CODE > ANDROID SDK

Create a Music Player on Android: Song Playback

by Sue Smith 24 Mar 2014

Difficulty: Beginner Length: Medium Languages: English

Android SDK | Mobile Development

In this series, we are creating a music player on Android using the `MediaPlayer` and `MediaController` classes. In the first part, we created the app and prepared the user interface for playback. We presented the list of songs on the user device and specified a method to execute when the user makes a selection. In this part of the series, we will implement a `Service` class to execute music playback continuously, even when the user is not directly interacting with the application.

Looking for a Quick Solution?

This series takes you through the full process of creating an Android music player from scratch, but another option is to use one of the music player app templates on Envato Market, such as Android Music Player, which lets users browse and play music by albums, artists, songs, playlists, folders, and album artists.

Android Music Player

Or, for more personalised help, you could hire a mobile app developer on Envato Studio. That way, you can get reliable, affordable help on any aspects of the development work you're not comfortable with, and focus your energy and time on areas in which you're stronger.

Introduction

We will need the app to bind to the music playing `Service` in order to interact with playback, so you will learn some of the basic aspects of the `Service` life cycle in this tutorial if you haven't explored them before. Here is a preview of the final result that we're working towards:

In the final installment of this series, we'll add user control over the playback and we'll also make sure the app will continue to function in various application states. Later, we will follow the series up with additional enhancements that you may wish to add, such as audio focus control, video and streaming media playback, and alternate ways of presenting the media files.

1. Create a Service

Step 1

Add a new class to your app, naming it **MusicService** or another name of your choice. Make sure it matches the name you listed in the Manifest. When creating the class in Eclipse, choose `android.app.Service` as its superclass. Eclipse should enter an outline:

```
public class MusicService extends Service {

  @Override
  public IBinder onBind(Intent arg0) {
    // TODO Auto-generated method stub
    return null;
  }
```

```
8
9  }
```

Extend the opening line of the class declaration to implement some interfaces we will use for music playback:

```
1  public class MusicService extends Service implements
2  MediaPlayer.OnPreparedListener, MediaPlayer.OnErrorListener,
3  MediaPlayer.OnCompletionListener {
```

Eclipse will display an error over the class name. Hover over the error and choose **Add unimplemented methods**. We will add code to the methods in a few moments. The interfaces we are implementing will aid the process of interacting with the `MediaPlayer` class.

Your class will also need the following additional imports:

```
1  import java.util.ArrayList;
2  import android.content.ContentUris;
3  import android.media.AudioManager;
4  import android.media.MediaPlayer;
5  import android.net.Uri;
6  import android.os.Binder;
7  import android.os.PowerManager;
8  import android.util.Log;
```

Step 2

Add the following instance variables to the new `Service` class:

```
1  //media player
2  private MediaPlayer player;
3  //song list
4  private ArrayList<Song> songs;
5  //current position
6  private int songPosn;
```

We will pass the list of songs into the `Service` class, playing from it using the `MediaPlayer` class and keeping track of the position of the current song using the `songPosn` instance variable. Now, implement the `onCreate` method for the `Service`:

```
1  public void onCreate(){
2    //create the service
3  }
```

Inside `onCreate`, call the superclass method, instantiating the position and `MediaPlayer` instance variables:

```
1  //create the service
2  super.onCreate();
3  //initialize position
4  songPosn=0;
5  //create player
6  player = new MediaPlayer();
```

Next, let's add a method to initialize the `MediaPlayer` class, after the `onCreate` method:

```
1  public void initMusicPlayer(){
```

```
  //set player properties
}
```

Inside this method, we configure the music player by setting some of its properties as shown below:

```
player.setWakeMode(getApplicationContext(),
  PowerManager.PARTIAL_WAKE_LOCK);
player.setAudioStreamType(AudioManager.STREAM_MUSIC);
```

The wake lock will let playback continue when the device becomes idle and we set the stream type to music. Set the class as listener for (1) when the `MediaPlayer` instance is prepared, (2) when a song has completed playback, and when (3) an error is thrown:

```
player.setOnPreparedListener(this);
player.setOnCompletionListener(this);
player.setOnErrorListener(this);
```

Notice that these correspond to the interfaces we implemented. We will be adding code to the `onPrepared`, `onCompletion`, and `onError` methods to respond to these events. Back in `onCreate`, invoke `initMusicPlayer`:

```
initMusicPlayer();
```

Step 3

It's time to add a method to the `Service` class to pass the list of songs from the `Activity`:

```
public void setList(ArrayList<Song> theSongs){
  songs=theSongs;
}
```

We will call this method later in the tutorial. This will form part of the interaction between the `Activity` and `Service` classes, for which we also need a `Binder` instance. Add the following snippet to the `Service` class after the `setList` method:

```
public class MusicBinder extends Binder {
  MusicService getService() {
    return MusicService.this;
  }
}
```

We will also access this from the `Activity` class.

2. Start the Service

Step 1

Back in your app's main `Activity` class, you will need to add the following additional imports:

```
import android.os.IBinder;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
```

```
import android.content.ServiceConnection;
import android.view.MenuItem;
import android.view.View;
```

And you'll also need to declare three new instance variables:

```
private MusicService musicSrv;
private Intent playIntent;
private boolean musicBound=false;
```

We are going to play the music in the `Service` class, but control it from the `Activity` class, where the application's user interface operates. To accomplish this, we will have to bind to the `Service` class. The above instance variables represent the `Service` class and `Intent`, as well as a flag to keep track of whether the `Activity` class is bound to the `Service` class or not. Add the following to your `Activity` class, after the `onCreate` method:

```
//connect to the service
private ServiceConnection musicConnection = new ServiceConnection(){

  @Override
  public void onServiceConnected(ComponentName name, IBinder service) {
    MusicBinder binder = (MusicBinder)service;
    //get service
    musicSrv = binder.getService();
    //pass list
    musicSrv.setList(songList);
    musicBound = true;
  }

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

The callback methods will inform the class when the `Activity` instance has successfully connected to the `Service` instance. When that happens, we get a reference to the `Service` instance so that the `Activity` can interact with it. It starts by calling the method to pass the song list. We set the boolean flag to keep track of the binding status. You will need to import the `Binder` class we added to the `Service` class at the top of your `Activity` class:

```
import com.example.musicplayer.MusicService.MusicBinder;
```

Don't forget to alter the package and class names to suit your own if necessary.

Step 2

We will want to start the `Service` instance when the `Activity` instance starts, so override the `onStart` method:

```
@Override
protected void onStart() {
  super.onStart();
  if(playIntent==null){
    playIntent = new Intent(this, MusicService.class);
    bindService(playIntent, musicConnection, Context.BIND_AUTO_CREATE);
    startService(playIntent);
  }
}
```

When the `Activity` instance starts, we create the `Intent` object if it doesn't exist yet, bind to it, and start it. Alter the code if you chose a different name for the `Service` class. Notice that we use the connection object we created so that when the connection to the bound `Service` instance is made, we pass the song list. We will also be able to interact with the `Service` instance in order to control playback later.

Return to your `Service` class to complete this binding process. Add an instance variable representing the inner `Binder` class we added:

```
private final IBinder musicBind = new MusicBinder();
```

Now amend the `onBind` method to return this object:

```
@Override
public IBinder onBind(Intent intent) {
  return musicBind;
}
```

Add the `onUnbind` method to release resources when the `Service` instance is unbound:

```
@Override
public boolean onUnbind(Intent intent){
  player.stop();
  player.release();
  return false;
}
```

This will execute when the user exits the app, at which point we will stop the service.

3. Begin Playback

Step 1

Let's now set the app up to play a track. In your `Service` class, add the following method:

```
public void playSong(){
  //play a song
}
```

Inside the method, start by resetting the `MediaPlayer` since we will also use this code when the user is playing subsequent songs:

```
player.reset();
```

Next, get the song from the list, extract the ID for it using its `Song` object, and model this as a URI:

```
//get song
Song playSong = songs.get(songPosn);
//get id
long currSong = playSong.getID();
//set uri
Uri trackUri = ContentUris.withAppendedId(
```

```
    android.provider.MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
    currSong);
```

We can now try setting this URI as the data source for the `MediaPlayer` instance, but an exception may be thrown if an error pops up so we use a `try/catch` block:

```
try{
  player.setDataSource(getApplicationContext(), trackUri);
}
catch(Exception e){
  Log.e("MUSIC SERVICE", "Error setting data source", e);
}
```

After the `catch` block, complete the `playSong` method by calling the asynchronous method of the `MediaPlayer` to prepare it:

```
player.prepareAsync();
```

Step 2

When the `MediaPlayer` is prepared, the `onPrepared` method will be executed. Eclipse should have inserted it in your `Service` class. Inside this method, start the playback:

```
@Override
public void onPrepared(MediaPlayer mp) {
  //start playback
  mp.start();
}
```

In order for the user to select songs, we also need a method in the `Service` class to set the current song. Add it now:

```
public void setSong(int songIndex){
  songPosn=songIndex;
}
```

We will call this when the user picks a song from the list.

Step 3

Remember that we added an `onClick` attribute to the layout for each item in the song list. Add that method to the main `Activity` class:

```
public void songPicked(View view){
  musicSrv.setSong(Integer.parseInt(view.getTag().toString()));
  musicSrv.playSong();
}
```

We set the song position as the tag for each item in the list view when we defined the `Adapter` class. We retrieve it here and pass it to the `Service` instance before calling the method to start the playback.

Before you run your app, implement the **end** button we added to the main menu. In your main `Activity` class, add the method to respond to menu item selection:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  //menu item selected
}
```

Inside the method, add a switch statement for the actions:

```
switch (item.getItemId()) {
case R.id.action_shuffle:
  //shuffle
  break;
case R.id.action_end:
  stopService(playIntent);
  musicSrv=null;
  System.exit(0);
  break;
}
return super.onOptionsItemSelected(item);
```

We will implement the shuffle function in the next tutorial. For the **end** button, we stop the `Service` instance and exit the app. Pressing the back button will not exit the app, since we will assume that the user wants playback to continue unless they select the **end** button. Use the same process if the the app is destroyed, overriding the activity's `onDestroy` method:

```
@Override
protected void onDestroy() {
  stopService(playIntent);
  musicSrv=null;
  super.onDestroy();
}
```

When you run the app at this point, you will be able to play songs by selecting them from the list and you can also exit the app using the **end** button.

Conclusion

We have now implemented basic playback of music tracks selected from the user's list of music files. In the final part of this series, we will add a media controller through which the user will be able to control playback. We will add a notification to let the user return to the app after navigating away from it and we will carry out some housekeeping to make the app cope with a variety of user actions.

Sue Smith

Technical writer (and sometimes developer) based in Glasgow, UK. Having worked with the Mozilla Foundation and various online publications, I enjoy helping people to learn web and software development topics, regardless of their existing skill level. Particular areas of interest include education technology and open source projects.

BrainDeadAir

Art82 • 3 years ago

I can't get this to work - clicking a song makes it to crash. I think musicSrv is still null, so a problem with the serviceConnection?. Do you know why?

8 • Reply • Share ›

Jer75 ➚ Art82 • 3 years ago

Yes, I'm getting the same problem, no errors in the code but followed the instructions exactly and when I click to play a song it crashes. anybody know what can fix it?

14 • Reply • Share ›

Lawrence Pickford ➚ Jer75 • 3 years ago

Make sure you renamed the line:

<service android:name="com.example.YOURAPPNAMEHERE.MusicService"/>

Doesn't flag up as an error but causes a hang.

13 • Reply • Share ›

Omar Devila ➚ Lawrence Pickford • 3 years ago

This was my problem. Problem solved

2 • Reply • Share ›

Inzi Nasir ➚ Jer75 • a year ago

add this line of code in your manifest according to your package name <service android:name="com.example.arslan.musicplayer.MusicService"/>

• Reply • Share ›

lokesh kumar ➚ Jer75 • 2 years ago

the Issue is that the method setList() is not called in the mainActivity. So the error of nullpointer exception is there because the songs array is null.

• Reply • Share ›

gamekathu ➚ Art82 • 3 years ago

Yes same problem, and I think the service is not being started as musicSrv is null... am I missing something?

2 • Reply • Share ›

haash ➚ gamekathu • 2 years ago

same problem....any one can help me... the code inside manifest are correct...but musicSrv is null

• Reply • Share ›

lokesh kumar ➚ Art82 • 2 years ago

Add this code -

musicSrv.setList(songList);

in the onServiceConnected() method in MainActivity.java.

Reply • Share ›

Joanna ➚ lokesh kumar • 2 months ago

I have the same problem, but I have already corrected the manifest file and I do have musicSrv.setList(songList) line. When I debug the application, it doesn't stop in onServiceConnected, so the musicSrv isn't initialized. Anyone has any idea how to fix it?

Reply • Share ›

anon ➚ Art82 • 3 years ago

You probably commented out, rather than implementing
public void onCompletion(MediaPlayer mp) {
}
public boolean onError(MediaPlayer mp, int a, int b) {
return false;
}

Reply • Share ›

Vyshnav Ramesh ➚ anon • 7 months ago

tnx, this helped

Reply • Share ›

lasya ➚ Vyshnav Ramesh • 7 months ago

I am getting the list of songs displayed when i open my application but when i click on the song it is not playing stating an error "Should have subtitle controller already set" ..... can u please solve it

Reply • Share ›

Vyshnav Ramesh ➚ lasya • 6 months ago

"Should have subtitle controller already set" appears since it is not set like Title of song in the controller class. It has nothing to do with not playing the song. Even I do get the same error. Your issue may be either you haven't set the Service class properly or any other missed any code. Kindly attach a Stackoverflow link to your question here. Don't worry, I'll help

Reply • Share ›

Vyshnav Ramesh ➚ lasya • 7 months ago

I too have got this: "Should have subtitle controller already set". But the player works fine, which means you may have code wrong elsewhere On clicking each song this appears to me.
Post a StackOverflow question and share me the link. Let me check

Reply • Share ›

bar ➚ Vyshnav Ramesh • 4 months ago

can you help me too?

Reply • Share ›

adi ➚ Vyshnav Ramesh • 5 months ago

On clicking the notification icon following error occors :

android.view.WindowLeaked: Activity com.example.music1.MainActivity has leaked window com.android.internal.policy.impl.PhoneWindow$DecorView{1161076b V.ED.... R.....ID 0,0-1080,264} that was originally added here
at android.view.ViewRootImpl.<init>(ViewRootImpl.java:363)
at android.view.WindowManagerGlobal.addView(WindowManagerGlobal.java:271)
at android.view.WindowManagerImpl.addView(WindowManagerImpl.java:85)
at android.widget.MediaControl...(MediaController.java:354)
at android.widget.MediaControl...(MediaController.java:314)
at com.example.music1.MainActivity.setController(MainActivity.java:87)
at com.example.music1.MainActivity.access$600(MainActivity.java:26)
at com.example.music1.MainActivity$6.onGlobalLayout(MainActivity.java:173)

at com.example.musicplayer.MainActivity$6.onGlobalLayout(MainActivity.java:173)
at android.view.ViewTreeObserver.dispatchOnGlobalLayout(ViewTreeObserver.java:912)

facing similiar error?

Reply • Share ›

Rakshith • 3 years ago

i get the following error:

9-08 17:52:12.289: E/AndroidRuntime(26927): FATAL EXCEPTION: main

09-08 17:52:12.289: E/AndroidRuntime(26927): java.lang.IllegalStateException: Could not execute method of the activity

09-08 17:52:12.289: E/AndroidRuntime(26927): at android.view.View$1.onClick(View.java:2165)

09-08 17:52:12.289: E/AndroidRuntime(26927): at android.view.View.performClick(View.java:2506)

09-08 17:52:12.289: E/AndroidRuntime(26927): at android.view.View$PerformClick.run(View.java:9112)

09-08 17:52:12.289: E/AndroidRuntime(26927): at android.os.Handler.handleCallback(Handler.java:587)

09-08 17:52:12.289: E/AndroidRuntime(26927): at android.os.Handler.dispatchMessage(Handler.java:92)

09-08 17:52:12.289: E/AndroidRuntime(26927): at android.os.Looper.loop(Looper.java:130)

09-08 17:52:12.289: E/AndroidRuntime(26927): at android.app.ActivityThread.main(ActivityThread.java:3835)

09-08 17:52:12.289: E/AndroidRuntime(26927): at java.lang.reflect.Method.invokeNative(Native Method)

see more

5 • Reply • Share ›

Beginner ➚ Rakshith • 3 years ago

In part 1 of this tutorial, we added
<service android:name="com.example.musicplayer.MusicService"/> in the manifest file which should have been
<service android:name="com.example.mediaplayer.MusicService"/> since we are using the name "mediaplayer.MusicService" in this tutorial for the service. Changing it made it work for me.

2 • Reply • Share ›

Payal ➚ Rakshith • 3 years ago

i am gettting same error ,how it get solved??

1 • Reply • Share ›

Meelan Lakhoo ➚ Rakshith • 2 years ago

Make sure that you enter android.app.Service as its superclass. when creating the class.

• Reply • Share ›

haash ➚ Rakshith • 2 years ago

i am getting following error :

02-22 20:30:00.925: E/AndroidRuntime(19394): FATAL EXCEPTION: main
02-22 20:30:00.925: E/AndroidRuntime(19394): java.lang.IllegalStateException: Could not execute method of the activity
02-22 20:30:00.925: E/AndroidRuntime(19394): at android.view.View$1.onClick(View.java:3804)
02-22 20:30:00.925: E/AndroidRuntime(19394): at android.view.View.performClick(View.java:4439)
02-22 20:30:00.925: E/AndroidRuntime(19394): at android.view.View$PerformClick.run(View.java:18395)
02-22 20:30:00.925: E/AndroidRuntime(19394): at android.os.Handler.handleCallback(Handler.java:725)
02-22 20:30:00.925: E/AndroidRuntime(19394): at android.os.Handler.dispatchMessage(Handler.java:92)
02-22 20:30:00.925: E/AndroidRuntime(19394): at android.os.Looper.loop(Looper.java:176)
02-22 20:30:00.925: E/AndroidRuntime(19394): at android.app.ActivityThread.main(ActivityThread.java:5317)
02-22 20:30:00.925: E/AndroidRuntime(19394): at java.lang.reflect.Method.invokeNative(Native Method)
02-22 20:30:00.925: E/AndroidRuntime(19394): at java.lang.reflect.Method.invoke(Method.java:511)
02-22 20:30:00.925: E/AndroidRuntime(19394): at
com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:1102)
02-22 20:30:00.925: E/AndroidRuntime(19394): at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:869)

02-22 20:30:00.925: E/AndroidRuntime(19394): at dalvik.system.NativeStart.main(Native Method)
02-22 20:30:00.925: E/AndroidRuntime(19394): Caused by: java.lang.reflect.InvocationTargetException

see more

Reply • Share ›

ritesh rana ➚ haash • a year ago
pls email me on ritesh.rana53@gmail.com
Reply • Share ›

ritesh rana ➚ haash • a year ago
hi friends i have the same problem how could i solve this pls help me
Reply • Share ›

Meelan Lakhoo ➚ haash • 2 years ago
Make sure that you enter android.app.Service as its superclass. when creating the class. this resolved my error
Reply • Share ›

Eric • 3 years ago
When my song is playing, or when the music player is open, I cant get the "Back" button to work so I can get back to my app main activity. Any help please
4 Reply • Share ›

Zaq Avila • 3 years ago
This is a great tutorial. Followed everything from top to bottom and got everything working perfect. I did get an error after onDestroy() is called - you didn't unBind service. Stopping the service is not enough.
Thank you so much, on to the next tutorial!
3 Reply • Share ›

Iasya ➚ Zaq Avila • 7 months ago
I am getting the list of songs displayed when i open my application but when i click on the song it is not playing stating an error "Should have subtitle controller already set" ..... can u please solve it
Reply • Share ›

xyz • 3 years ago
i want a back button
3 Reply • Share ›

Abidroid • 3 years ago
Where is the seekbar update?
2 Reply • Share ›

Leo • 3 years ago
why we can not using back key in here? Do any one know how to play music in background
2 Reply • Share ›

Vyshnav Ramesh ➚ Leo • 7 months ago
hi, have you fixed it?
Reply • Share ›

ritesh goel • a year ago
Hi, when i press the back button, music gets stopped. how to fix this?
1 Reply • Share ›

Vyshnav Ramesh ➚ ritesh goel • 7 months ago
yeah me too, have u fixed it?
Reply • Share ›

Anindya Dutta ➚ Vyshnav Ramesh • 6 months ago

Did this get fixed?

▲ ▼ • Reply • Share ›

Vyshnav Ramesh ➚ Anindya Dutta • 6 months ago

Yeah, I have provided its solution in the comments section on the 3rd part of this series. You can find by sorting the comments by Newest

▲ ▼ • Reply • Share ›

Vyshnav Ramesh ➚ Anindya Dutta • 6 months ago

Yeah, I have provided its solution in the comments section on the 3rd part of this series. You can find by sorting the comments by Newest

▲ ▼ • Reply • Share ›

Rajkumar • a year ago

part one- http://code.tutsplus.com/tu...
part two- http://code.tutsplus.com/tu...
part three- http://code.tutsplus.com/tu...

1 ▲ ▼ • Reply • Share ›

Abad Ullah • 2 years ago

Can i read from one specific folder rather than reading the all media files?

1 ▲ ▼ • Reply • Share ›

VIET NGUYEN HONG • 2 years ago

Dear Sue, it's really a good tutorial. I have a problem that the song do not play, nothing happen. In my debuging, it's called to playSong() method anh the trackUri already get infomation. the block code: player.setDataSource(getApplicationContext(), trackUri); is called but the song is not played. Can you help me???

1 ▲ ▼ • Reply • Share ›

matteo ferri ➚ VIET NGUYEN HONG • 2 years ago

Same problem, I have a breakpoint in onPrepared method, where is the mp.start(), but I never stop it.

▲ ▼ • Reply • Share ›

StPatrick • 3 years ago

I have my service set up correctly, STILL I get a NullPointerException on playSong. Everything is done EXACTLY as it's supposed to be. onServiceConnected never gets called at all. Flat out doesn't work. I've been wrestling with this for 4 hours now. Service is defined as the last element inside the application element:

<application android:allowbackup="true" android:icon="@drawable/ic_launcher_v2" android:label="@string/app_name" android:theme="@style/AppTheme">
<activity android:name=".ServerActivity" android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<service android:name="com.patrickchristensen.qup.services.MusicService"/>
</application>

I'm debugging on my Nexus 5 and 7 respectively running Android 5.0 and 5.0.1. No fix online has worked.

EDIT: I'm an idiot. My intent was going to MusicBinder.class instead of MusicService.class. 'Doh!

1 ▲ ▼ • Reply • Share ›

Gio Mateo • 3 years ago

Helpppp

1 • Reply • Share ›

Vyshnav Ramesh ➚ Gio Mateo • 7 months ago

change 'ampersand gt;' to >

• Reply • Share ›

maarten ➚ Gio Mateo • 3 years ago

">" means ">", it must have automaticly converted it when copying the text.

• Reply • Share ›

Aman Matharu • 3 years ago

public void songPicked(View view) {
musicSrv.setSong(Integer.parseInt(view.getTag().toString()));
musicSrv.playSong();
}

//i don't know why I have errors with setSong and playSong when both are already inside service class.
any help?

1 • Reply • Share ›

Đạt Heo ➚ Aman Matharu • 2 years ago

maybe you need declare MusicService in your manifest

• Reply • Share ›

Zac ➚ Aman Matharu • 3 years ago

Bump for this post I'm getting Null Point Exception in the songPicked method as well anyone have any ideas?

• Reply • Share ›

siddhu • 3 years ago

Great Thankyou

1 • Reply • Share ›

bhaskar • 15 days ago

guys i am working on fragments ...and there is an error the songpicked method is not calling from song.xml onclick .....help please

• Reply • Share ›

Load more comments