CODE > ANDROID SDK

Create a Music Player on Android: Project Setup

by Sue Smith 12 Mar 2014

Difficulty: Beginner Length: Long Languages: English

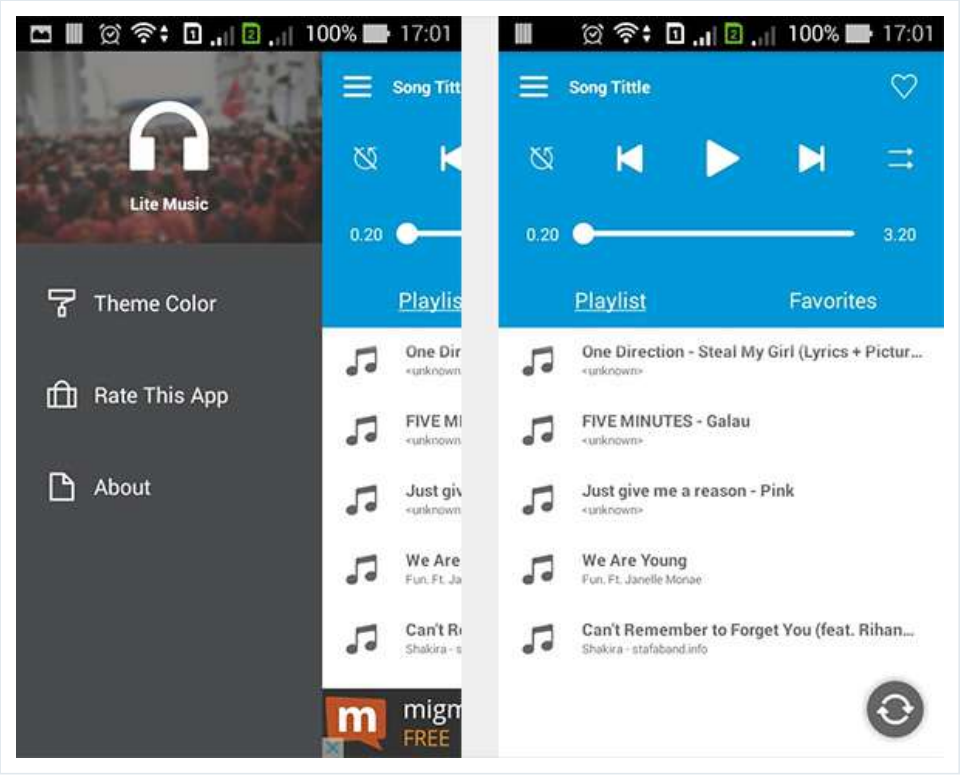
Android SDK Mobile Development

The Android platform provides resources for handling media playback, which your apps can use to create an interface between the user and their music files. In this tutorial series, we will create a basic music player application for Android. The app will present a list of songs on the user device, so that the user can select songs to play. The app will also present controls for interacting with playback and will continue playing when the user moves away from the app, with a notification displayed while playback elapses.

Looking for a Quick Solution?

If you're looking for a quick solution, there's a great collection of Android app templates over at Envato Market.

In particular, this Android Music Player app template is a great way to get started with building your own app. "Lite Music" is a premium player app template in Android, with a clean interface, that's simple and elegant to use.



Introduction

Building the music player will involve using the `ContentResolver` class to retrieve tracks on the device, the `MediaPlayer` class to play audio and the `MediaController` class to control playback. We will also use a `Service` instance to play audio when the user is not directly interacting with the app. You should be able to complete this series if you're an intermediate Android developer, so if you've already built a few apps, then this series shouldn't be a problem for you. Here is a preview of the final app:

In this tutorial, we will create the app and query the user device for audio files using the `ContentResolver` and `Cursor` classes. In the next part, we will use an `Adapter` instance to present the songs in a list view, starting playback when the user taps an item from the list. In the final installment of this series, we'll use the `MediaController` class to give the user control over playback, implement functions to skip forward and back, and include a shuffle function. After this series, we will explore other aspects of media playback that can enhance the app, such as handling audio focus, presenting media files in different ways, and playing streaming media.

1. Create and Configure a New Project

Step 1

Create a new Android project. If you are using Eclipse, then let the IDE (Integrated Development Environment) create a main `Activity` class and layout file for you. For some of the code we use in the series, you will need a minimum API level of 16, so you will need to take additional steps to support older versions. Once your project is created, open the project's Manifest file. Inside the `manifest` element, add the following permission:

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

We will use this permission to let music playback continue when the user's device becomes idle. Your Manifest should already contain an element for your main `Activity` class. Add the following attributes to the `activity` element to set the `screenOrientation` and `launchMode`:

```
<activity
  android:name="com.example.musicplayer.MainActivity"
  android:label="@string/app_name"
  android:launchMode="singleTop"
  android:screenOrientation="portrait" >
```

We will stick to portrait orientation for simplicity. The `launchMode` will aid the process of navigating back to the app after moving away from it. We will display a notification indicating the song currently being played, tapping the notification will take the user back to the app. We are also going to use a `Service` class for music playback. Add the following line to the project's Manifest inside the `application` element and after the `activity` element:

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

Alter the package name to suit your own and change the class name if you wish.

Step 2

Open the project's main layout file and replace its contents with the following layout:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical"
  android:background="#FF330000"
  tools:context=".MainActivity" >

  <ListView
    android:id="@+id/song_list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
  </ListView>

</LinearLayout>
```

Makes sure to alter the **tools:context** attribute if your main `Activity` class is named differently. The layout includes a `ListView` in which we will present the list of songs.

We are going to include two menu items for toggling the shuffle function and for exiting the app. Open your main menu file (**res/menu/main.xml**) and replace its contents with the following:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

  <item
    android:id="@+id/action_shuffle"
    android:icon="@drawable/rand"
    android:orderInCategory="1"
    android:showAsAction="always"
    android:title="Shuffle"/>

  <item
    android:id="@+id/action_end"
    android:icon="@drawable/end"
    android:orderInCategory="2"
```

```
    android:showAsAction="always"
    android:title="End"/>

</menu>
```

If you prefer, you can store the title strings in the **res/values/strings.xml** file. The two items refer to drawable files. Create your own or use these two images to start with:

We will also use an icon to display in the playback notification. Create one now or use the one below:

The code will refer to the images using the names **rand**, **end**, and **play** so make sure that you use the same file names. Copy the images to your project's drawables folder(s). We will implement the actions later.

2. Query the Device for Songs

Step 1

Let's query the user's device for audio files. First, add a new class to your project, naming it `Song`. We will use this class to model the data for a single audio file. Inside the class declaration, add three instance variables for the data we want to store for each track:

```
private long id;
private String title;
private String artist;
```

Next, add a constructor method in which we instantiate the instance variables:

```
public Song(long songID, String songTitle, String songArtist) {
  id=songID;
  title=songTitle;
  artist=songArtist;
}
```

Finally, add **get** methods for the instance variables:

```
public long getID(){return id;}
public String getTitle(){return title;}
public String getArtist(){return artist;}
```

If you plan to use more track information, then you are free to add additional instance variables to the class.

Step 2

Open the main `Activity` class and add the following imports:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import android.net.Uri;
import android.content.ContentResolver;
import android.database.Cursor;
import android.widget.ListView;
```

Declare the following instance variables before the `onCreate` method:

```
private ArrayList<Song> songList;
private ListView songView;
```

We will store the songs in a list and display them in the `ListView` instance in the main layout. In `onCreate`, after setting the content view, retrieve the `ListView` instance using the ID we gave it in the main layout:

```
songView = (ListView)findViewById(R.id.song_list);
```

Instantiate the list as shown below:

```
songList = new ArrayList<Song>();
```

Next, in the main `Activity` class declaration, after the existing methods, create a helper method to retrieve the audio file information:

```
public void getSongList() {
  //retrieve song info
}
```

Inside this method, create a `ContentResolver` instance, retrieve the URI for external music files, and create a `Cursor` instance using the `ContentResolver` instance to query the music files:

```
ContentResolver musicResolver = getContentResolver();
Uri musicUri = android.provider.MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
Cursor musicCursor = musicResolver.query(musicUri, null, null, null, null);
```

Now we can iterate over the results, first checking that we have valid data:

```
if(musicCursor!=null && musicCursor.moveToFirst()){
  //get columns
  int titleColumn = musicCursor.getColumnIndex
    (android.provider.MediaStore.Audio.Media.TITLE);
  int idColumn = musicCursor.getColumnIndex
    (android.provider.MediaStore.Audio.Media._ID);
  int artistColumn = musicCursor.getColumnIndex
    (android.provider.MediaStore.Audio.Media.ARTIST);
  //add songs to list
  do {
    long thisId = musicCursor.getLong(idColumn);
    String thisTitle = musicCursor.getString(titleColumn);
    String thisArtist = musicCursor.getString(artistColumn);
    songList.add(new Song(thisId, thisTitle, thisArtist));
```

```
    }
    while (musicCursor.moveToNext());
}
```

We first retrieve the column indexes for the data items that we are interested in for each song, then we use these to create a new `Song` object and add it to the list, before continuing to loop through the results.

Back in `onCreate`, after the code we added, call this new method:

```
getSongList();
```

3. Display the Songs

Step 1

Now we can display the list of songs in the user interface. In the `onCreate` method, after calling the helper method we created a moment ago, let's sort the data so that the songs are presented alphabetically:

```
Collections.sort(songList, new Comparator<Song>(){
  public int compare(Song a, Song b){
    return a.getTitle().compareTo(b.getTitle());
  }
});
```

We use the `title` variable in the `Song` class, using the **get** methods we added, to implement a `compare` method, sorting the songs by title.

Step 2

Let's define a layout to represent each song in the list. Add a new file to your project's **res/layout** folder, naming it **song.xml** and entering the following:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:onClick="songPicked"
  android:orientation="vertical"
  android:padding="5dp" >

  <TextView
    android:id="@+id/song_title"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="#FFFFFF99"
    android:textSize="20sp"
    android:textStyle="bold" />

  <TextView
    android:id="@+id/song_artist"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="#FFFFFF99"
    android:textSize="18sp" />

</LinearLayout>
```

Feel free to amend the layout to suit your preferences. Each song in the list will be represented by title and artist text strings, so we will use the `TextViews` to display this data. Notice that the `LinearLayout` opening tag lists an `onClick` attribute. We will use this method in the main `Activity` class to respond to user taps on the songs in the list, playing the song represented by the list item that was tapped.

Step 3

We will use an `Adapter` to map the songs to the list view. Add a new class to your app, naming it **SongAdapter** or another name of your choice. When creating the class, give it the superclass `android.widget.BaseAdapter`. Eclipse should insert the following outline:

```
01 public class SongAdapter extends BaseAdapter {
02
03   @Override
04   public int getCount() {
05     // TODO Auto-generated method stub
06     return 0;
07   }
08
09   @Override
10   public Object getItem(int arg0) {
11     // TODO Auto-generated method stub
12     return null;
13   }
14
15   @Override
16   public long getItemId(int arg0) {
17     // TODO Auto-generated method stub
18     return 0;
19   }
20
21   @Override
22   public View getView(int arg0, View arg1, ViewGroup arg2) {
23     // TODO Auto-generated method stub
24     return null;
25   }
26
27 }
```

You'll need to add the following imports:

```
1 import java.util.ArrayList;
2 import android.content.Context;
3 import android.view.LayoutInflater;
4 import android.widget.LinearLayout;
5 import android.widget.TextView;
```

Inside the class declaration, declare the following instance variables:

```
1 private ArrayList<Song> songs;
2 private LayoutInflater songInf;
```

We'll pass the song list from the main `Activity` class and use the `LayoutInflater` to map the title and artist strings to the `TextViews` in the song layout we created.

After the instance variables, give the adapter a constructor method to instantiate them:

```
1 public SongAdapter(Context c, ArrayList<Song> theSongs){
2   songs=theSongs;
```

```
    songInf=LayoutInflater.from(c);
  }
```

Alter the content of the `getCount` method to return the size of the list:

```
@Override
public int getCount() {
  return songs.size();
}
```

You can leave the `getItem` and `getItemId` methods untouched. Update the implementation of the `getView` method as shown below:

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
  //map to song layout
  LinearLayout songLay = (LinearLayout)songInf.inflate
      (R.layout.song, parent, false);
  //get title and artist views
  TextView songView = (TextView)songLay.findViewById(R.id.song_title);
  TextView artistView = (TextView)songLay.findViewById(R.id.song_artist);
  //get song using position
  Song currSong = songs.get(position);
  //get title and artist strings
  songView.setText(currSong.getTitle());
  artistView.setText(currSong.getArtist());
  //set position as tag
  songLay.setTag(position);
  return songLay;
}
```

We set the title and artist text by retrieving the correct `Song` instance from the list using the position index, mapping these strings to the views we added to the song layout file. We also set the position as the view tag, which will let us play the correct song when the user clicks an item in the list. Remember that the **song.xml** layout file included an `onClick` attribute. We will use the method listed there to retrieve the tag in the `Activity`.

Step 3

Back in the main `Activity` class, in the `onCreate` method after sorting the list, create a new instance of the `Adapter` class and set it on the `ListView`:

```
SongAdapter songAdt = new SongAdapter(this, songList);
songView.setAdapter(songAdt);
```

When you run the app, it should present the list of songs on the device, clicking them will cause the app to throw an exception at the moment, but we will implement the click handler in the next tutorial.

Conclusion

We've now set the app up to read songs from the user device. In the next part, we will begin playback when the user selects a song using the `MediaPlayer` class. We will implement playback using a `Service` class so that it will continue as the user interacts with other apps. Finally, we will use a `MediaController` class to give the user control over playback.

If you're ever in need of extra help with your Android app development projects, you can find experienced Android developers on Envato Studio to help you with everything from UI design to creating a native Android app.

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

Download Attachment

OR SIGN UP WITH DISQUS ?

Name

Vyshnav Ramesh • 7 months ago

For all new visitors who find errors as mentioned below is due to the the new versions of Android (M and N as of now).

Solution:
First add this in Mainfest:
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

Then add this in Main Activity just after 'setContentView(R.layout.activity_main)';
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
if (checkSelfPermission(Manifest.permission.READ_EXTERNAL_STORAGE)
!= PackageManager.PERMISSION_GRANTED) {

requestPermissions(new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},1);

// MY_PERMISSIONS_REQUEST_READ_EXTERNAL_STORAGE is an
// app-defined int constant

return;
}}

Please note to run the output in phone so as to see the list of songs. If output is run on emulator you will only see a blank screen since the external storage we read is of phone's (not computer's).
(vyshnavkr@gmail.com)
5 ^ v • Reply • Share ›

abhishek ➚ Vyshnav Ramesh • 6 months ago

Love you bro <3

^ v • Reply • Share ›

Vyshnav Ramesh ➚ abhishek • 6 months ago

Hihi..same tu u bro

^ v • Reply • Share ›

shanky dazzler ➚ Vyshnav Ramesh • 4 months ago

i need a simple android project which i can show on my macbook like a small music or video player, i need to show every details of it like documentation, codes and layout, can u help me?

^ v • Reply • Share ›

Dave Park ➚ Vyshnav Ramesh • 6 months ago

But how about for instantiation code like adding songs to the array, sorting the songs, and the adapter instantiation all within onCreate()? Where do those code go considering that you included a return statement within the control flow statements (you posted above)?

Because as of now, I'd need to restart the app in order for the permissions to go through as well as rendering out the song list layout.

Thanks!

^ v • Reply • Share ›

Vyshnav Ramesh ➚ Dave Park • 6 months ago

Hi Dave, I have detailed it in part 2 or 3 of this series in comment section. You may get it by checking the latest comments there

^ v • Reply • Share ›

Stabja Hazra • 2 years ago

Where is the next part ?
5 • Reply • Share ›

Midhun ➚ Stabja Hazra • 2 years ago
code.tutsplus.com/tutorials...
1 • Reply • Share ›

Midhun ➚ Midhun • 2 years ago
and the next part : http://code.tutsplus.com/tu...
1 • Reply • Share ›

Samuel Ndhlovu • 3 years ago
i have completed,on the emulator it says,the applicaton closed unexpectedly,tried it on real device it can't even launch.please help
7 • Reply • Share ›

EngineerKunle ➚ Samuel Ndhlovu • 3 years ago
Just add the following:

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

<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
5 • Reply • Share ›

Srimoyee Sarkar ➚ EngineerKunle • 2 years ago
I have added the permissions in the manifest but still it is crashing..!!
1 • Reply • Share ›

slimsim ➚ Srimoyee Sarkar • 2 years ago
If you have "targetSdkVersion 23" or higher, the quick-fix is to lower it to you can lower it to 22, The good fix is to ask for permission when you need it:
https://developer.android.c...
3 • Reply • Share ›

Piyush ➚ slimsim • 2 years ago
make sure you add those permission outside the application tag, direct child of manifest tag
1 • Reply • Share ›

Sean ➚ EngineerKunle • a year ago
The second and third permissions here both look identical to each other. Do I really need both of them?

EDIT: answered my own question. I only needed the READ_EXTERNAL_STORAGE element once.
• Reply • Share ›

Yuvaraj Rajamanickam Coimbator ➚ Samuel Ndhlovu • 2 years ago
no probs in internet you ll get many codes.
• Reply • Share ›

Nguyễn Hoài Nam ➚ Samuel Ndhlovu • 3 years ago
Please notice that this is the first part of the whole application. And the most important class is created in the second part (MusicService.java), so you should follow the next parts to create the complete tutorial.
• Reply • Share ›

abdulwaheed ➚ Nguyễn Hoài Nam • 2 years ago
It is very nice but where is the second part?
3 • Reply • Share ›

Avata This comment was deleted.

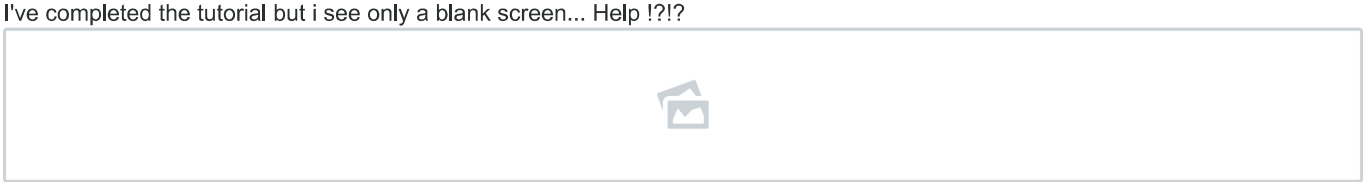
Cryptonix ➚ Guest • 2 years ago

Hi Pho Tran ... i have been building a internet radio app and have successfully implemented all the playback function to it .. can you tell me how do i retrieve the album art for the music currently being played on the live stream provided that the music title is present

Reply • Share ›

Roger Wu • 3 years ago

I've completed the tutorial but i see only a blank screen... Help !?!?



2 Reply • Share ›

gR33D 99 ➚ Roger Wu • 2 years ago

did you solve it i have the same problem

1 Reply • Share ›

Aamir Shahzad ➚ Roger Wu • 2 years ago

I was facing the same issue and finally solved when the app was run by unplugging the device from system

Reply • Share ›

gR33D 99 ➚ Aamir Shahzad • 2 years ago

i didnt get that having the same problem could you help please

Reply • Share ›

EngineerKunle ➚ Roger Wu • 3 years ago

The best thing I can advise.. is to download the source code and compare your code with this one.. or before you do that... do this

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

Put this permission in your manifest file.

Happy coding :)

Reply • Share ›

General Tso • 3 years ago

Just a heads up, the app required "READ_EXTERNAL_STORAGE" permission to work, at least on Android L preview

2 Reply • Share ›

Gordeych • 3 years ago

Hi Sue! Greate article.
Please help me. How play music from Fragment?

2 Reply • Share ›

Sharang • 3 years ago

I did the first part of project setup and got an error which stated that the MusicPlayer has stopped. Help, please?

2 Reply • Share ›

disqus_hBQgnLQLhM ➚ Sharang • a year ago

look at method onCreate. this two rows must be the first and the second:
1) setContentView(R.layout.activity_main);
2) super.onCreate(savedInstanceState);
This error ocurre because you have findViewById before setContentView.

1 Reply • Share ›

Harry Martin • 2 years ago

Thats great tutorial found here. I also recommend you to learn in creating Soundcloud like music sharing from online experts. You can also get more details here @ http://learnsauce.com/sound...

3 ^ v • Reply • Share ›

CMD Dzeko • 3 months ago

how to add res/menu /main.xml.. im start my new project in Android Studio as a Blank activity..
anyone can help.?

1 ^ v • Reply • Share ›

Joanna ➚ CMD Dzeko • 2 months ago

I found the simple workaround. I create a file and a directory manually. Then AndroidStudio sees it and I can edit the file.

^ v • Reply • Share ›

Mithun • a year ago

how to download this application source code

1 ^ v • Reply • Share ›

Saswat • a year ago

I completed this tutorial on the AIDE app in android.
When I use the findViewById(R.id.song_title or song_artist) it returns an error.
I am not able to understand what went wrong. Can anyone help

1 ^ v • Reply • Share ›

ana • 3 years ago

Hi Sue! Thanks for this tutorial, it's really helpful for someone just starting out with android. I'm having a problem though, every time I run the first part of the app (just trying to get the list with song names to come up), the app stops working and crashes. Based on other comments i added the android permissions:

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

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

However, the problem still repeats :/ .

Anyone have any idea why this could be happening? I'm using android studio IDE and a samsung S3 if that's helpful to know, and I follow the above code and steps.

Please let me know

1 ^ v • Reply • Share ›

Aro ➚ ana • 3 years ago

Add below in the manifest file.

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

^ v • Reply • Share ›

Xenolion • 2 days ago

Ooooh God this woman Sue Smith my love!.....it was helpfuuu

Ishank Sharma • 25 days ago

MY compareTo function is not working
it is not inbuilt fuction
so where is the implementation..

Further in writing
public String getTitle();

it is saying cannot override
as its is inbuilt final type

Reply • Share ›

Fawad khan • 2 months ago

can someone provide link to the source code? i would really appreciate it

Reply • Share ›

Pedro Duarte • 3 months ago

Hi everyone i try to run my app my code looks right but in the debug get error

FATAL EXCEPTION: main
Process: com.pedroduarte.music, PID: 26129
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.pedroduarte.music/com.pedroduarte.music.MainActivity}:
android.view.InflateException: Binary XML file line #15: Error inflating class menu
//----------//
Here the MainActivity code
//----------
public class MainActivity extends AppCompatActivity {

private ArrayList<song> songList;
private ListView songView;

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

see more

Reply • Share ›

Code Lord • 4 months ago

how can I retrieve the images for the song, pls help

Reply • Share ›

johan pf • 4 months ago

I've completed the tutorial but .....

,,, help me...

Reply • Share ›

CMD Dzeko ➚ johan pf • 3 months ago

this is an halfhearted tutorial lol

Reply • Share ›

Aswa • 4 months ago

Link to the next part please?

Reply • Share ›

Vignesh Iyer • 4 months ago

The app was crashing so I changed Cursor musicCursor = musicResolver.query(musicUri,null,null,null,null) to Cursor musicCursor = null as adviced in the comments below but then blank screen started to show up.... please help!!

Reply • Share ›

Cenk Camkıran ➚ Vignesh Iyer • 8 days ago

did you find the solution?Please help

Reply • Share ›

Kurt Bowes • 5 months ago

Anyone get the message that this class must be declared abstract? I followed tutorial to the letter so confused.

Reply • Share ›

Paperwrk Labs ➚ Kurt Bowes • 4 months ago

Just hover over it and implement the require methods!

Reply • Share ›

Load more comments