MENU

Android Services Tutorial

View more categories:

Android Programming Tutorials

1- The types of services on Android
2- Unbounded Service
3- Bouned Service
4- IntentService service

4
Shares

1- The types of services on Android

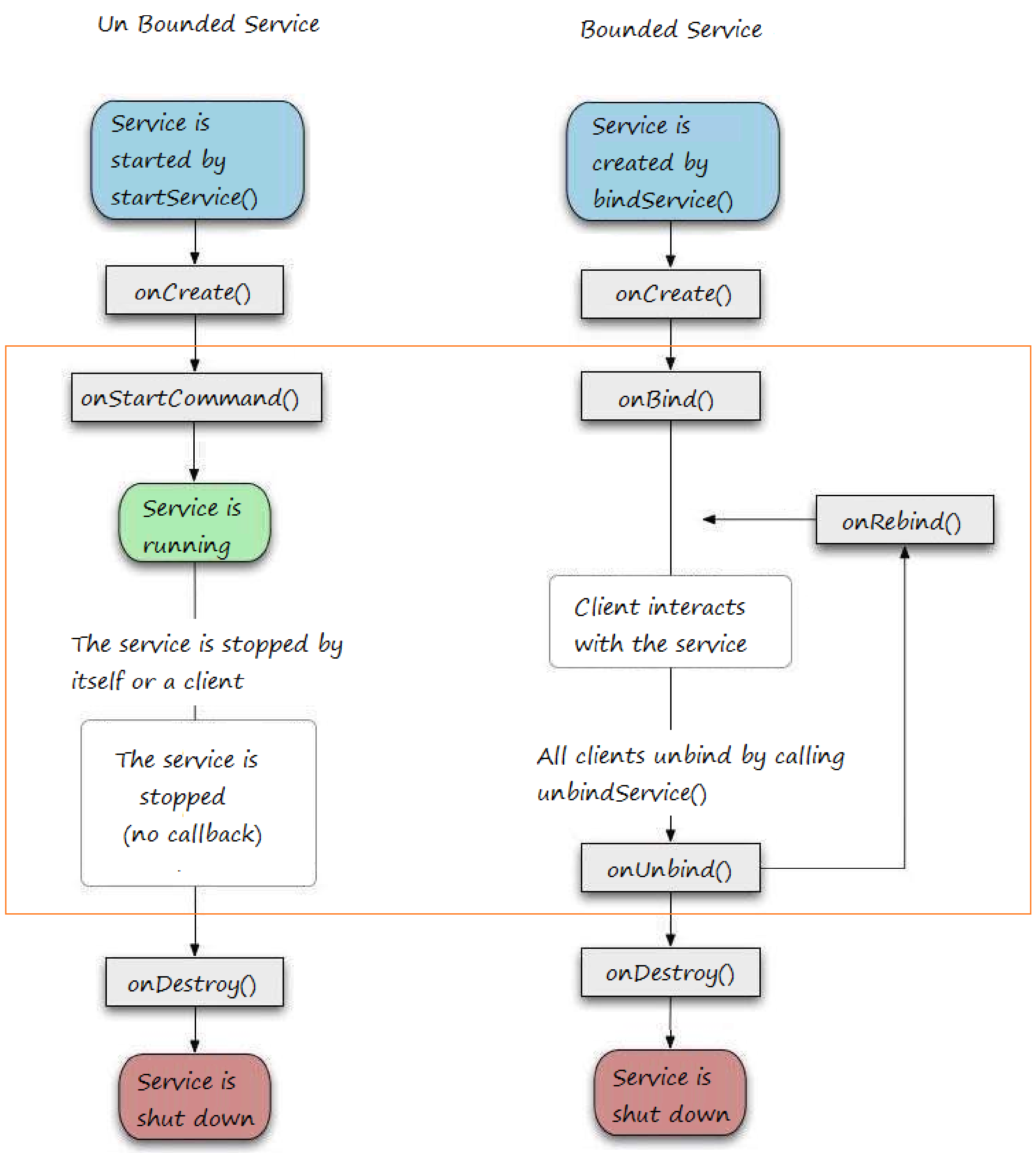
What is service?

A **service** is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed. A service can essentially take two states

State	Description
Started	A service is **started** when an application component, such as an activity, starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed. This service is also known as **Un Bounded Service**.
Bound	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, get results, and even do so across processes with interprocess communication (IPC).

> *In computer science,* ***inter-process communication (IPC)*** *is the activity of sharing data across multiple and commonly specialized processes using communication protocols. Typically, applications using IPC are categorized as clients and servers, where the client requests data and the server responds to client requests.*

A service has life cycle callback methods that you can implement to monitor changes in the service's state and you can perform work at the appropriate stage. The following diagram on the left shows the life cycle when the service is created with ***startService()*** and the diagram on the right shows the life cycle when the service is created with ***bindService()***


Un Bounded Service
Bounded Service
Service is started by startService()
Service is created by bindService()
onCreate()
onCreate()
onStartCommand()
onBind()
Service is running
onRebind()
Client interacts with the service
The service is stopped by itself or a client
The service is stopped (no callback)
All clients unbind by calling unbindService()
onUnbind()
onDestroy()
onDestroy()
Service is shut down
Service is shut down

To create an service, you create a Java class that extends the **Service** base class or one of its existing subclasses. The Service base class defines various callback methods and the most important are given below. You don't need to implement all the callbacks methods. However, it's important that you understand each one and implement those that ensure your app behaves the way users expect.

Also, there are another service called **IntentService**. **Intent Service** is used to perform one time task i.e when the task completes the service destroys itself.

Comparison of services:

Unbound Service	Bound Service	Intent Service
Unbounded Service is used to perform long repetitive task	**Bounded Service** is used to perform background task in bound with another component	**Intent Service** is used to perform one time task i.e when the task completes the service destroys itself.
Unbound Service gets starts by calling ***startService()***.	**Bounded Service** gets starts by calling ***bindService()***.	**Intent Service** gets starts by calling ***startService()***.
Unbound Service is stopped or destroyed explicitly by calling ***stopService()***.	**Bounded Service** is unbind or destroyed by calling ***unbindService().***	**IntentService** Implicitly calls ***stopself()*** to destroy
Unbound Service is independent of the component in which it is started.	**Bound Service** dependents on the component in which it is started.	**Intent Service** is independent of the component in which it is started.

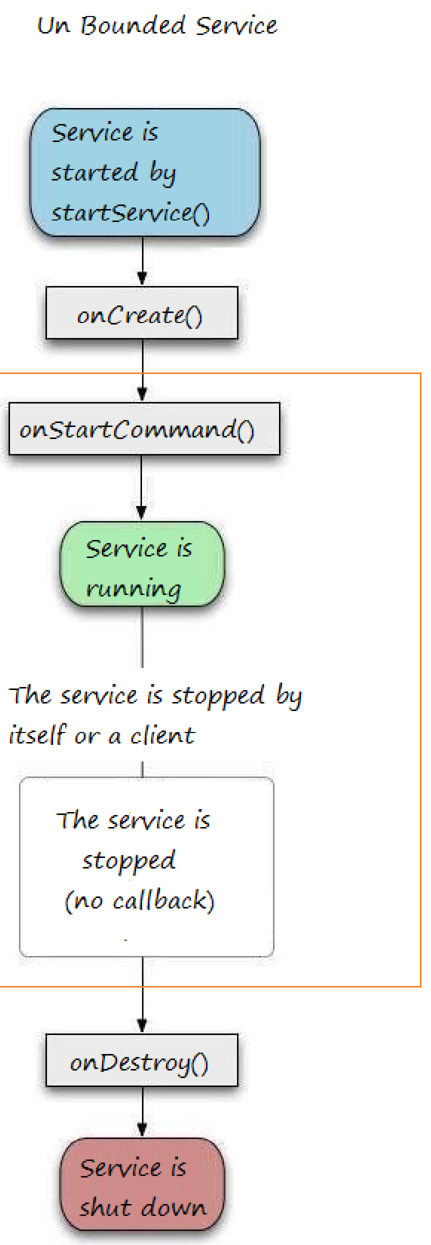
The callback methods and description:

Callback	Description
onStartCommand()	The system calls this method when another component, such as an activity, requests that the service be started, by calling ***startService()***. If you implement this method, it is your responsibility to stop the service when its work is done, by calling ***stopSelf()*** or ***stopService()*** methods.
onBind()	The system calls this method when another component wants to bind with the service by calling ***bindService()***. If you implement this

	method, you must provide an interface that clients use to communicate with the service, by returning an *IBinder* object. You must always implement this method, but if you don't want to allow binding, then you should return null.
onUnbind()	The system calls this method when all clients have disconnected from a particular interface published by the service.
onRebind()	The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its *onUnbind(Intent)*.
onCreate()	The system calls this method when the service is first created using *onStartCommand()* or *onBind()*. This call is required to perform one-time set-up.
onDestroy()	The system calls this method when the service is no longer used and is being destroyed. Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc.

2- Unbounded Service

Unbound Service (or Started Service): In this case, an application component starts the service by calling ***startService()*** , and it would continue to run in the background, even if the original component that initiated it is destroyed. For instance, when started, a service would continue to play music in the background indefinitely.


Un Bounded Service
Service is started by startService()
onCreate()
onStartCommand()
Service is running
The service is stopped by itself or a client
The service is stopped (no callback)
onDestroy()
Service is shut down

onStartCommand() method has integer return type value which can be any of the following:

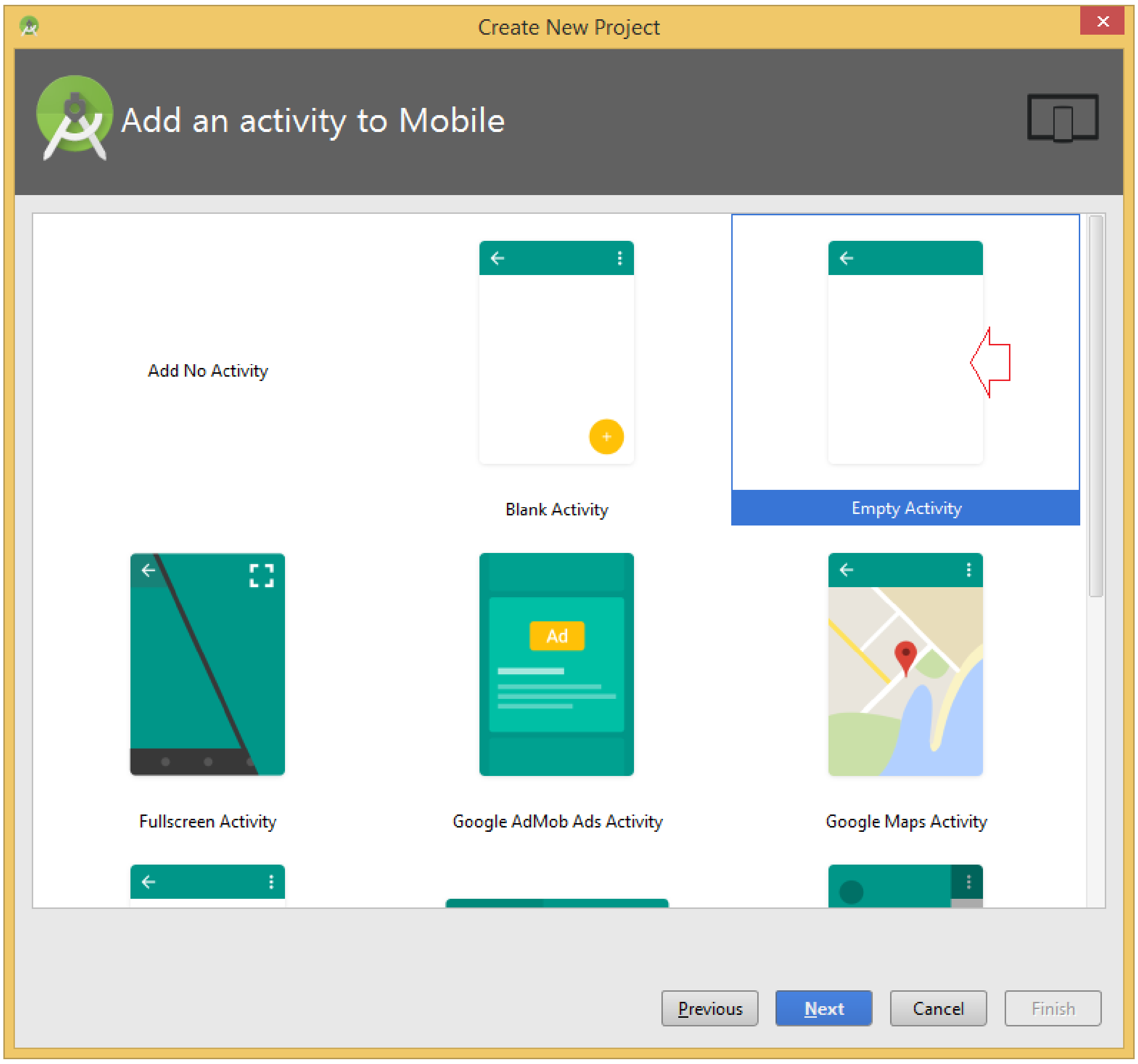
- START_STICKY

- START_NOT_STICKY
- TART_REDELIVER_INTENT

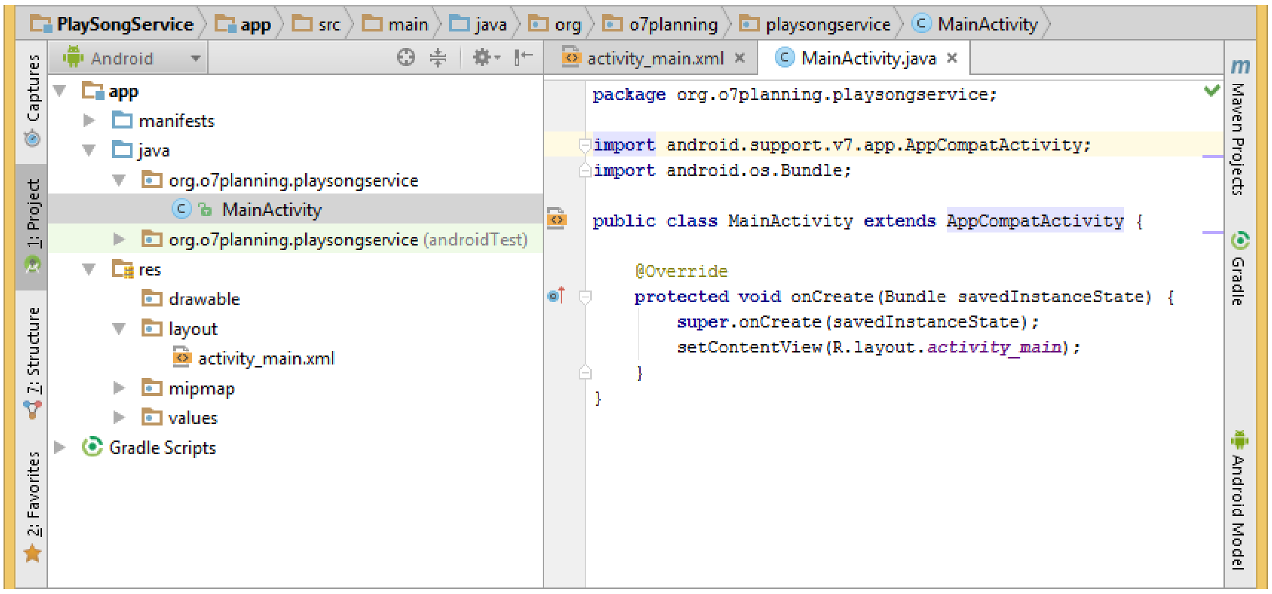
> *START_STICKY & START_NOT_STICKY*
>
> - *Both values are only relevant when the phone runs out of memory and kills the service before it finishes executing.*
> - ***START_STICKY** tells the OS to recreate the service after it has enough memory and call **onStartCommand()** again with a null intent.*
> - ***START_NOT_STICKY** tells the OS to not bother recreating the service again.*
>
> *There is also a third code **START_REDELIVER_INTENT** that tells the OS to recreate the service AND redelivery the same intent to **onStartCommand()**.*

Playing music service example (Run in background)

Create **"Empty Activity"** project with name **PlaySongService**

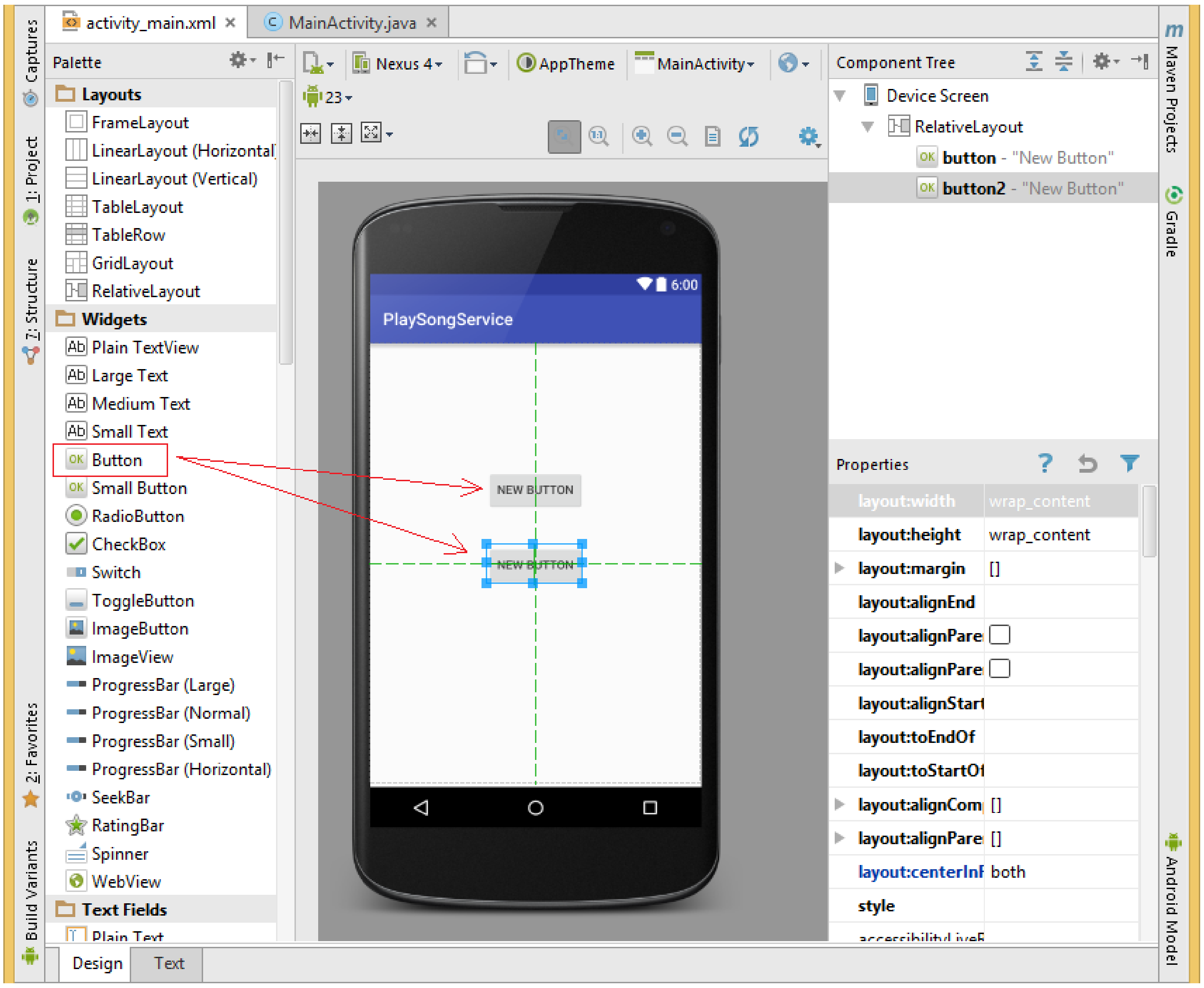


Project created.



```
package org.o7planning.playsongservice;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Drag and drop 2 buttons to the screen.



layout:width	wrap_content
layout:height	wrap_content
layout:margin	[]
layout:alignEnd	
layout:alignPare	☐
layout:alignPare	☐
layout:alignStart	
layout:toEndOf	
layout:toStartOf	
layout:alignCom	[]
layout:alignPare	[]
layout:centerInP	both
style	

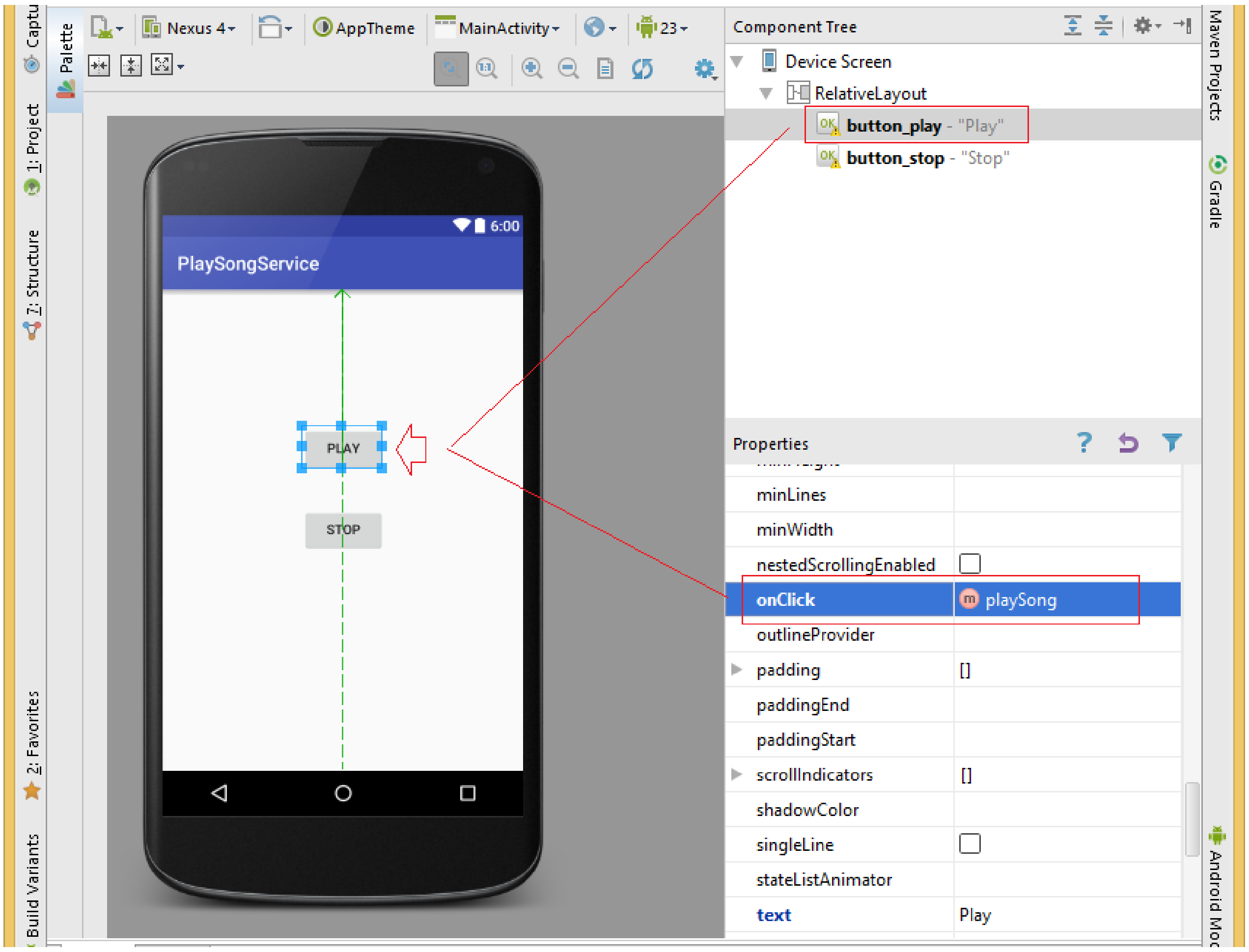
Double-click the buttons to change ID and text for buttons.

Button 1:

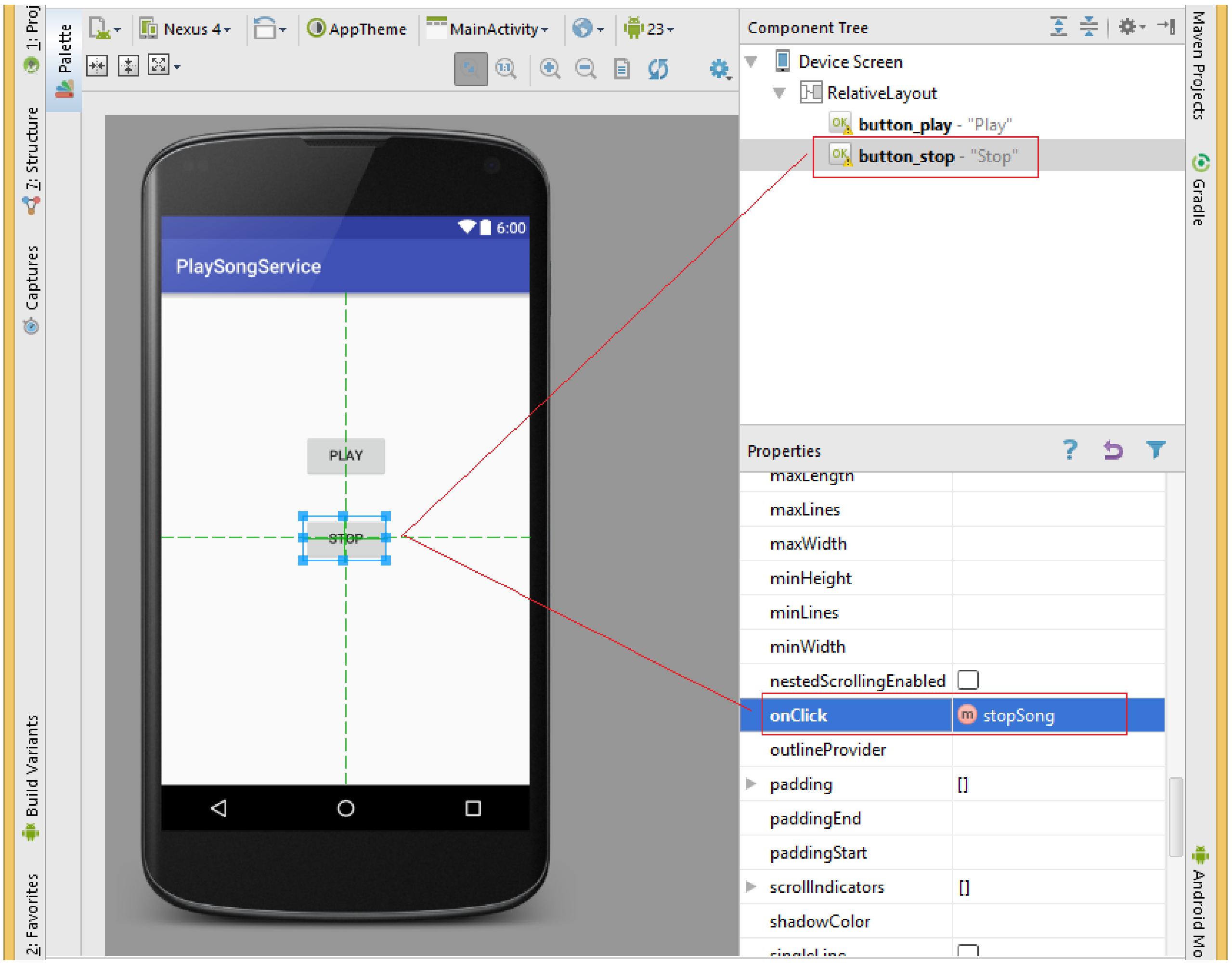
- **ID:** button_play
- **Text:** Play
- **Properties**
  - **onClick:** playSong

Button 2:

- **ID:** button_stop
- **Text:** Stop
- **Properties**
  - **onClick:** stopSong



minLines	
minWidth	
nestedScrollingEnabled	☐
onClick	playSong
outlineProvider	
padding	[]
paddingEnd	
paddingStart	
scrollIndicators	[]
shadowColor	
singleLine	☐
stateListAnimator	
text	Play



maxLength	
maxLines	
maxWidth	
minHeight	
minLines	
minWidth	
nestedScrollingEnabled	☐
onClick	stopSong
outlineProvider	
padding	[]
paddingEnd	
paddingStart	
scrollIndicators	[]
shadowColor	

activity_main.xml

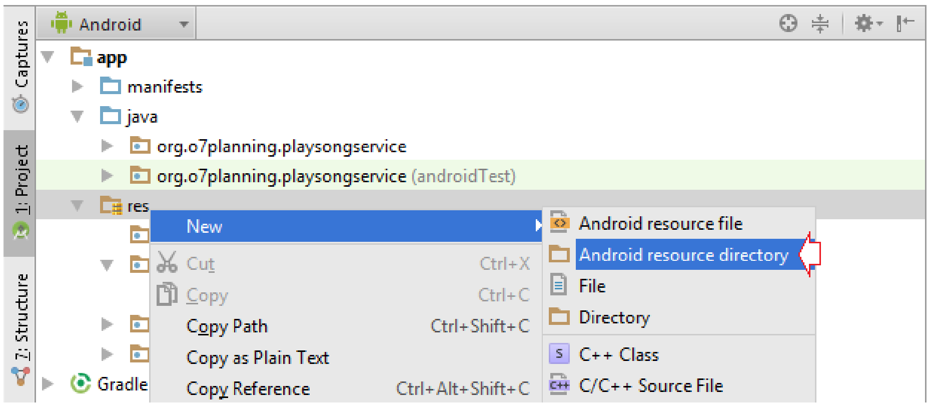
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

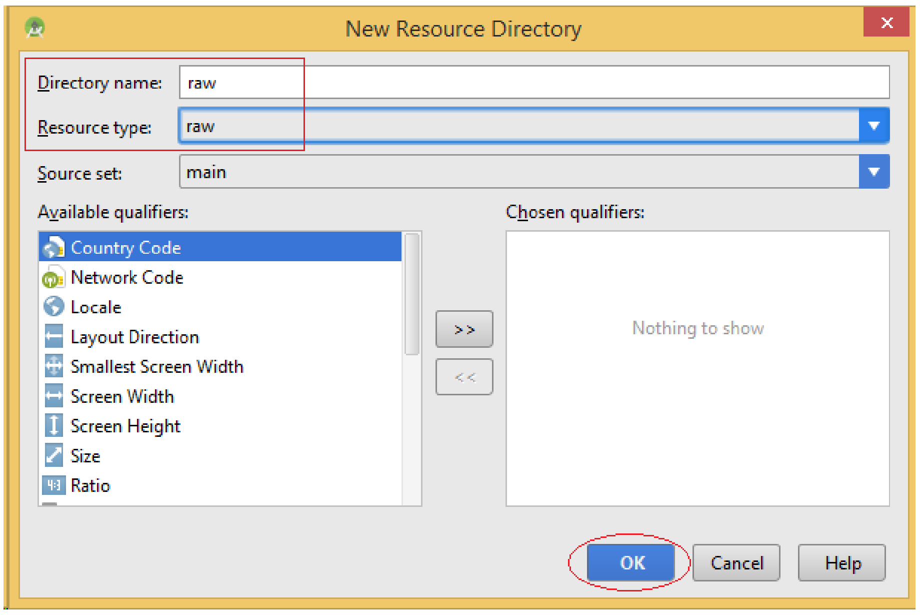
```
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Play"
        android:id="@+id/button_play"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="129dp"
        android:onClick="playSong" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Stop"
        android:id="@+id/button_stop"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:onClick="stopSong" />
</RelativeLayout>
```

Prepare mp3 file:

Right-click on the **'res'** folder, and select:

- New/Android resource directory



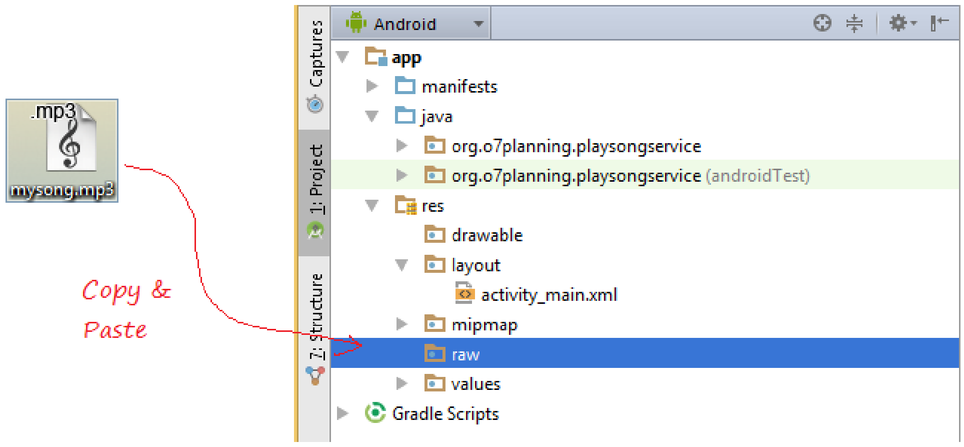


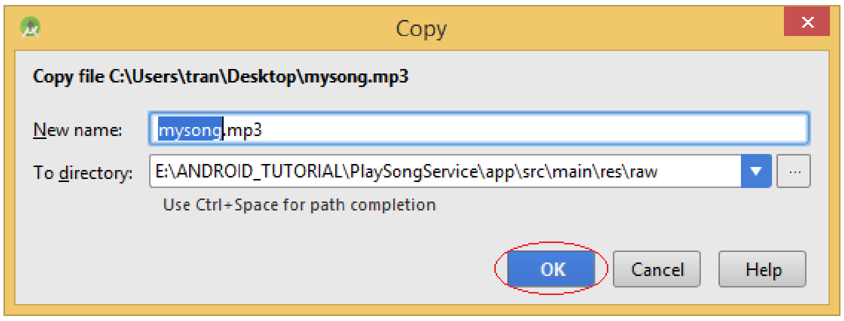
Directory name: raw

Resource type: raw

Source set: main

Copy and Paste a mp3 file to **'raw'** folder that you just have been created



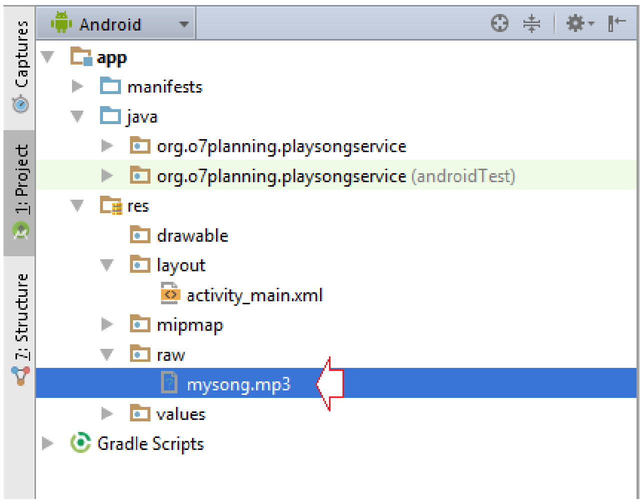


Copy file C:\Users\tran\Desktop\mysong.mp3

New name: mysong.mp3

To directory: E:\ANDROID_TUTORIAL\PlaySongService\app\src\main\res\raw

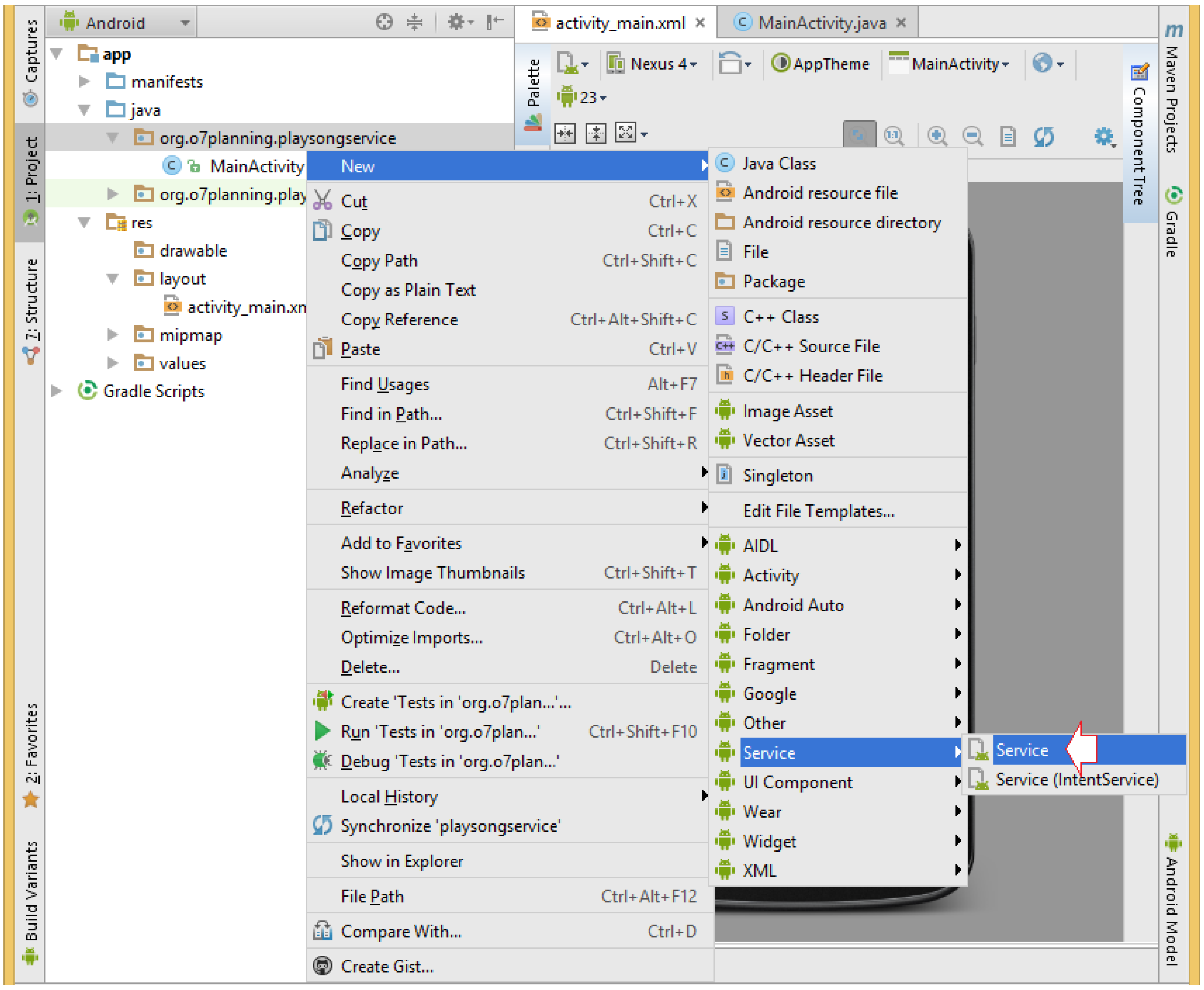
Use Ctrl+Space for path completion



Create Service class

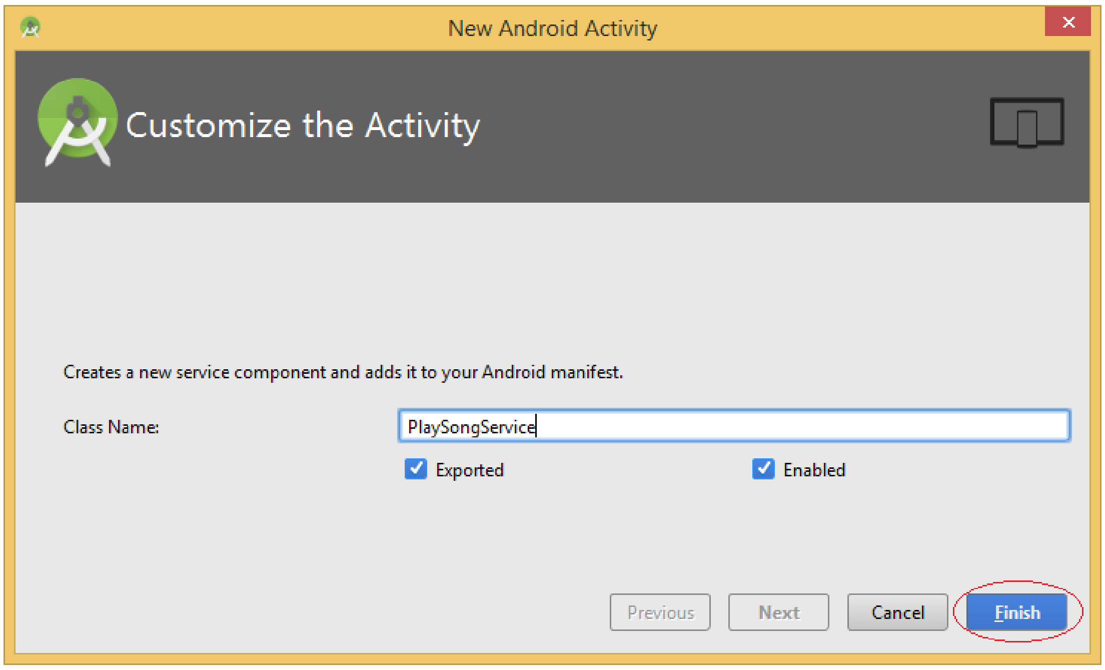
Right-click a Java package, select:

- New/Service/Service



Enter class name:

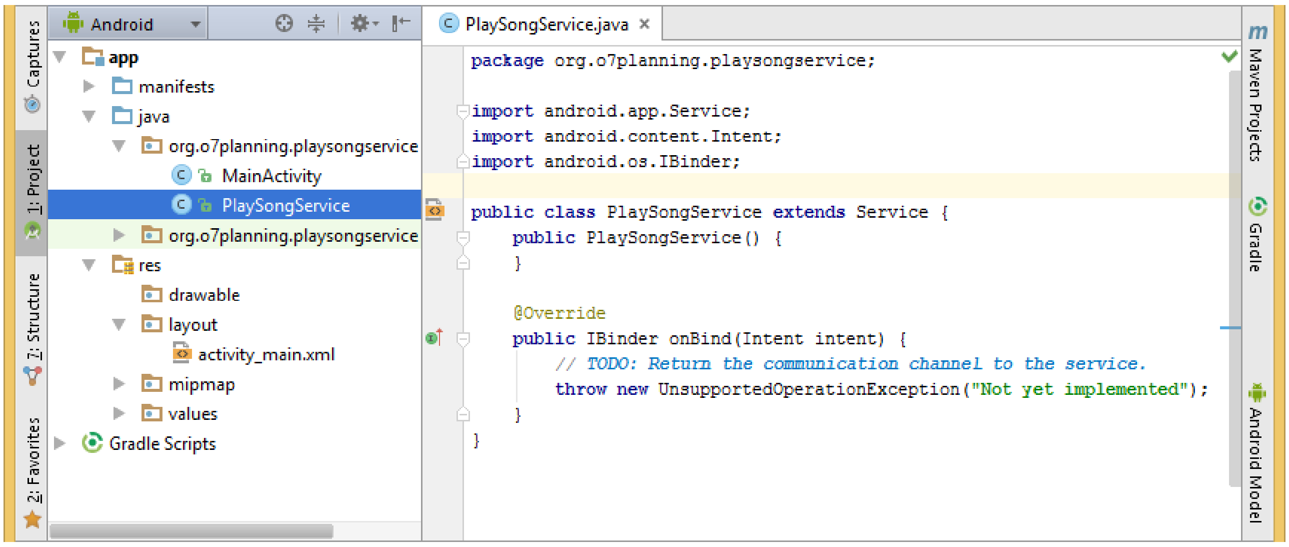
- PlaySongService



Creates a new service component and adds it to your Android manifest.

Class Name: PlaySongService

✓ Exported ✓ Enabled



```java
package org.o7planning.playsongservice;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

public class PlaySongService extends Service {
    public PlaySongService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

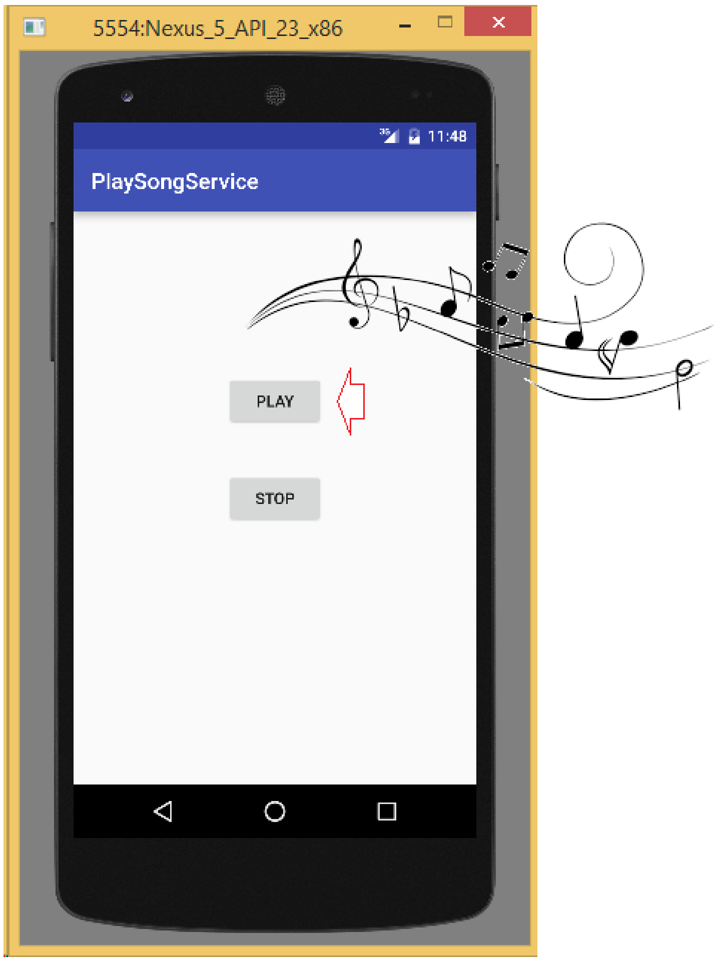
PlaySongService.java

```java
package org.o7planning.playsongservice;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;

public class PlaySongService extends Service {

    private MediaPlayer mediaPlayer;

    public PlaySongService() {
    }


    @Override
    public IBinder onBind(Intent intent){
        // This service is unbounded
        // So this method is never called.
        return null;
    }


    @Override
    public void onCreate(){
        super.onCreate();
        // Create MediaPlayer object, to play your song.
        mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.mysong);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        // Play song.
        mediaPlayer.start();

        return START_STICKY;
    }

    // Destroy
    @Override
    public void onDestroy() {
        // Release the resources
        mediaPlayer.release();
        super.onDestroy();
    }
}
```

MainActivity.java

```java
package org.o7planning.playsongservice;

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);
    }

    // This method is called when users click on the Start button.
    public void playSong(View view)  {
        // Create Intent object for PlaySongService.
        Intent myIntent = new Intent(MainActivity.this, PlaySongService.class);

        // Call startService with Intent parameter.
        this.startService(myIntent);
    }

    // This method is called when users click on the Stop button.
    public void stopSong(View view)  {

        // Create Intent object
        Intent myIntent = new Intent(MainActivity.this, PlaySongService.class);
        this.stopService(myIntent);
    }
}
```

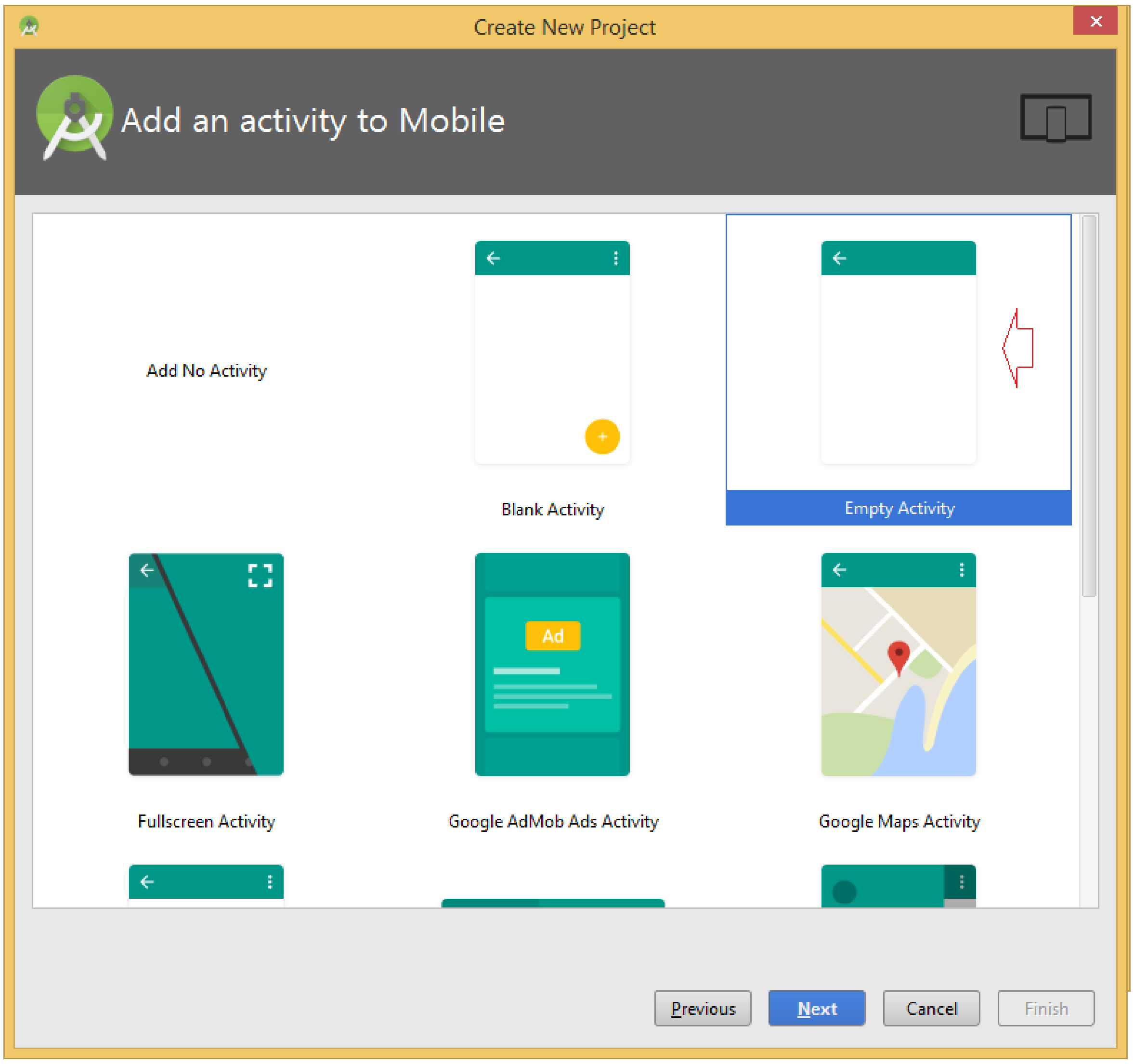
That's OK, you can run your application and enjoy the song.

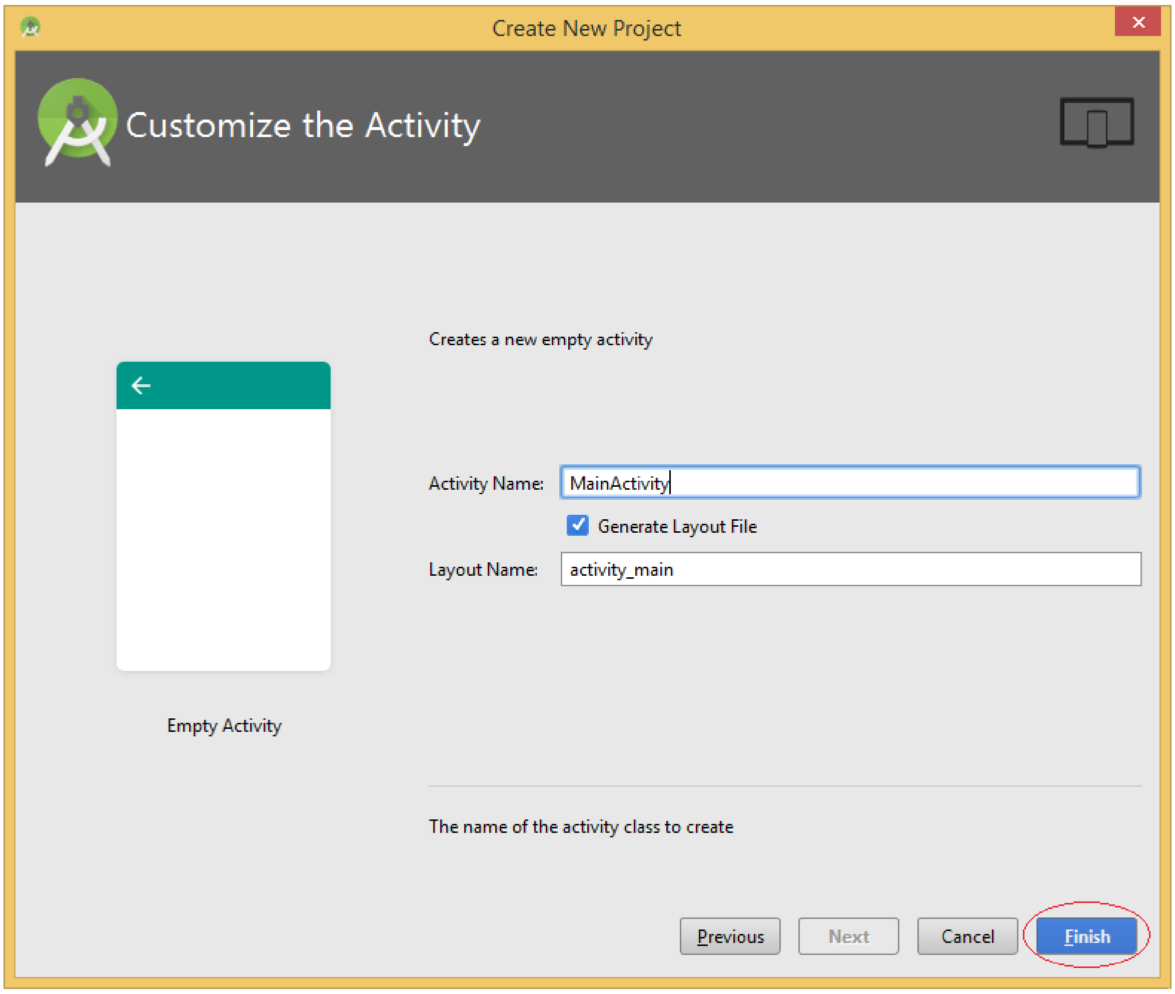


3- Bouned Service

Hereinafter, I simulate a serivce that provides weather information current day, with the input is geographical location (Hanoi, Chicago, ...), the result returned is rainy, sunny, ...

Create project named **WeatherService**.


Create New Project
Add an activity to Mobile
Add No Activity
Blank Activity
Empty Activity
Ad
Fullscreen Activity
Google AdMob Ads Activity
Google Maps Activity
Previous
Next
Cancel
Finish



Create New Project

Customize the Activity

Creates a new empty activity

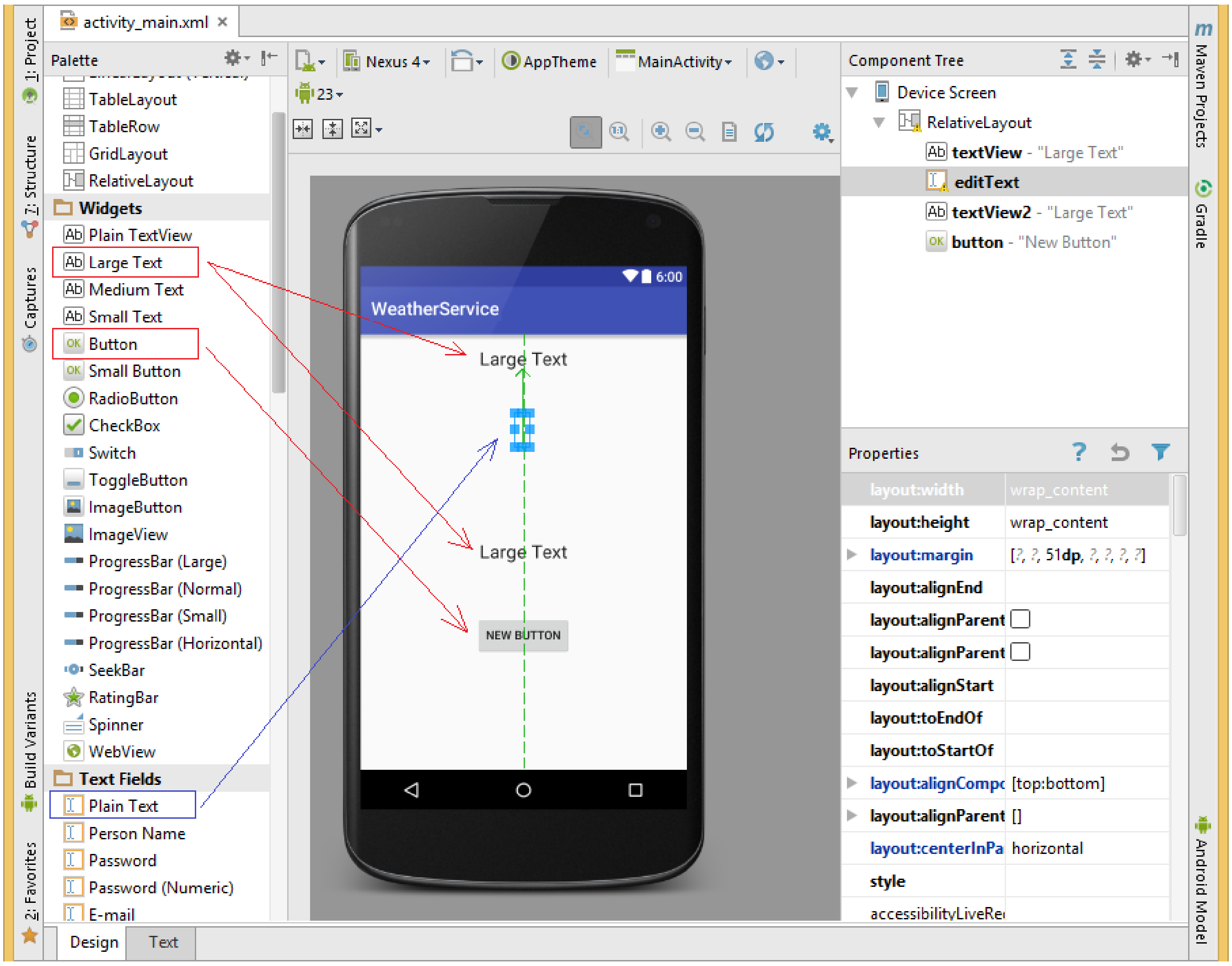
Activity Name: MainActivity

✓ Generate Layout File

Layout Name: activity_main

The name of the activity class to create

Drag and drop some widgets to the screen.



layout:width	wrap_content
layout:height	wrap_content
layout:margin	[?, ?, 51dp, ?, ?, ?, ?]
layout:alignEnd	
layout:alignParent	☐
layout:alignParent	☐
layout:alignStart	
layout:toEndOf	
layout:toStartOf	
layout:alignCompo	[top:bottom]
layout:alignParent	[]
layout:centerInPa	horizontal
style	
accessibilityLiveRe	

By double-clicking on the widget, you can set **text** and **ID** for it:

TextView 1:

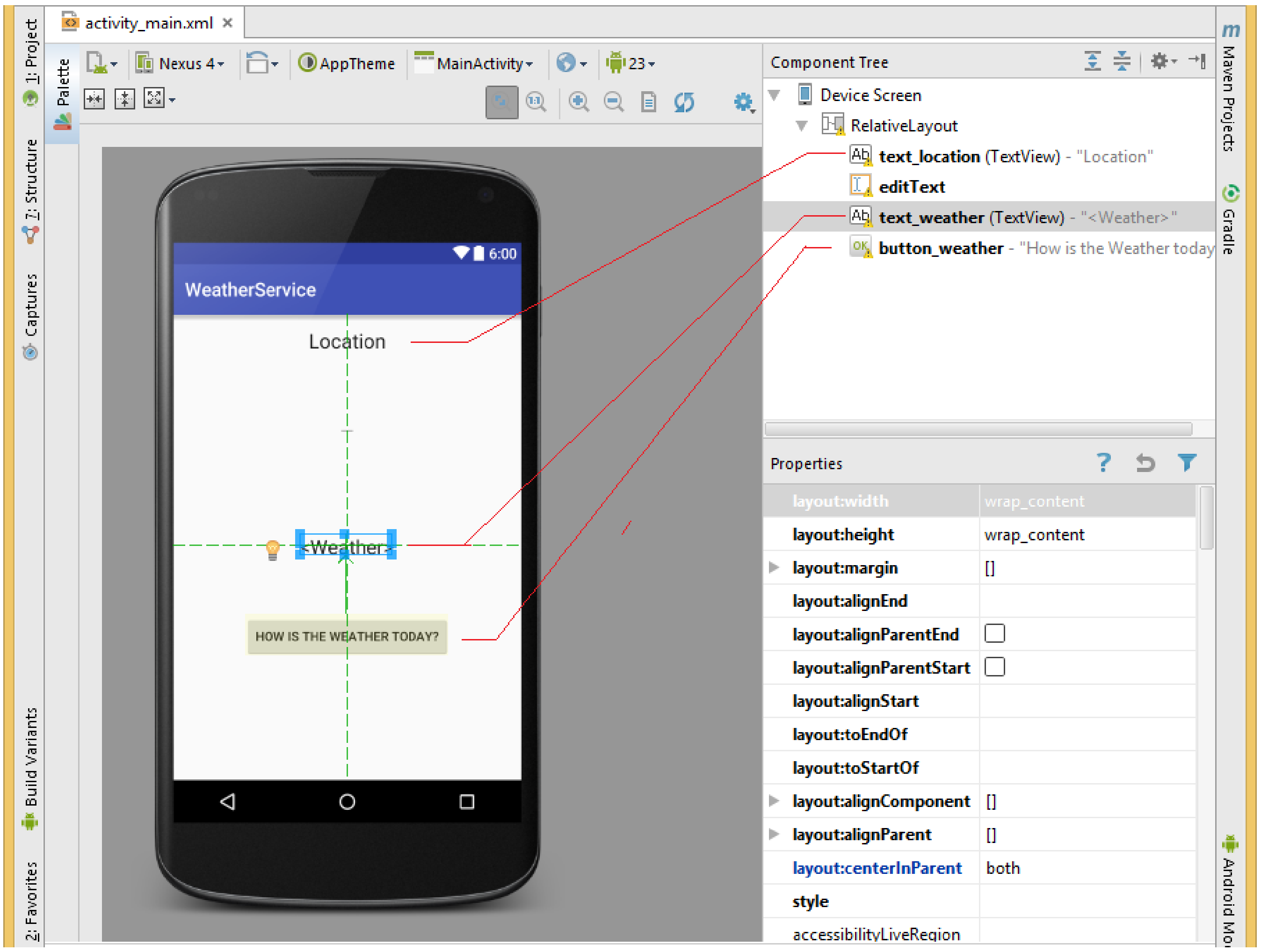
- **ID:** text_location
- **Text:** Location

TextView 2:

- **ID:** text_weather
- **Text:** <Weather>

Button:

- **ID:** button_weather
- **Text:** How is the Weather today?



Component Tree

- Device Screen
  - RelativeLayout
    - **text_location** (TextView) - "Location"
    - **editText**
    - **text_weather** (TextView) - "<Weather>"
    - **button_weather** - "How is the Weather today

Properties

layout:width	wrap_content
layout:height	wrap_content
layout:margin	[]
layout:alignEnd	
layout:alignParentEnd	☐
layout:alignParentStart	☐
layout:alignStart	
layout:toEndOf	
layout:toStartOf	
layout:alignComponent	[]
layout:alignParent	[]
layout:centerInParent	both
style	
accessibilityLiveRegion	

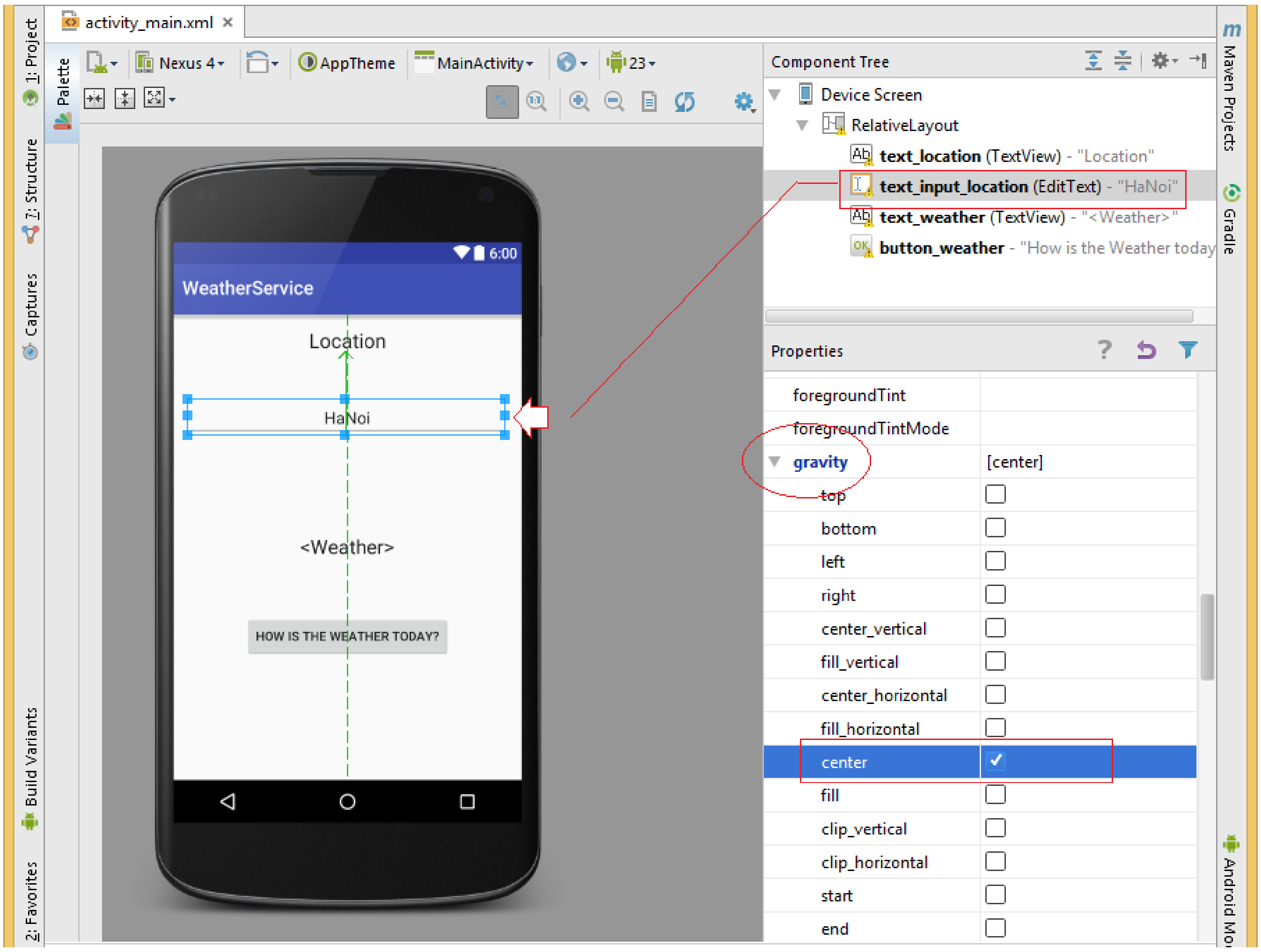
Set ID and text for **EditText** object:

EditText

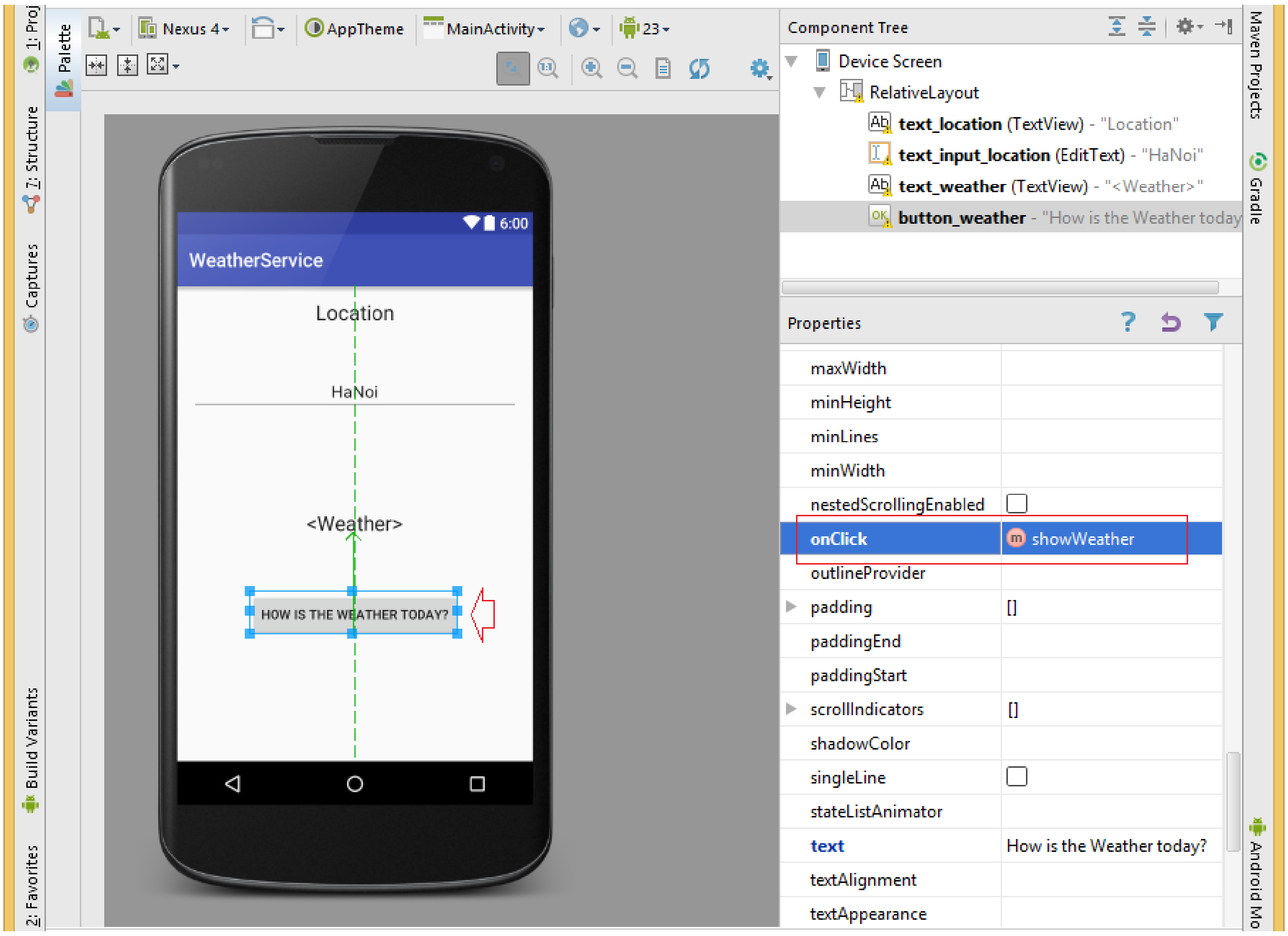
- **ID:** text_input_location
- **Text:** Hanoi

Properties

- **layout:width:** fill_parent
- **gravity**
  - **center:** checked



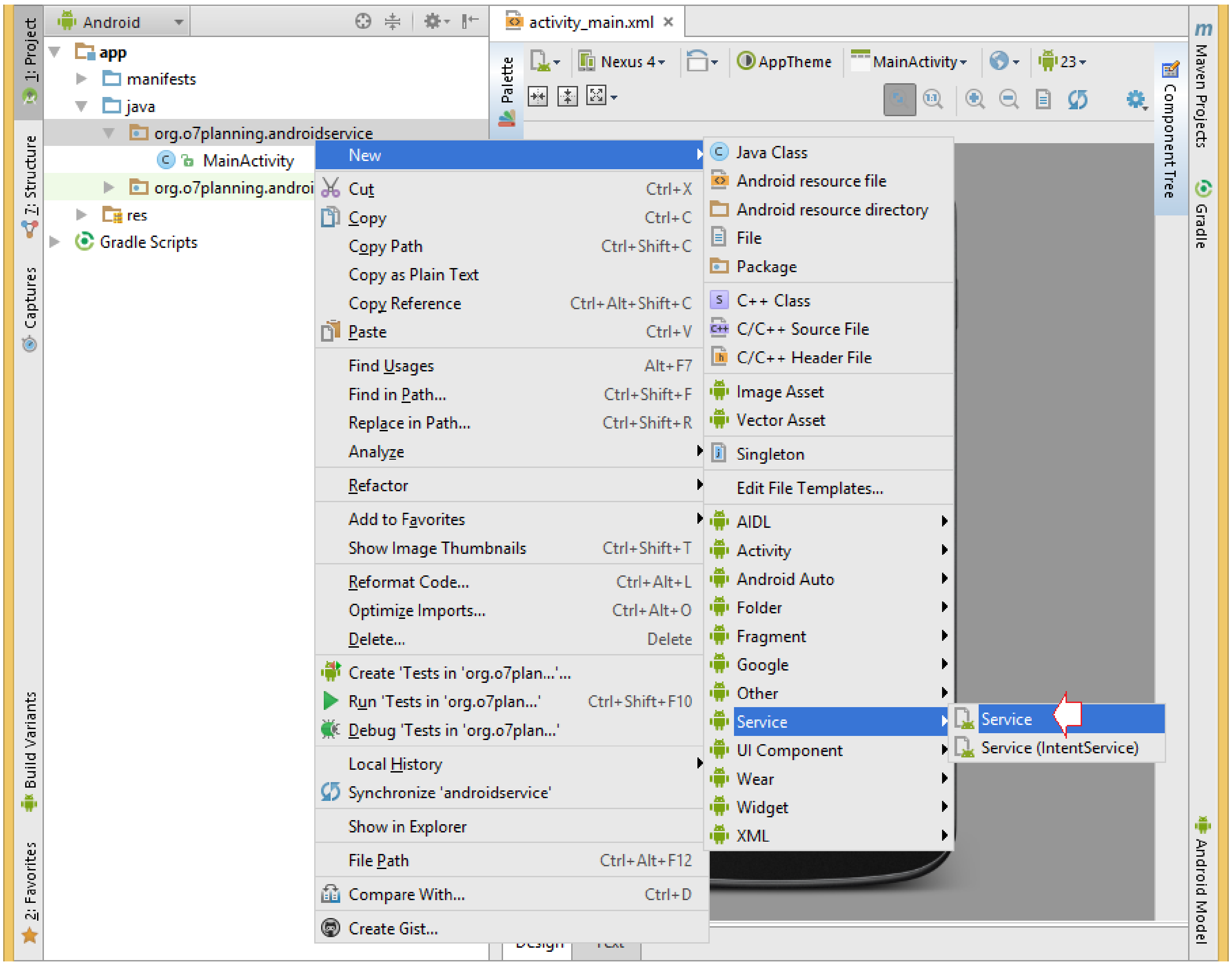
Set *onClick* attribute for button is *showWeather* which means that when clicking button, the *showWeather* method will be called. We will write this method thereinafter.



Create Service:

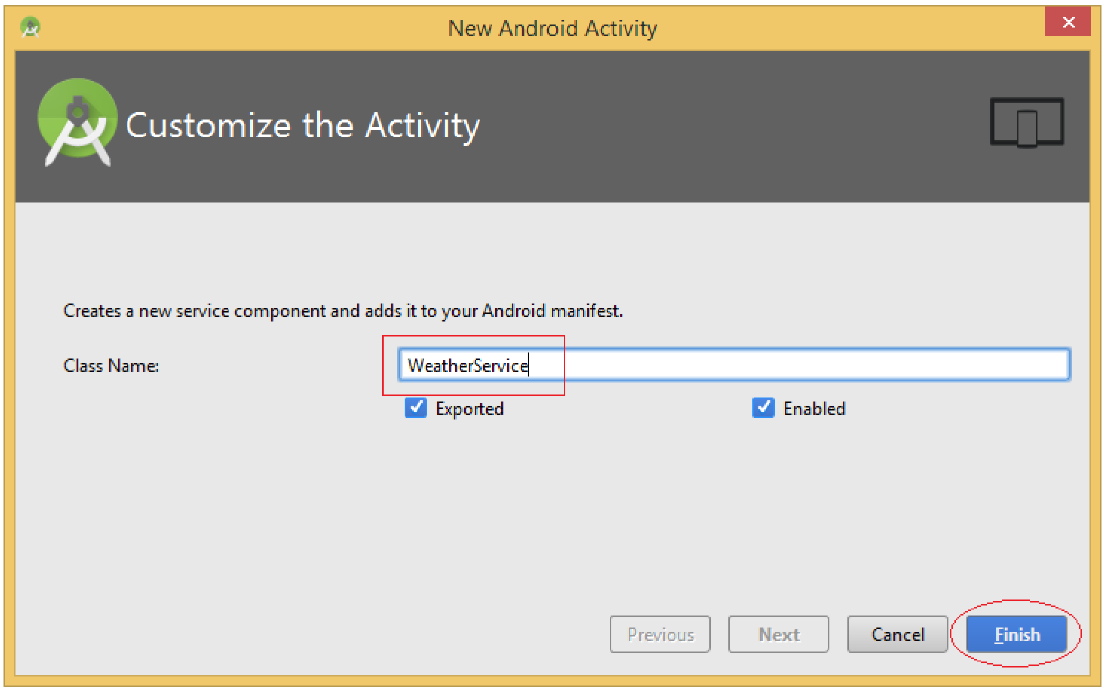
Right-click a **Java package**, select:

- **New/Service/Service**



Enter:

- **Class name:** WeatherService

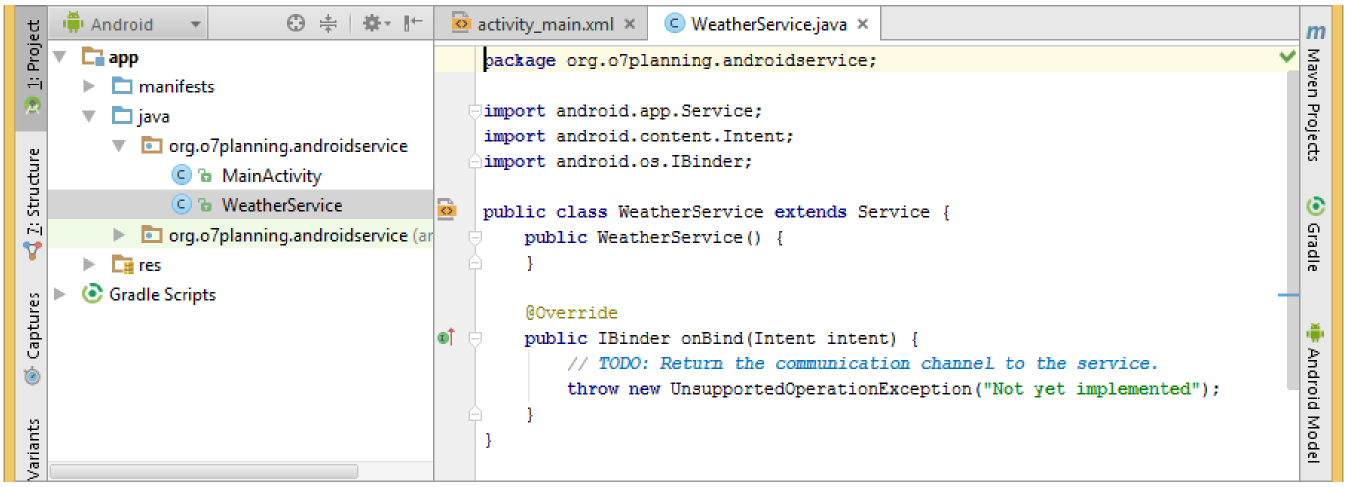


Creates a new service component and adds it to your Android manifest.

Class Name: WeatherService

☑ Exported ☑ Enabled

WeatherService class which is extended from **android.app.Service** class.has been created.



```
package org.o7planning.androidservice;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

public class WeatherService extends Service {
    public WeatherService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

WeatherService.java

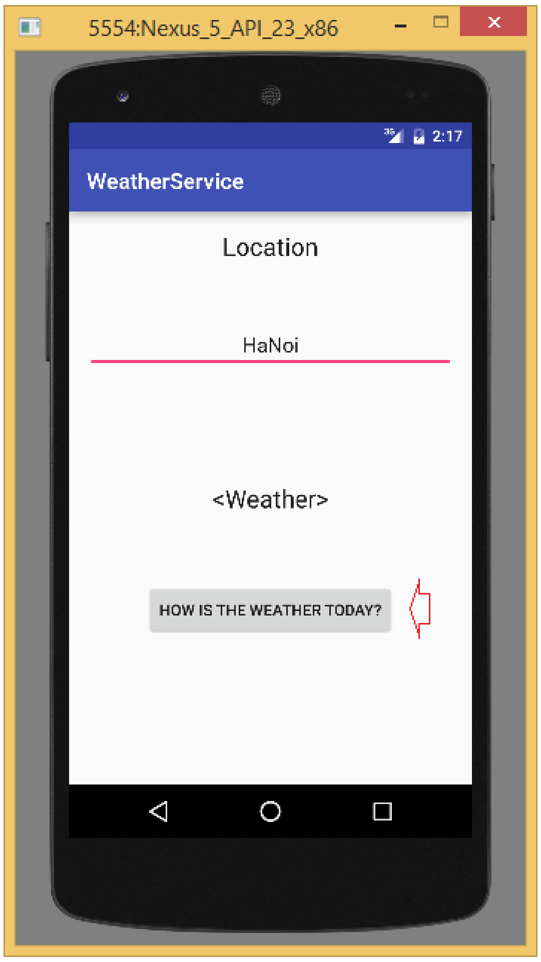
```
package org.o7planning.weatherservice;

import android.app.Service;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.util.Log;


import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;


public class WeatherService extends Service {

  private static String LOG_TAG = "WeatherService";

  // Store the weather data.
  private static final Map<String, String> weatherData = new HashMap<String,String>();

  private final IBinder binder = new LocalWeatherBinder();

  public class LocalWeatherBinder extends Binder {

      public WeatherService getService() {
          return WeatherService.this;
      }
  }

  public WeatherService() {
  }

  @Override
  public IBinder onBind(Intent intent) {
      Log.i(LOG_TAG,"onBind");
      return this.binder;
  }

  @Override
  public void onRebind(Intent intent) {
      Log.i(LOG_TAG, "onRebind");
      super.onRebind(intent);
  }

  @Override
  public boolean onUnbind(Intent intent) {
      Log.i(LOG_TAG, "onUnbind");
      return true;
  }

  @Override
  public void onDestroy() {
      super.onDestroy();
      Log.i(LOG_TAG, "onDestroy");
  }

  // Returns the weather information corresponding to the location of the current date.
  public String getWeatherToday(String location) {
      Date now= new Date();
      DateFormat df= new SimpleDateFormat("dd-MM-yyyy");

      String dayString = df.format(now);
      String keyLocAndDay = location + "$"+ dayString;

      String weather=  weatherData.get(keyLocAndDay);
      //
      if(weather != null)  {
          return weather;
      }

      //
      String[] weathers = new String[]{"Rainy", "Hot", "Cool", "Warm" ,"Snowy"};

      // Random value from 0 to 4
      int i= new Random().nextInt(5);

      weather =weathers[i];
      weatherData.put(keyLocAndDay, weather);
```

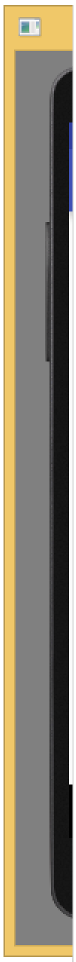
```
        //
        return weather;
    }



}
```

MainActivity.java

```
package org.o7planning.weatherservice;

import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.IBinder;
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 {


    private boolean binded=false;
    private WeatherService weatherService;

    private TextView weatherText;
    private EditText locationText;

    ServiceConnection weatherServiceConnection = new ServiceConnection() {

        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            WeatherService.LocalWeatherBinder binder = (WeatherService.LocalWeatherBinder) service;
            weatherService = binder.getService();
            binded = true;
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            binded = false;
        }
    };

    // When the Activity creating its interface.
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        weatherText = (TextView) this.findViewById(R.id.text_weather);
        locationText = (EditText)this.findViewById(R.id.text_input_location);
    }

    // When Activity starting.
    @Override
    protected void onStart() {
        super.onStart();

        // Create Intent object for WeatherService.
        Intent intent = new Intent(this, WeatherService.class);

        // Call bindService(..) method to bind service with UI.
        this.bindService(intent, weatherServiceConnection, Context.BIND_AUTO_CREATE);
    }

    // Activity stop
    @Override
    protected void onStop() {
        super.onStop();
        if (binded) {
            // Unbind Service
            this.unbindService(weatherServiceConnection);
            binded = false;
        }
    }

    // When user click on 'see weather' button.
    public void showWeather(View view)  {
        String location = locationText.getText().toString();

        String weather= this.weatherService.getWeatherToday(location);

        weatherText.setText(weather);
    }

}
```

OK, now you can run the application.

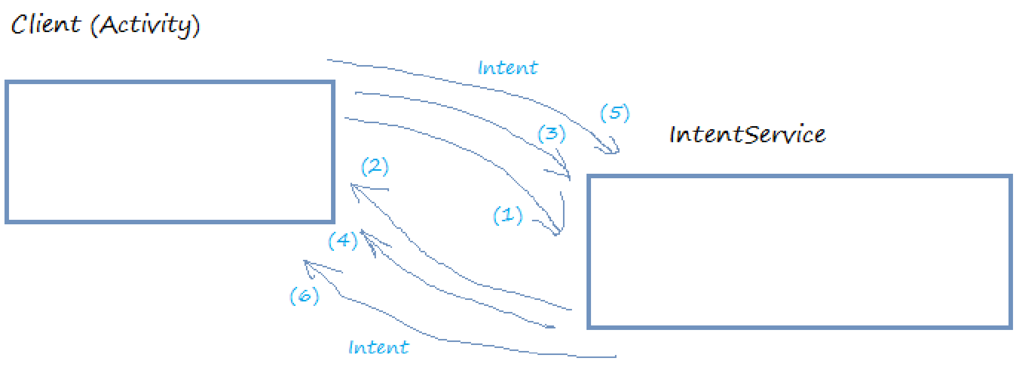




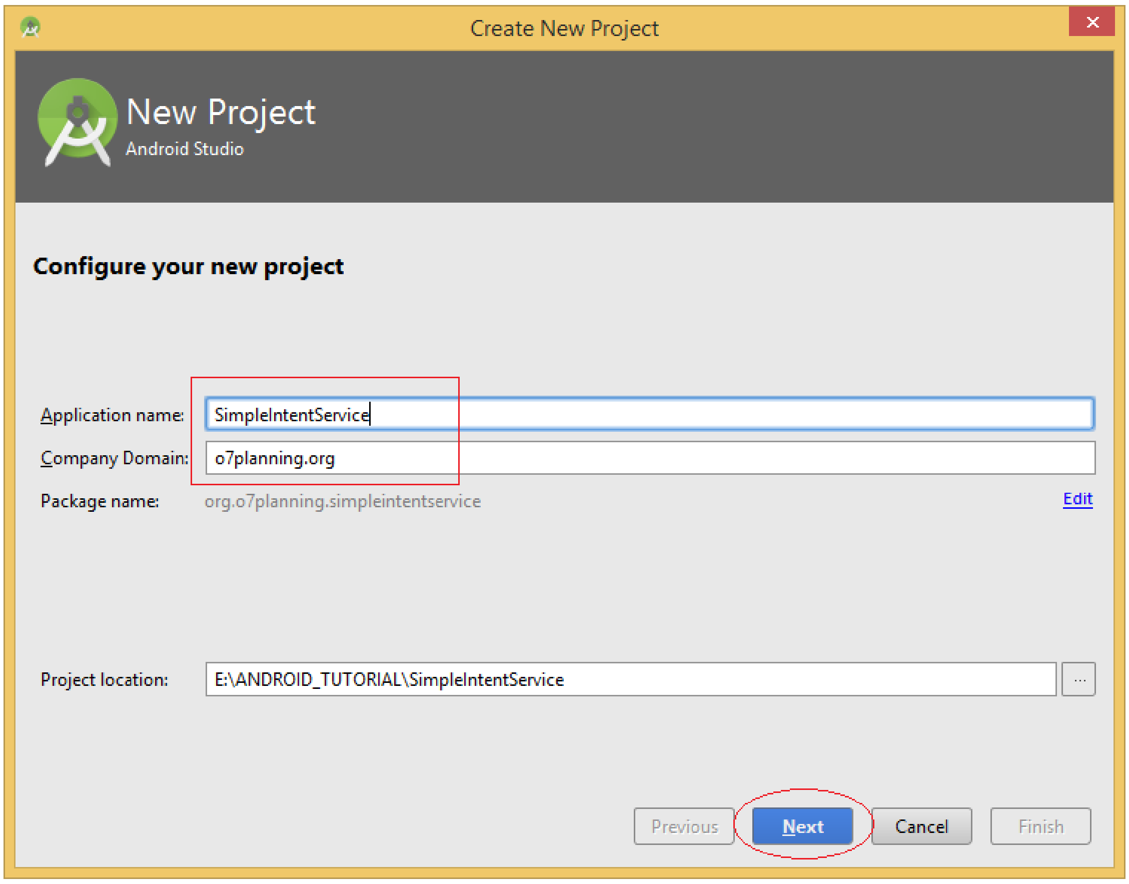
4- IntentService service

IntentService example:

The below imagine illustrates the communication between Client (**Activity**) and **IntentService**, Client start the service, it sends request through an **Intent** object, the service is run and do their duties, at the same time, it can send information relating to its working situation, for example, how many percentage does it work. At client, you can use **ProgressBar** to display the percentage of work.



Create **SimpleIntentService** project.

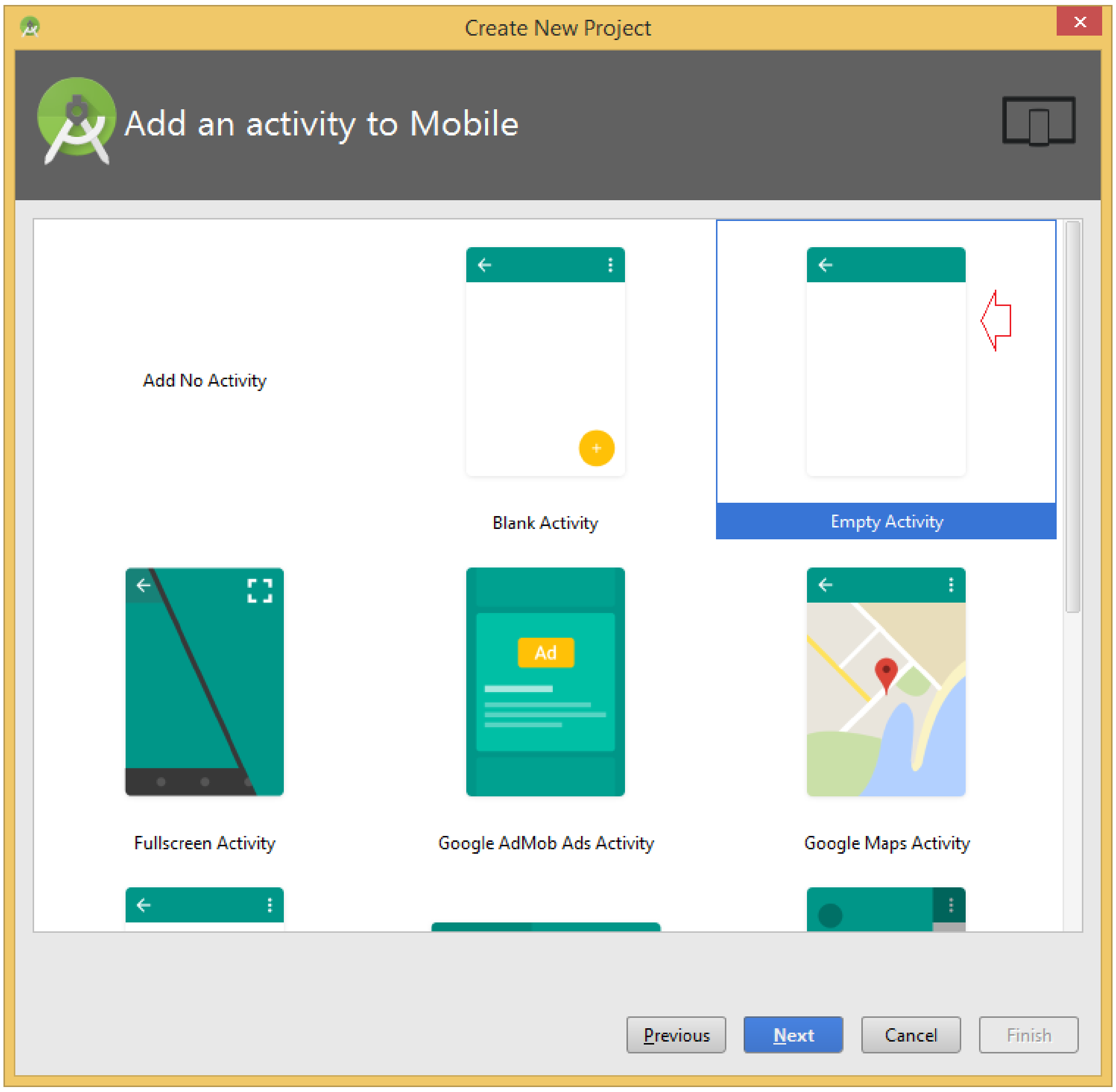

Create New Project

New Project
Android Studio

Configure your new project

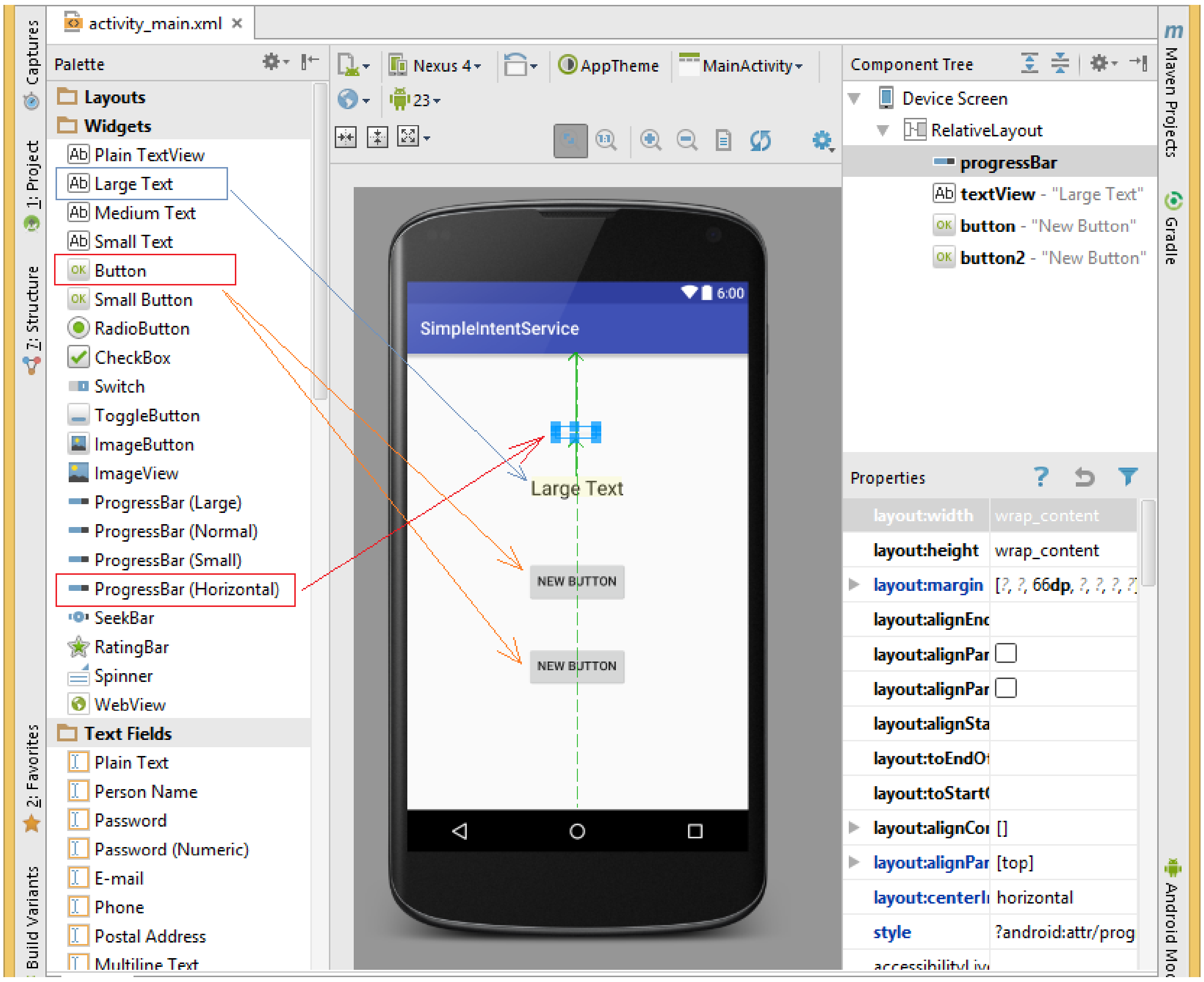
Application name: SimpleIntentService
Company Domain: o7planning.org
Package name: org.o7planning.simpleintentservice

Project location: E:\ANDROID_TUTORIAL\SimpleIntentService


Create New Project

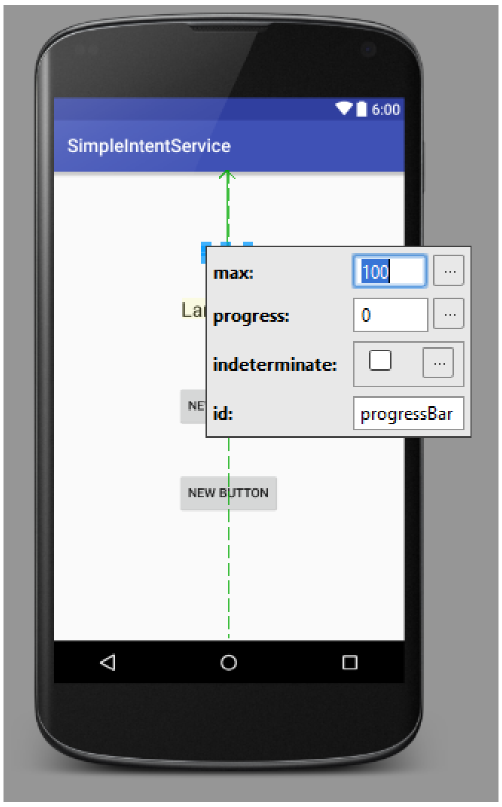
Add an activity to Mobile

Drag and drop some of the components to UI:



layout:width	wrap_content
layout:height	wrap_content
layout:margin	[?, ?, 66dp, ?, ?, ?, ?
layout:alignEnd	
layout:alignPar	☐
layout:alignPar	☐
layout:alignSta	
layout:toEndOf	
layout:toStart	
layout:alignCo	[]
layout:alignPar	[top]
layout:centerI	horizontal
style	?android:attr/prog

Double-click on **ProgressBar** to change ID and its values.



max:	100
progress:	0
indeterminate:	☐
id:	progressBar

Set ID and text for components on interfaces.

ProgressBar:

- **ID:** progressBar
- Properties:
  - **layout:width:** fill_parent

TextView

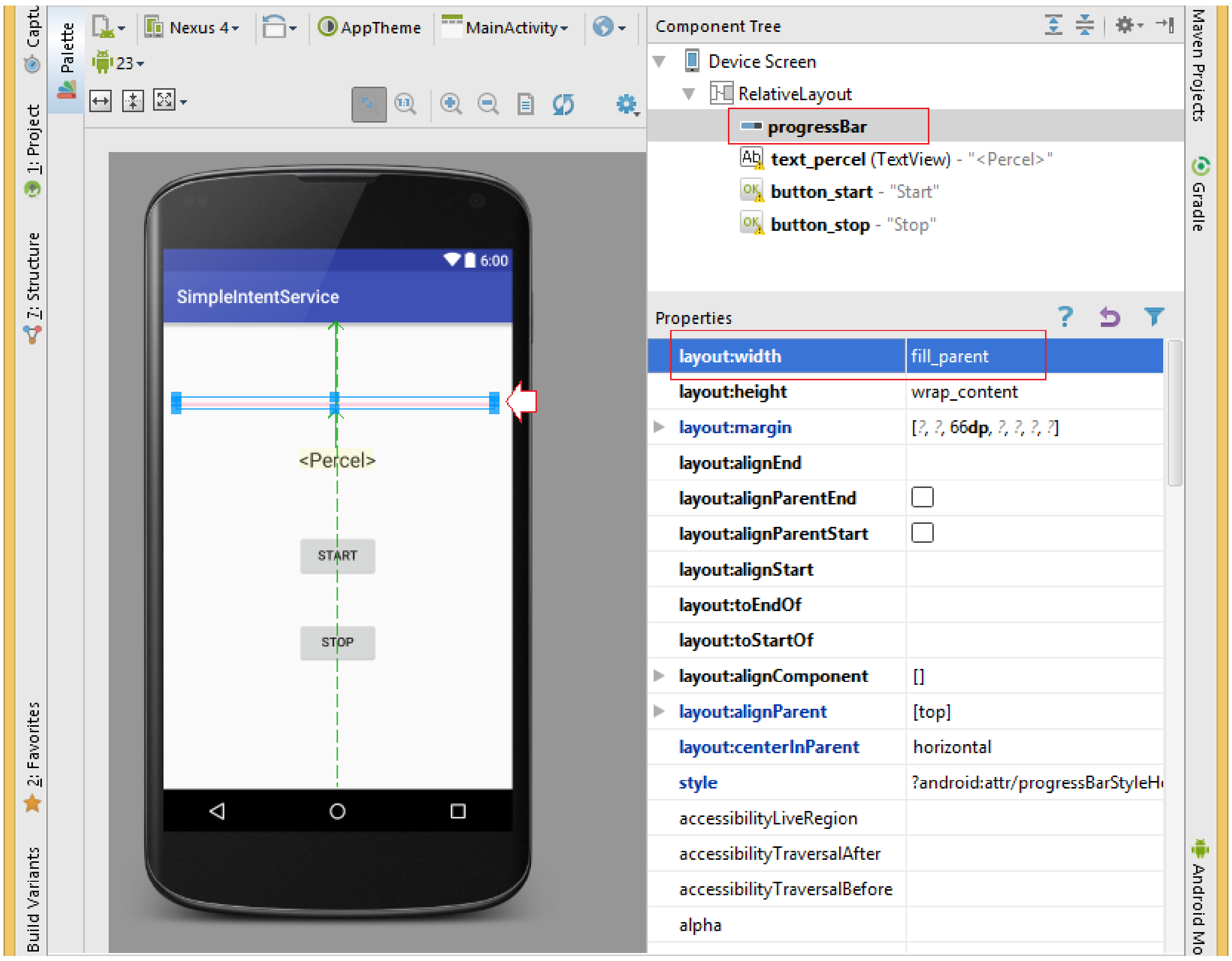
- **ID:** text_percel
- **Text:** <Percel>

Button 1

- **ID:** button_start
- **Text:** Start
- **Properties**
  - **onClick:** startButtonClicked

Button 2

- **ID:** button_stop
- **Text:** Stop
- **Properties**
  - **onClick:** stopButtonClicked



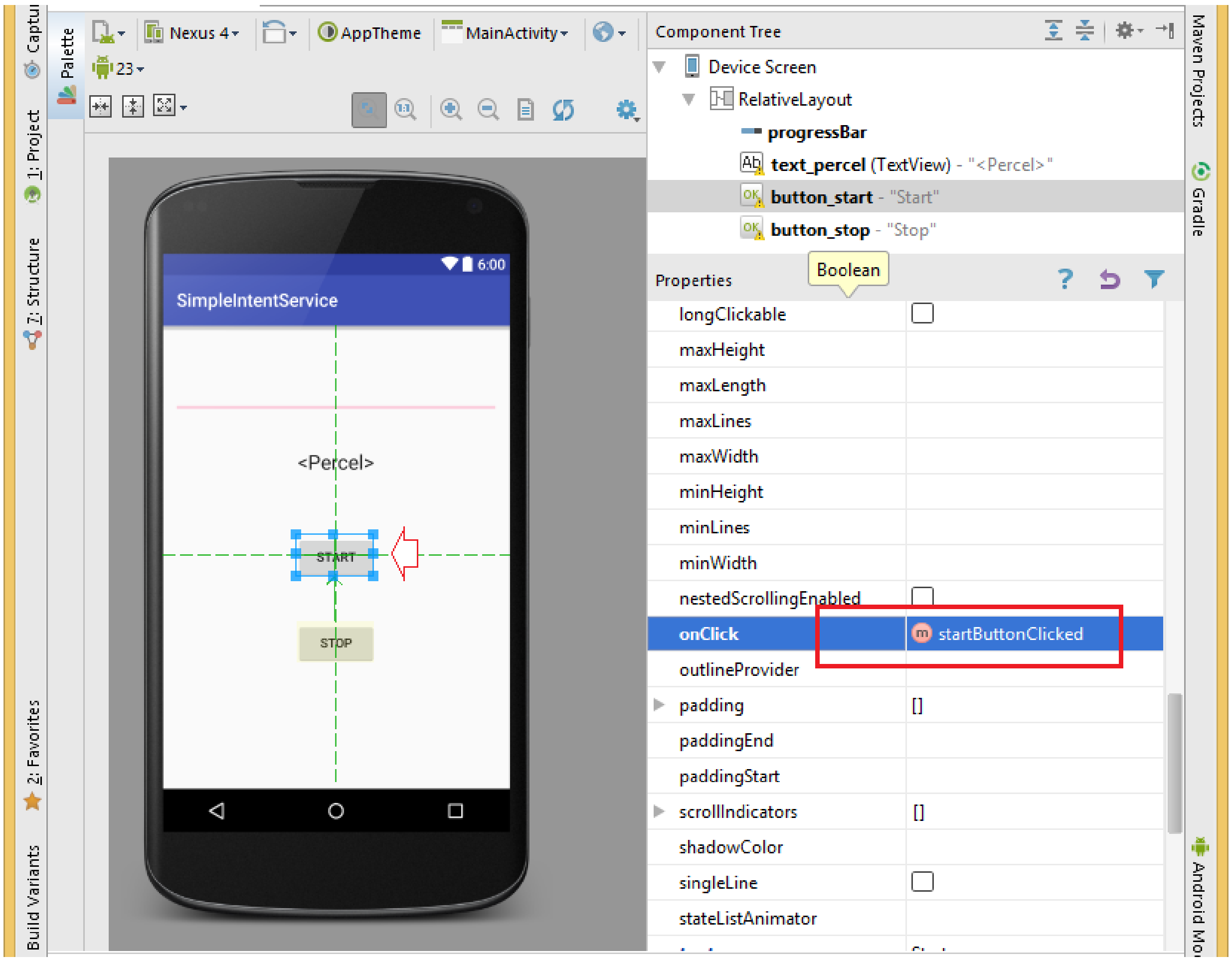
Component Tree

- Device Screen
  - RelativeLayout
    - **progressBar**
    - **text_percel** (TextView) - "<Percel>"
    - **button_start** - "Start"
    - **button_stop** - "Stop"

Properties

layout:width	fill_parent
layout:height	wrap_content
layout:margin	[?, ?, 66dp, ?, ?, ?, ?]
layout:alignEnd	
layout:alignParentEnd	☐
layout:alignParentStart	☐
layout:alignStart	
layout:toEndOf	
layout:toStartOf	
layout:alignComponent	[]
layout:alignParent	[top]
layout:centerInParent	horizontal
style	?android:attr/progressBarStyleH
accessibilityLiveRegion	
accessibilityTraversalAfter	
accessibilityTraversalBefore	
alpha	

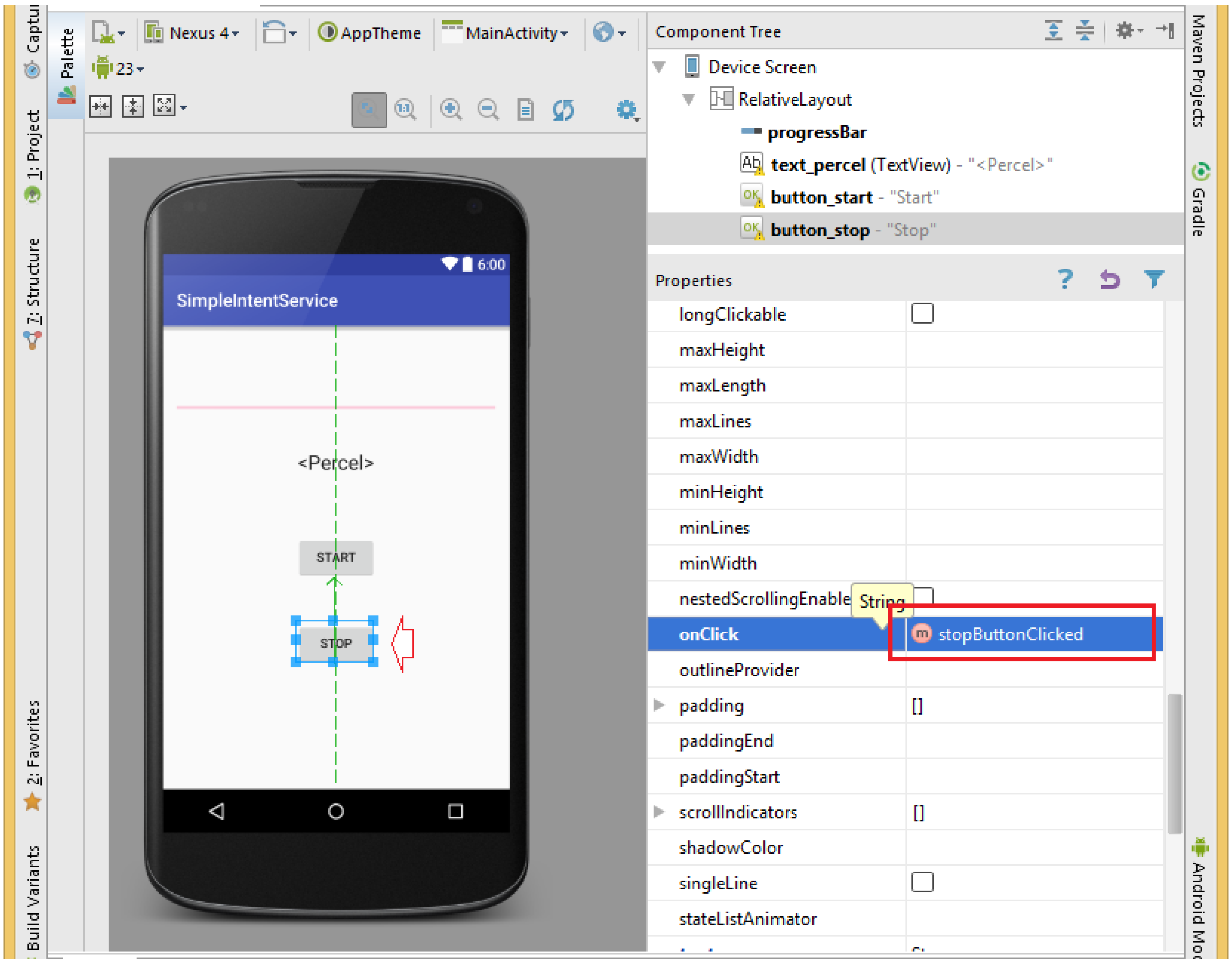
Set method that will be called when user clicks to **Start** button



Component Tree
- Device Screen
  - RelativeLayout
    - progressBar
    - text_percel (TextView) - "<Percel>"
    - button_start - "Start"
    - button_stop - "Stop"

Properties

longClickable	☐
maxHeight	
maxLength	
maxLines	
maxWidth	
minHeight	
minLines	
minWidth	
nestedScrollingEnabled	☐
onClick	startButtonClicked
outlineProvider	
padding	[]
paddingEnd	
paddingStart	
scrollIndicators	[]
shadowColor	
singleLine	☐
stateListAnimator	



Component Tree
- Device Screen
  - RelativeLayout
    - progressBar
    - text_percel (TextView) - "<Percel>"
    - button_start - "Start"
    - button_stop - "Stop"

Properties

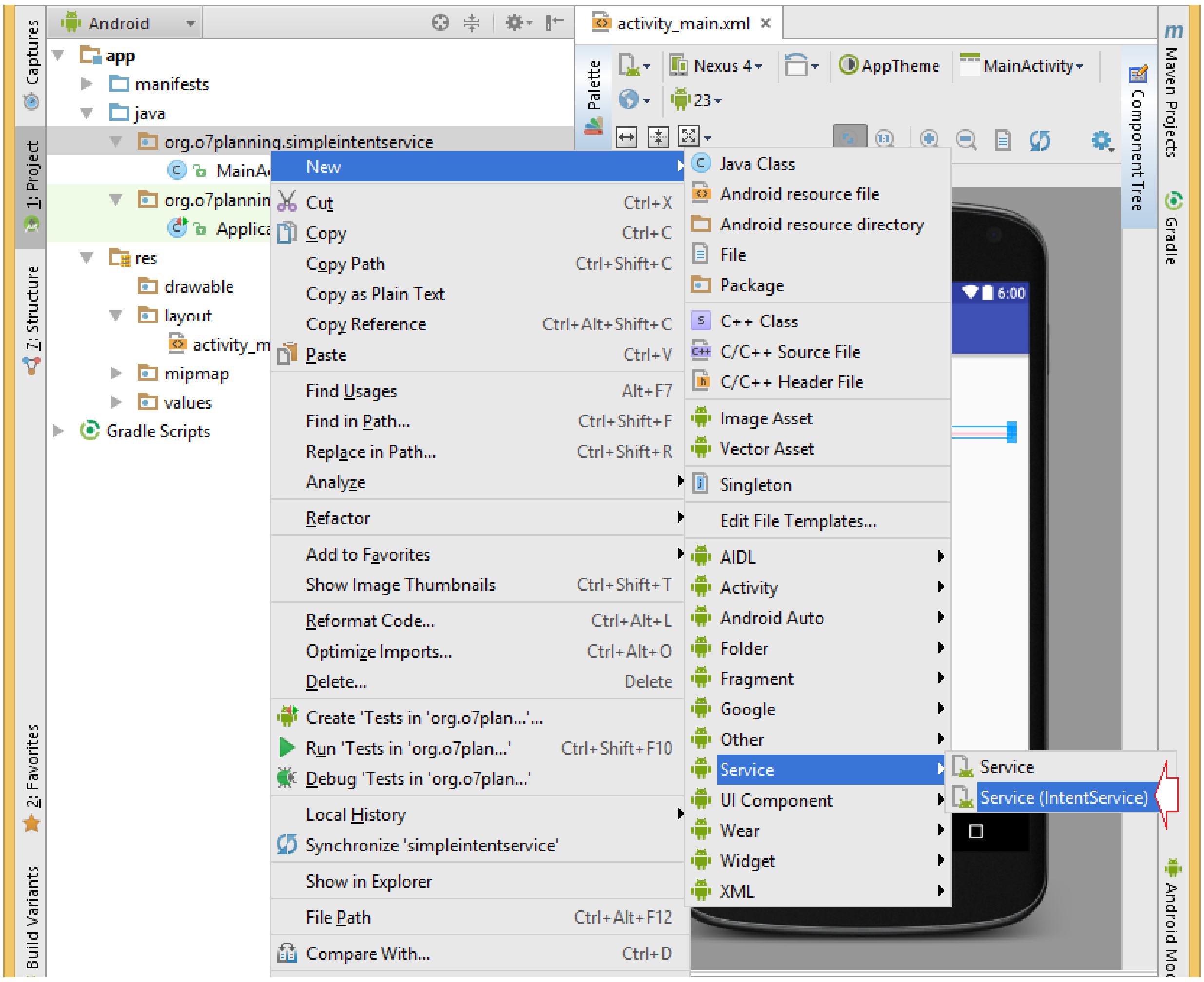
longClickable	☐
maxHeight	
maxLength	
maxLines	
maxWidth	
minHeight	
minLines	
minWidth	
nestedScrollingEnable	☐
onClick	stopButtonClicked
outlineProvider	
padding	[]
paddingEnd	
paddingStart	
scrollIndicators	[]
shadowColor	
singleLine	☐
stateListAnimator	

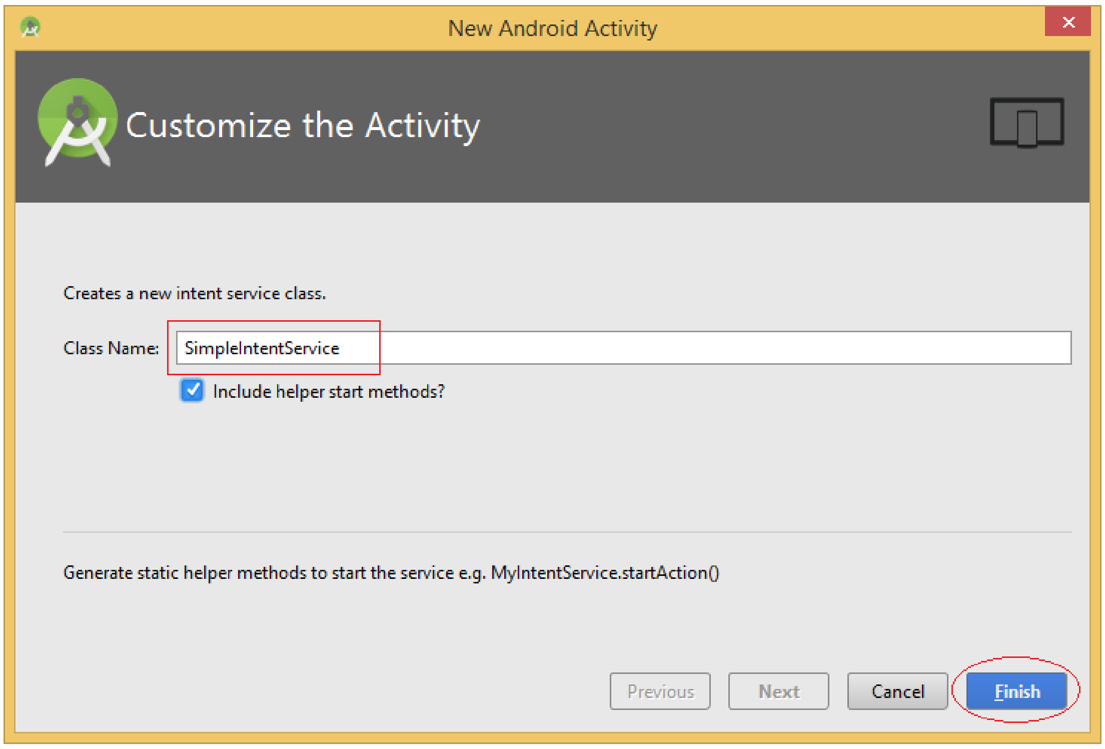
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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=".MainActivity">

    <ProgressBar
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/progressBar"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="66dp"
        android:indeterminate="false"
        android:max="100"
        android:progress="0" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="&lt;Percel>"
        android:id="@+id/text_percel"
        android:layout_below="@+id/progressBar"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="40dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start"
        android:id="@+id/button_start"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:onClick="startButtonClicked" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Stop"
        android:id="@+id/button_stop"
        android:layout_below="@+id/button_start"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="47dp"
        android:onClick="stopButtonClicked" />

</RelativeLayout>
```

Create **IntentService** by right-clicking to a package, and select:

- **New/Service/IntentService**





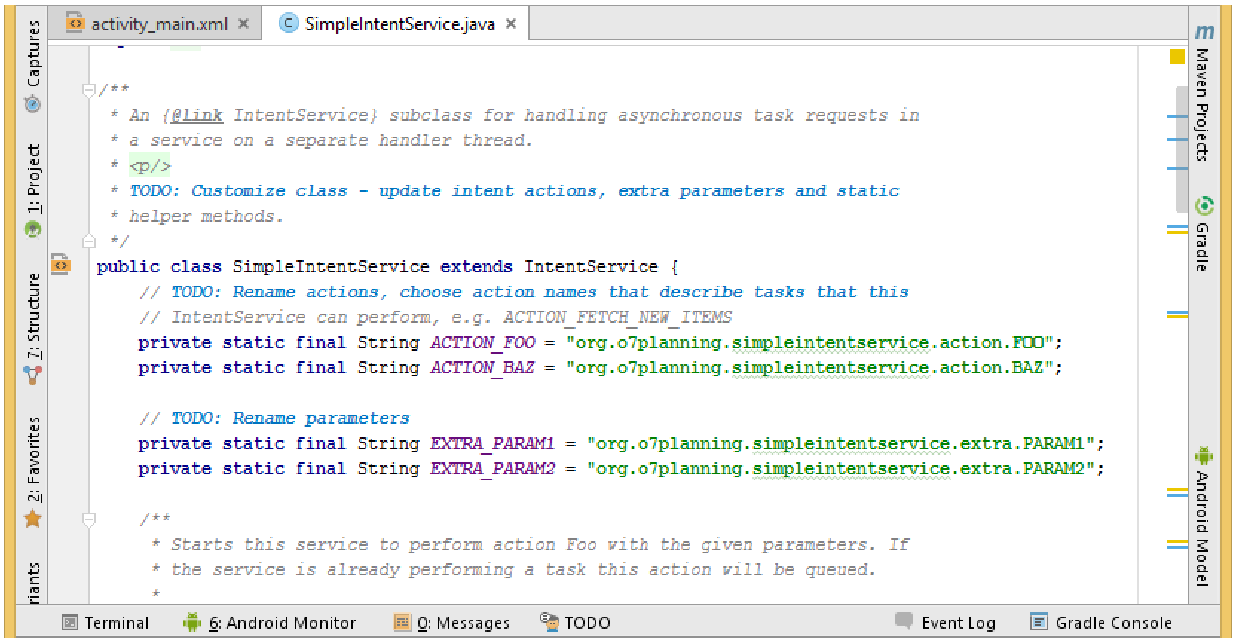
Creates a new intent service class.

Class Name: SimpleIntentService

☑ Include helper start methods?

Generate static helper methods to start the service e.g. MyIntentService.startAction()

SimpleIntentService has been created, it is also registered with **AndroidManifest.xml**, code generated is a suggestion for you to write a **IntentService**, you can erase the code generated.



```
/**
 * An {@link IntentService} subclass for handling asynchronous task requests in
 * a service on a separate handler thread.
 * <p/>
 * TODO: Customize class - update intent actions, extra parameters and static
 * helper methods.
 */
public class SimpleIntentService extends IntentService {
    // TODO: Rename actions, choose action names that describe tasks that this
    // IntentService can perform, e.g. ACTION_FETCH_NEW_ITEMS
    private static final String ACTION_FOO = "org.o7planning.simpleintentservice.action.FOO";
    private static final String ACTION_BAZ = "org.o7planning.simpleintentservice.action.BAZ";

    // TODO: Rename parameters
    private static final String EXTRA_PARAM1 = "org.o7planning.simpleintentservice.extra.PARAM1";
    private static final String EXTRA_PARAM2 = "org.o7planning.simpleintentservice.extra.PARAM2";

    /**
     * Starts this service to perform action Foo with the given parameters. If
     * the service is already performing a task this action will be queued.
     *
```

SimpleIntentService.java

```
package org.o7planning.simpleintentservice;

import android.app.IntentService;
import android.content.Intent;
import android.os.SystemClock;


public class SimpleIntentService extends IntentService {

   public static final String ACTION_1 ="MY_ACTION_1";

   public SimpleIntentService() {
      super("SimpleIntentService");
   }

   @Override
   protected void onHandleIntent(Intent intent) {

      // Create Intent object (to broadcast).
      Intent broadcastIntent = new Intent();

      // Set Action name for this Intent.
      // A Intent can perform many different actions.
      broadcastIntent.setAction(SimpleIntentService.ACTION_1);

      // Loop 100 times broadcast of Intent.
      for (int i = 0; i <= 100; i++) {

         // Set data
         // (Percent of work)
         broadcastIntent.putExtra("percel", i);
```

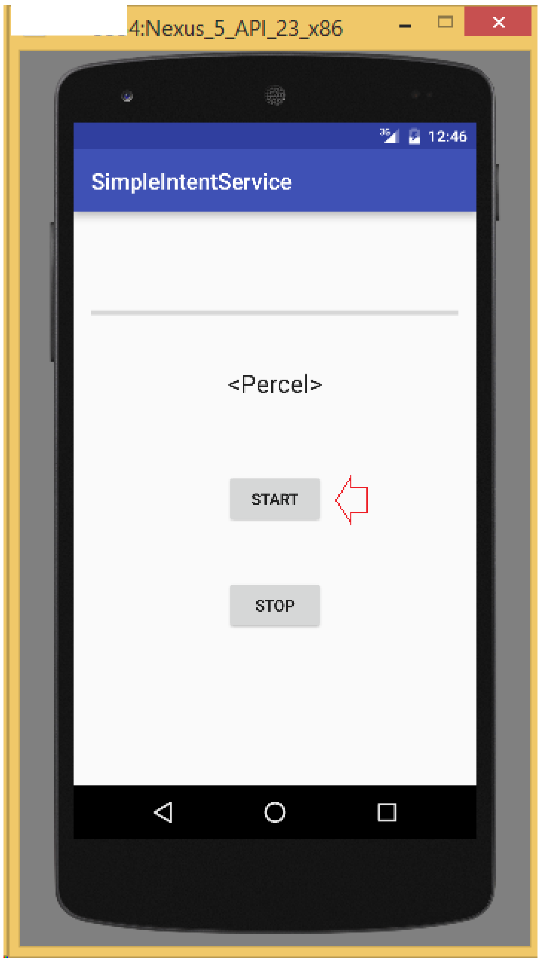
```
            // Send broadcast
            sendBroadcast(broadcastIntent);

            // Sleep 100 Milliseconds.
            SystemClock.sleep(100);
        }

    }
}
```

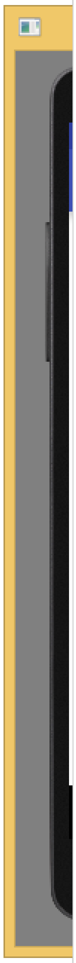
MainActivity.java

```
package org.o7planning.simpleintentservice;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    private Button startButton;
    private Button stopButton;
    private TextView percelText;

    private ProgressBar progressBar;

    private Intent serviceIntent;

    private ResponseReceiver receiver = new ResponseReceiver();


    // Broadcast component
    public class ResponseReceiver extends BroadcastReceiver {

        // on broadcast received
        @Override
        public void onReceive(Context context, Intent intent) {

            // Check action name.
            if(intent.getAction().equals(SimpleIntentService.ACTION_1)) {
                int value = intent.getIntExtra("percel", -1);

                new ShowProgressBarTask().execute(value);
            }
        }
    }

    // Display value for the ProgressBar.
    class ShowProgressBarTask extends AsyncTask<Integer, Integer, Integer> {

        @Override
        protected Integer doInBackground(Integer... args) {

            return args[0];
        }

        @Override
        protected void onPostExecute(Integer result) {
            super.onPostExecute(result);

            progressBar.setProgress(result);

            percelText.setText(result + " % Loaded");

            if (result == 100) {
                percelText.setText("Completed");
                startButton.setEnabled(true);
            }

        }
    }

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

        this.startButton= (Button) this.findViewById(R.id.button_start);
        this.stopButton = (Button)this.findViewById(R.id.button_stop);
        this.percelText = (TextView) this.findViewById(R.id.text_percel);
        this.progressBar = (ProgressBar) this.findViewById(R.id.progressBar);
    }


    @Override
    protected void onResume() {
        super.onResume();

        // Register receiver with Activity.
        registerReceiver(receiver, new IntentFilter(
                SimpleIntentService.ACTION_1));
    }

    @Override
    protected void onStop() {
        super.onStop();

        // Unregister receiver with Activity.
        unregisterReceiver(receiver);
    }

    // Method is called when the user clicks on the Start button.
    public void startButtonClicked(View view)  {
        startButton.setEnabled(false);

```

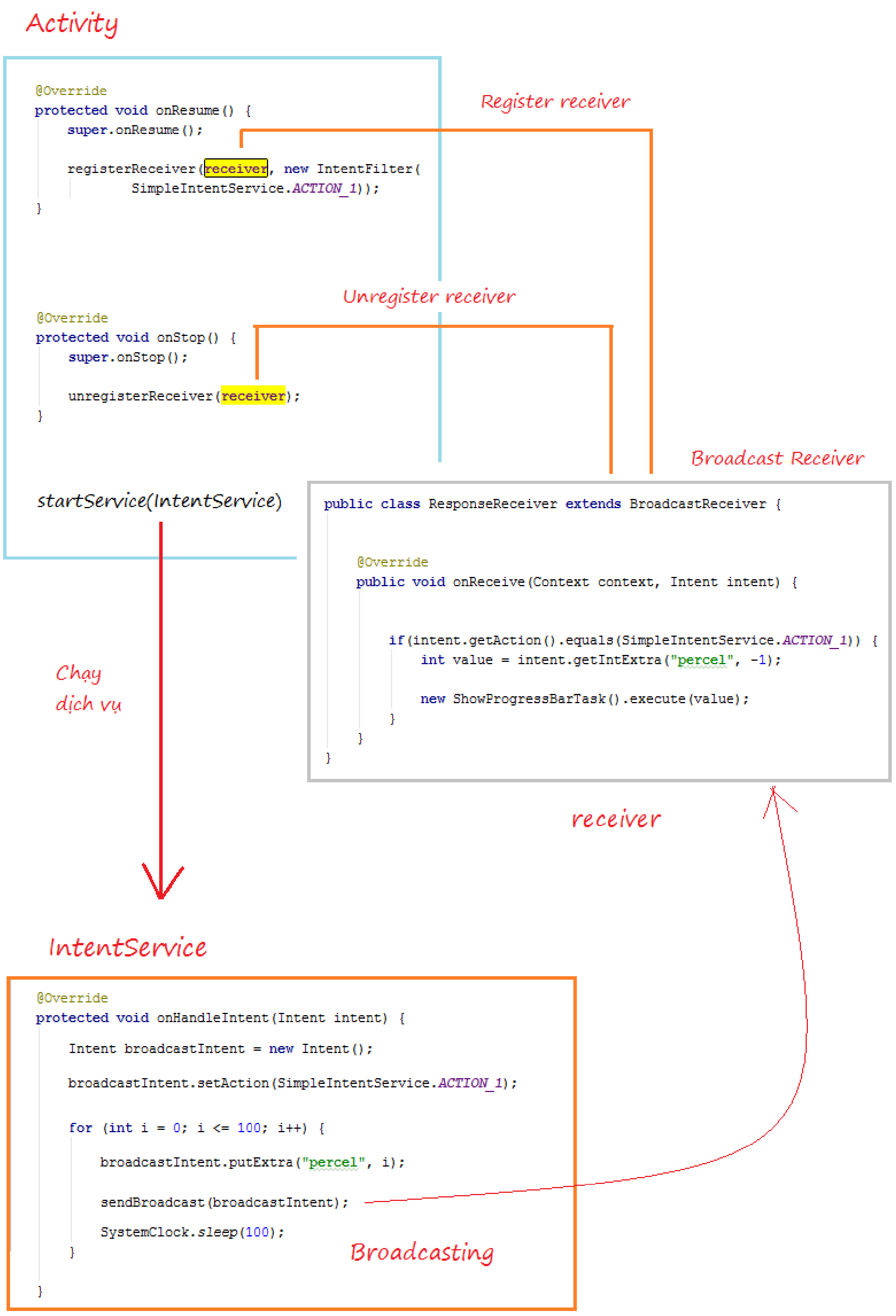
```
            serviceIntent = new Intent(this, SimpleIntentService.class);

            startService(serviceIntent);
        }


        public void stopButtonClicked(View view)  {
            if(serviceIntent!= null)  {
                // serviceIntent.get
            }
        }

    }
```

- Running the app (View slider):





And you can see the working principle of this example according to the illustration below:



```
@Override
protected void onResume() {
    super.onResume();

    registerReceiver(receiver, new IntentFilter(
            SimpleIntentService.ACTION_1));
}
```

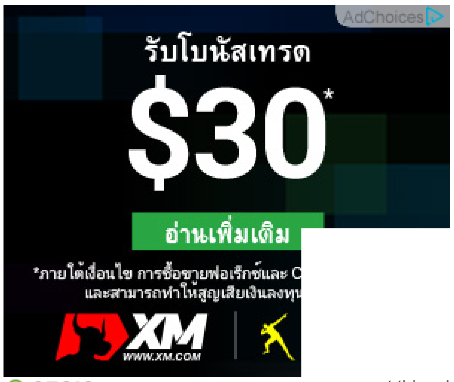
```
@Override
protected void onStop() {
    super.onStop();

    unregisterReceiver(receiver);
}
```

```
public class ResponseReceiver extends BroadcastReceiver {


    @Override
    public void onReceive(Context context, Intent intent) {


        if(intent.getAction().equals(SimpleIntentService.ACTION_1)) {
            int value = intent.getIntExtra("percel", -1);

            new ShowProgressBarTask().execute(value);
        }
    }
}
```

```
@Override
protected void onHandleIntent(Intent intent) {

    Intent broadcastIntent = new Intent();

    broadcastIntent.setAction(SimpleIntentService.ACTION_1);

    for (int i = 0; i <= 100; i++) {

        broadcastIntent.putExtra("percel", i);

        sendBroadcast(broadcastIntent);

        SystemClock.sleep(100);
    }

}
```

View more categories:

Android Programming Tutorials


AdChoices
รับโบนัสเทรด
$30*
อ่านเพิ่มเติม
*ภายใต้เงื่อนไข การซื้อขายฟอเร็กซ์และ C
และสามารถทำให้สูญเสียเงินลงทุน
XM
WWW.XM.COM

ezoic report this ad

Android Programming Tutorials

- What is needed to get started with Android?
- Installing and Configuring Android Studio
- Installing Intel Hardware Accelerated Execution Manager (Intel® HAXM)
- Configuring Android Emulator on Android Studio
- Android Tutorial for Beginners - Hello Android
- Android Tutorial for Beginners - Basic examples
- Using the Android Device Monitor
- How to add external libraries to Android Project in Android Studio?
- How to disable the permissions already granted to the Android application?
- Android UI Layouts Tutorial
- Android RadioGroup & RadioButton Tutorial
- Android AutoCompleteTextView & MultiAutoCompleteTextView Tutorial
- Android ImageView Tutorial
- Android ImageSwitcher Tutorial
- Android WebView Tutorial
- Android SeekBar Tutorial
- Android Fragments Tutorial
- Android ListView Tutorial
- Android GridView Tutorial
- Android StackView Tutorial
- Android Camera Tutorial
- Android MediaPlayer and VideoView Tutorial
- Playing Sound effects in Android with SoundPool
- Android Networking Tutorial
- Android JSON Parser Tutorial
- Android SharedPreferences Tutorial
- Android Internal Storage Tutorial
- Android External Storage Tutorial
- Android Intents Tutorial
- Android Notifications Tutorial
- Android SQLite Database Tutorial
- Google Maps Android API Tutorial
- Android Text to Speech Tutorial
- Android 2D Game Tutorial for Beginners

AdChoices
รับโบนัสเทรด
$30*
อ่านเพิ่มเติม
*ภายใต้เงื่อนไข การซื้อขายฟอเร็กซ์และ C
และสามารถทำให้สูญเสียเงินลงทุน
XM WWW.XM.COM

ezoic report this ad

Newest Documents

- Create a simple Chat application with Spring Boot and Websocket
- Spring Email Tutorial
- Bootstrap List Groups Tutorial
- Bootstrap Alerts Tutorial
- Bootstrap Models Tutorial
- Introducing Bootstrap
- Bootstrap 4 Grid System Tutorial
- Connecting to MySQL Database using NodeJS
- NodeJS EventEmitter Tutorial
- Understanding Event Loop in NodeJS

AdChoices
รับโบนัสเทรด
$30*
อ่านเพิ่มเติม
*ภายใต้เงื่อนไข การซื้อขายฟอเร็กซ์และ C
และสามารถทำให้สูญเสียเงินลงทุน
XM WWW.XM.COM

ezoic report this ad

- **Coupons On Auto Services**
- **Harley Davidson Service Manuals**
- **Repair Service Manual**

เทรดกับโบรกเกอร์แบบ
True ECN
ที่ใหญ่ที่สุด ในออสเตรเลียได้แล้ววันนี้
0.0 PIPS
เปิดบัญชี
IC Markets
การซื้อขายฟอเร็กซ์ และตราสารอนุพันธ์มีความเสี่ยงสูง
กรุณาอ่านเอกสารเปิดเผยข้อมูลผลิตภัณฑ์ บนเว็บไซต์ของเรา
www.icmarkets.com AFSL 335692

ezoic report this ad

o7planning.org