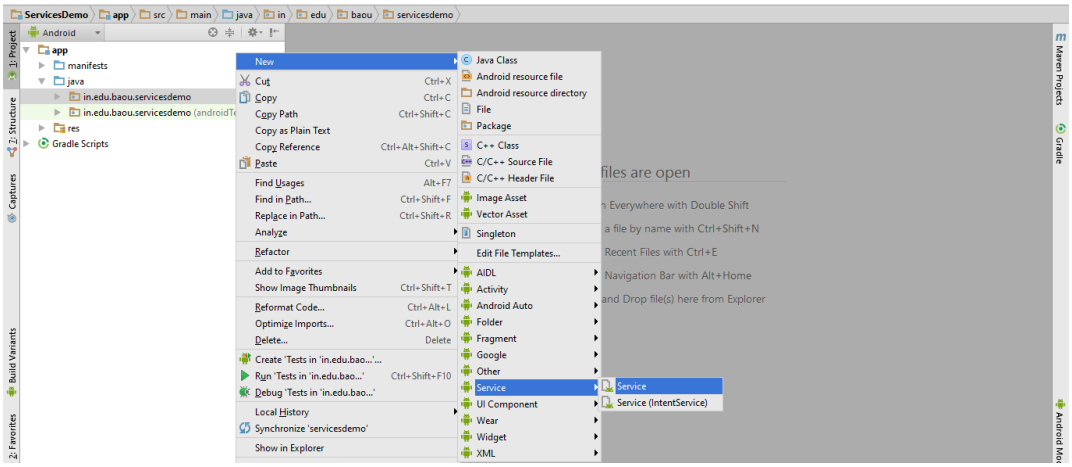

Figure-2

3. In dialog box, Enter class name as TimerService as shown in figure and press finish Button.

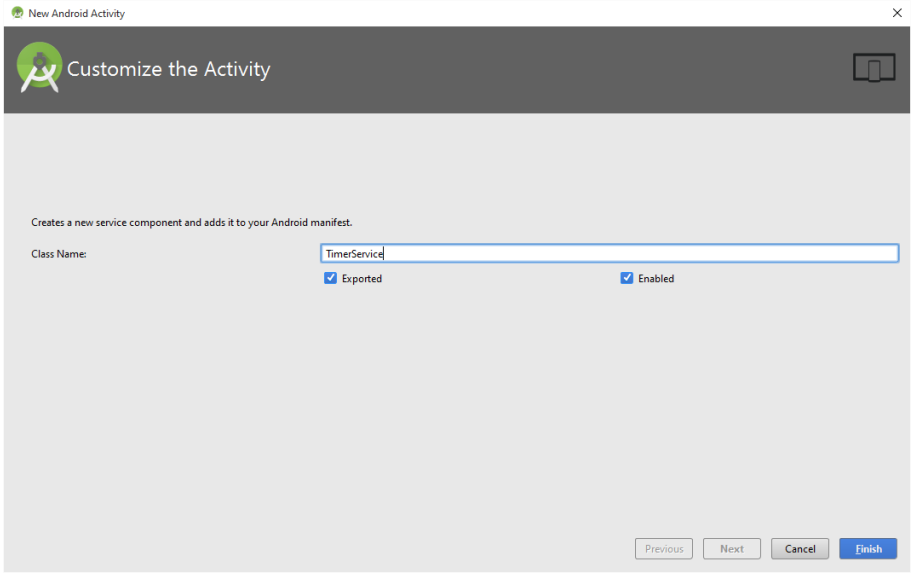

Figure-3

4. Write following code inside TimerService class

```
package in.edu.baou.servicesdemo;

import android.util.Log;
import android.widget.Toast;

import java.util.Timer;
import java.util.TimerTask;

public class TimerService extends Service {
    int counter = 0;
    Timer timer = new Timer();

    public TimerService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Toast.makeText(this, "Service Started!", Toast.LENGTH_LONG).show();
        timer.scheduleAtFixedRate(new TimerTask() {
            public void run() {
                Log.d("MyService", String.valueOf(++counter));
            }
        }, 0, 1000);
        return START_STICKY;
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (timer != null){
            timer.cancel();
        }
        Toast.makeText(this, "Service Destroyed!", Toast.LENGTH_LONG).show();
    }
}
```

5. In android_service.xml file add the following statements in bold

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"
```

```
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".ServiceActivity">

   <TextView android:text="Service Demonstration"
android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/textView" />

   <Button
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Start Timer Service"
      android:id="@+id/btnStartTimer"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_below="@+id/textView" />

   <Button
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Stop Timer Service"
      android:id="@+id/btnStopTimer"
      android:layout_below="@+id/btnStartTimer"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true" />

</RelativeLayout>
```

6. Add the following statements in bold to the ServiceActivity.java file:

```
package in.edu.baou.servicesdemo;

import android.content.Intent;
import android.view.View;
import android.widget.Button;
public class ServiceActivity extends ActionBarActivity {

   Button startTimer,stopTimer;

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

```
        startTimer = (Button)findViewById(R.id.btnStartTimer);
        stopTimer = (Button)findViewById(R.id.btnStopTimer);

        startTimer.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startService(new Intent(getBaseContext(), TimerService.class));
            }
        });

        stopTimer.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                stopService(new Intent(getBaseContext(), TimerService.class));
            }
        });
}
```

7. Press Shift+F10 or 'Run App' button in taskbar. It will launch following dialog box. Press OK.

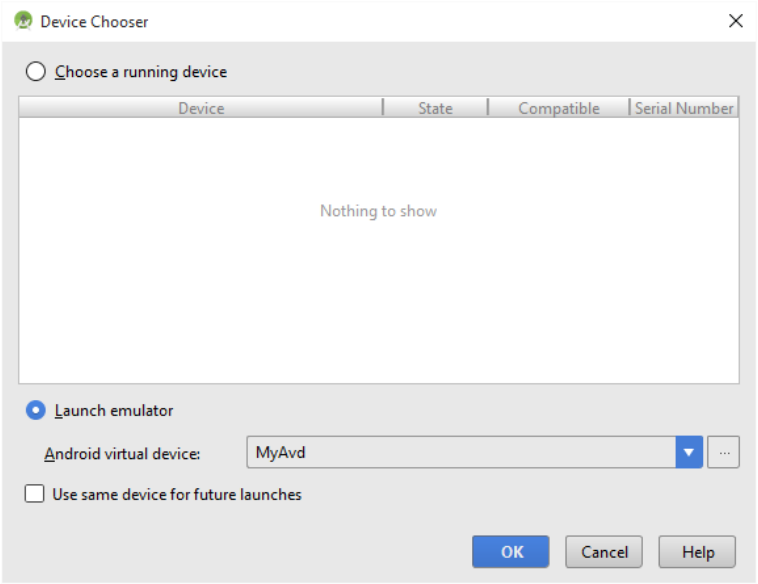

Figure-4

8. It will open activity in emulator as shown below. Clicking the **START TIMER SERVICE** button will start the service as shown below. To stop the service, click the **STOP TIMER SERVICE**.

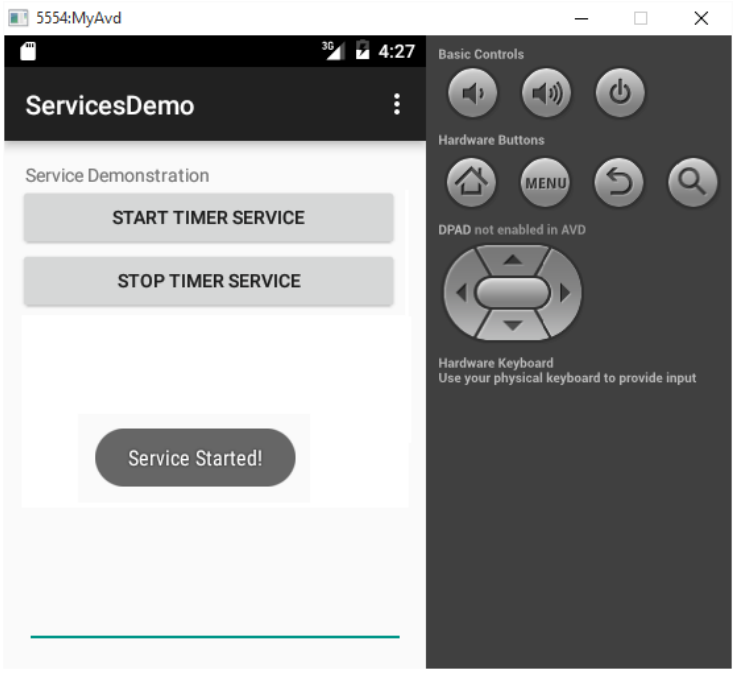

Figure-5

9. Once service is started, you can see counter value incremented by in LogCat window

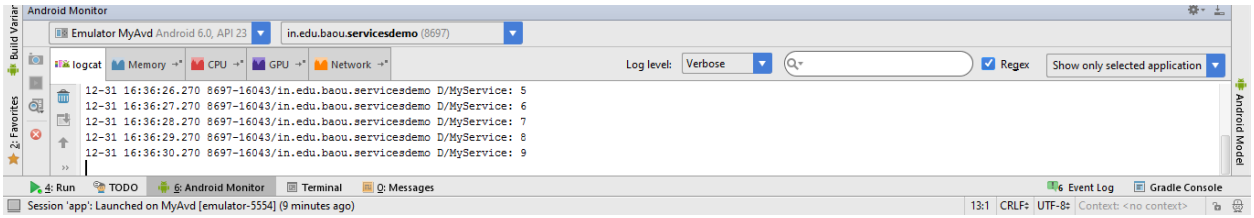

Figure-6

Explanation

- ⇨ Inside project layout file we created two buttons to start and stop service with ID btnStartTimer and btnStopTimer..
- ⇨ Inside ServiceActivity we define two button objects that represents button in layout file.
- ⇨ The findViewByID() method is used to take reference of button.
- ⇨ Button clicked event is handled by onClick() method of OnClickListener associated to button using setOnClickListener().
- ⇨ Inside TimerService class we define counter variable which initialized to zero at the start of service and increment by one every one second using Timer class scheduledAtFixedRate() Method.

- ⇨ The value of counter is logs inside LogCat window using Log.d() Method with tag “MyService”.
- ⇨ Toast is temporary message displayed on screen such as “Service Started” in step-8 and is displayed using makeText method of Toast class.

Let us sum up

In this module we have learn about service, uses of services, how to create, start and stop services. We have also discussed service life cycle and demonstrated how to create your own service.

Further Reading

1. https://developer.android.com/reference/android/app/Service

Activity

- Creating your own service

Acknowledgement: “The content in this module is modifications based on work created and shared by the Android Open-Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.”