SIMPLIFIED CODING

Firebase Realtime Database CRUD Operation for Android

February 28, 2017 by Belal Khan — 18 Comments

Hey guys here is the complete **Firebase Realtime Database** guide. Now days NoSQL databases are gaining popularity and **Firebase Realtime Database** is one of the NoSQL database.

In this tutorial we will learn modeling SQL tables to NoSQL Firebase Database. And then we will also learn the basic CRUD operation on the database **(Create, Read, Update and Delete)**.

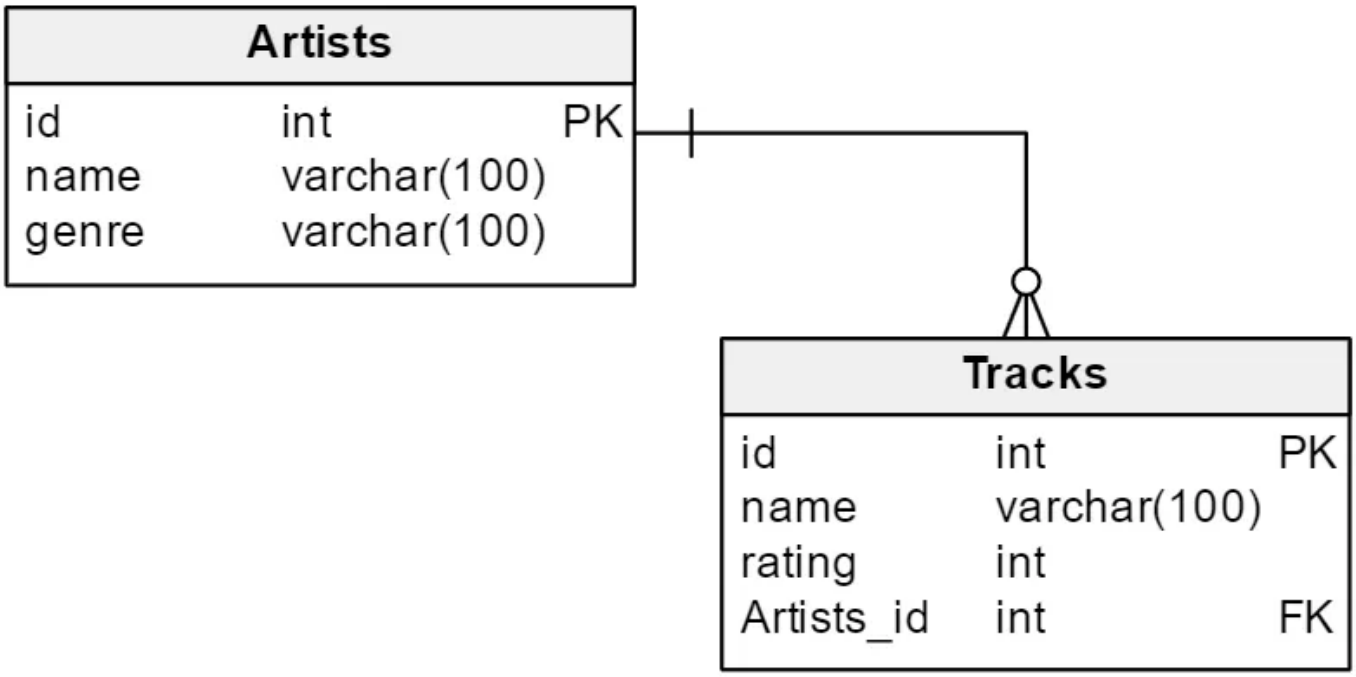
Contents [hide]

Firebase Realtime Database Video Tutorial

SIMPLIFIED CODING

Designing Database

- For this example I have following two tables.



- But firebase is not an SQL database and it does not stores data in tabular format. It uses JSON tree structure. So for firebase realtime database the structure for above database will be.

```
        },
        "artist_id_2":{
            "name":"Arijit Singh",
            "genre":"Rock"
        }
    },

    "Tracks":{
        "artist_id_1":{
            "track_id_1":{
                "name":"Taj dar e haram",
                "rating":5
            }
        }
    }
}
```


- I hope you got a clear understanding of the Firebase Realtime Database structure now. So lets jump into Android Studio.

Firebase Realtime Database Basics

- You have the database structure now you need to know the basic operations. So lets understand the steps of handling firebase database.

Getting Database Reference

SIMPLIFIED CODING

```
FirebaseDatabase.getInstance().getReference("path"); //Dont pass any path if you want root of the tree
```

- The data is stored in the JSON Tree form so you need to get the reference of an specified path. Like in the above database we can get all the Artists by passing "Artists". If you want to access everything don't pass anything and it will create a reference of the root of the tree.

Write Operation

- **setValue() –** This method will take a model java class object that will hold all the variables to be stored in the reference. The same method will be used to update the values at it overwrites the data of the specified reference.
- Suppose we have to store an Artist to our reference then we will create a model class as below.

Java

```java
@IgnoreExtraProperties
public class Artist {
    private String artistId;
    private String artistName;
    private String artistGenre;

    public Artist(){

    }

    public Artist(String artistId, String artistName, String artistGenre) {
        this.artistId = artistId;
        this.artistName = artistName;
        this.artistGenre = artistGenre;
    }

    public String getArtistId() {
        return artistId;
    }

    public String getArtistName() {
        return artistName;
    }

    public String getArtistGenre() {
        return artistGenre;
    }
}
```

- The update operation will also be done in the same way.

Read Operation

- We will attache a **ValueEventListener** to the reference to read the data.

```java
databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {

            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
```

- Whenever you will change something in the Database the method **onDataChange()** will be executed. It contains all the data inside the specified path in the reference. We can use the **DataSnapshot** object to read all the data inside the reference. If some error occurres **onCancelled()** method will be called.
- **onDataChange()** method will also called once after the app launch and hence you can read the data at starting as well.

Delete Operation

- **removeValue()** can be used to delete the data.
- Now lets understand the operations in an Android Project.

Firebase Realtime Database Project

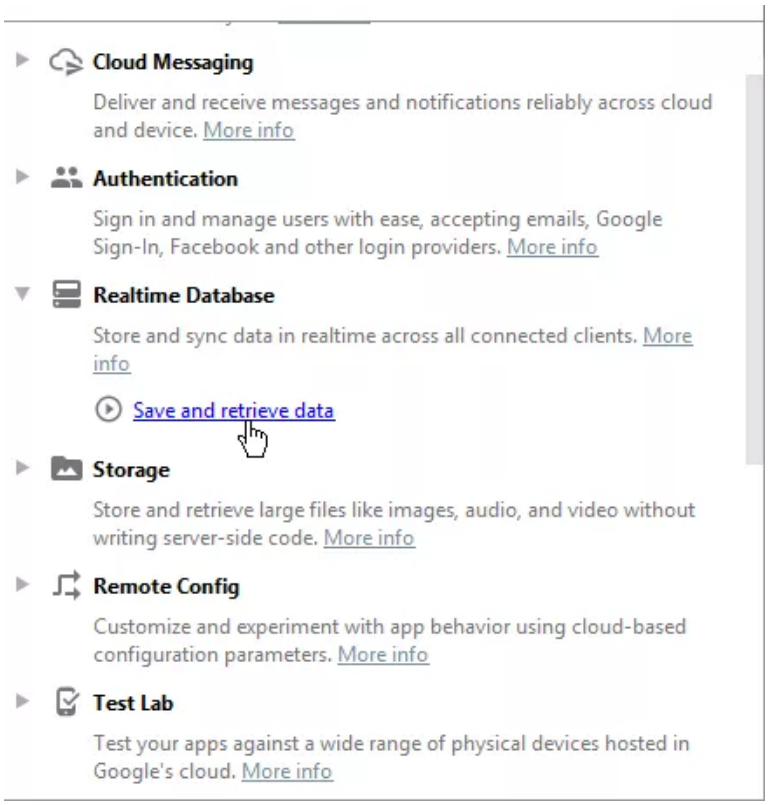
Creating Android Studio Project

- Open Android Studio and create a new project. In my case I have created **FirebaseDatabaseExample**.
- Now once your project is loaded completely, add Firebase Database in it.

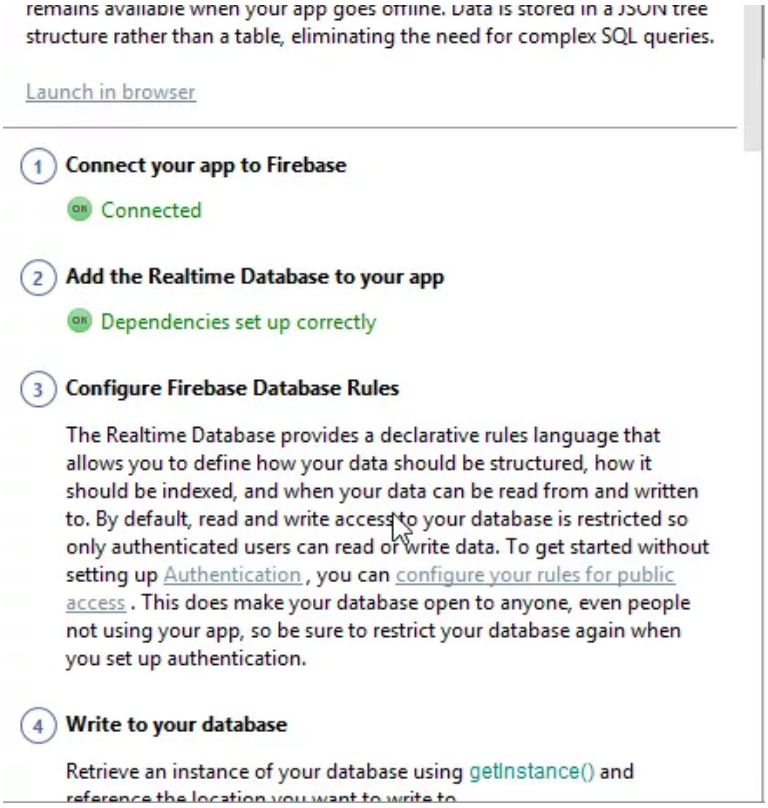
Adding Firebase Database

- Go to **tools -> Firebase,** it will open an assistant. Now from the assistant go to Realtime Database.

SIMPLIFIED CODING



- Connect to your Firebase Project and Setup the dependencies.

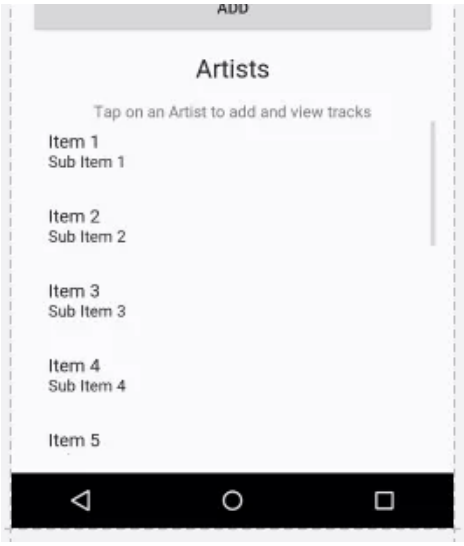


- Now you are ready to use Firebase Database in your project.

Creating Activity Layouts

- We need two activities. One is to add **Artists** and other one is to add **Tracks** to database. We will use **MainActivity.java and activity_main.xml** for Artists. But you need to create one more activity for Tracks. So I have created **ArtistActivity and activity_artist.xml.**
- Now first come inside **activity_main.xml**. Here we will create the following layout.

SIMPLIFIED CODING



- As you can see the layout contains the following thing.
  EditText: Here we will enter the name.
  Spinner: Here we will give some genres so that we can select one of them.
  Button: To save the new artist in Firebase.
  ListView: To display all the saved artist from Firebase.
- For creating this layout you can use the following xml code.

activity_main.xml

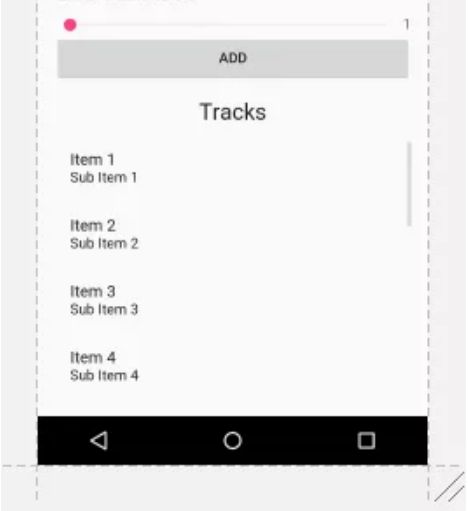
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="net.simplifiedcoding.firebasedatabaseexample.MainActivity">


    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter name" />

    <Spinner
```

SIMPLIFIED CODING

```
    <Button
        android:id="@+id/buttonAddArtist"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/spinnerGenres"
        android:text="Add" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/buttonAddArtist"
        android:padding="@dimen/activity_horizontal_margin"
        android:text="Artists"
        android:textAlignment="center"
        android:textAppearance="@style/Base.TextAppearance.AppCompat.Large" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/textView"
        android:text="Tap on an Artist to add and view tracks"
        android:textAlignment="center" />

    <ListView
        android:id="@+id/listViewArtists"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"></ListView>

</RelativeLayout>
```

- For **ArtistsActivity** we will create almost the same layout with little changes. So this activity will be like.

SIMPLIFIED CODING



- For this activity you can again use the following xml.

activity_artist.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:id="@+id/activity_artist"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="net.simplifiedcoding.firebasedatabaseexample.ArtistActivity">

    <TextView
        android:id="@+id/textViewArtist"
        android:padding="@dimen/activity_horizontal_margin"
        android:textAlignment="center"
        android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"
        android:textStyle="bold"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <EditText
        android:layout_below="@id/textViewArtist"
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter track name" />
```

SIMPLIFIED CODING

```
        android:layout_below="@id/editTextName">

        <SeekBar
            android:layout_weight="1"
            android:id="@+id/seekBarRating"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:max="5"></SeekBar>

        <TextView
            android:text="1"
            android:id="@+id/textViewRating"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

    </LinearLayout>


    <Button
        android:id="@+id/buttonAddTrack"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/linearLayout"
        android:text="Add" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/buttonAddTrack"
        android:padding="@dimen/activity_horizontal_margin"
        android:text="Tracks"
        android:textAlignment="center"
        android:textAppearance="@style/Base.TextAppearance.AppCompat.Large" />

    <ListView
        android:id="@+id/listViewTracks"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView"></ListView>

</RelativeLayout>
```

SIMPLIFIED CODING

- Create a java class named **Artist** and write the following code.

```java
package net.simplifiedcoding.firebasedatabaseexample;

import com.google.firebase.database.IgnoreExtraProperties;

/**
 * Created by Belal on 2/26/2017.
 */
@IgnoreExtraProperties
public class Artist {
    private String artistId;
    private String artistName;
    private String artistGenre;

    public Artist(){
        //this constructor is required
    }

    public Artist(String artistId, String artistName, String artistGenre) {
        this.artistId = artistId;
        this.artistName = artistName;
        this.artistGenre = artistGenre;
    }

    public String getArtistId() {
        return artistId;
    }

    public String getArtistName() {
        return artistName;
    }

    public String getArtistGenre() {
        return artistGenre;
    }
}
```

- Now again create one more class named **Track** and write the following code.

Track.java Java

```java
package net.simplifiedcoding.firebasedatabaseexample;

```

SIMPLIFIED CODING

```
public class Track {
    private String id;
    private String trackName;
    private int rating;

    public Track() {

    }

    public Track(String id, String trackName, int rating) {
        this.trackName = trackName;
        this.rating = rating;
        this.id = id;
    }

    public String getTrackName() {
        return trackName;
    }

    public int getRating() {
        return rating;
    }
}
```

- Now lets save an artist to the database.

Saving an Artist

- Come inside **MainActivity.java** and write the following code.

MainActivity.java Java

```
package net.simplifiedcoding.firebasedatabaseexample;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Spinner;
```

SIMPLIFIED CODING

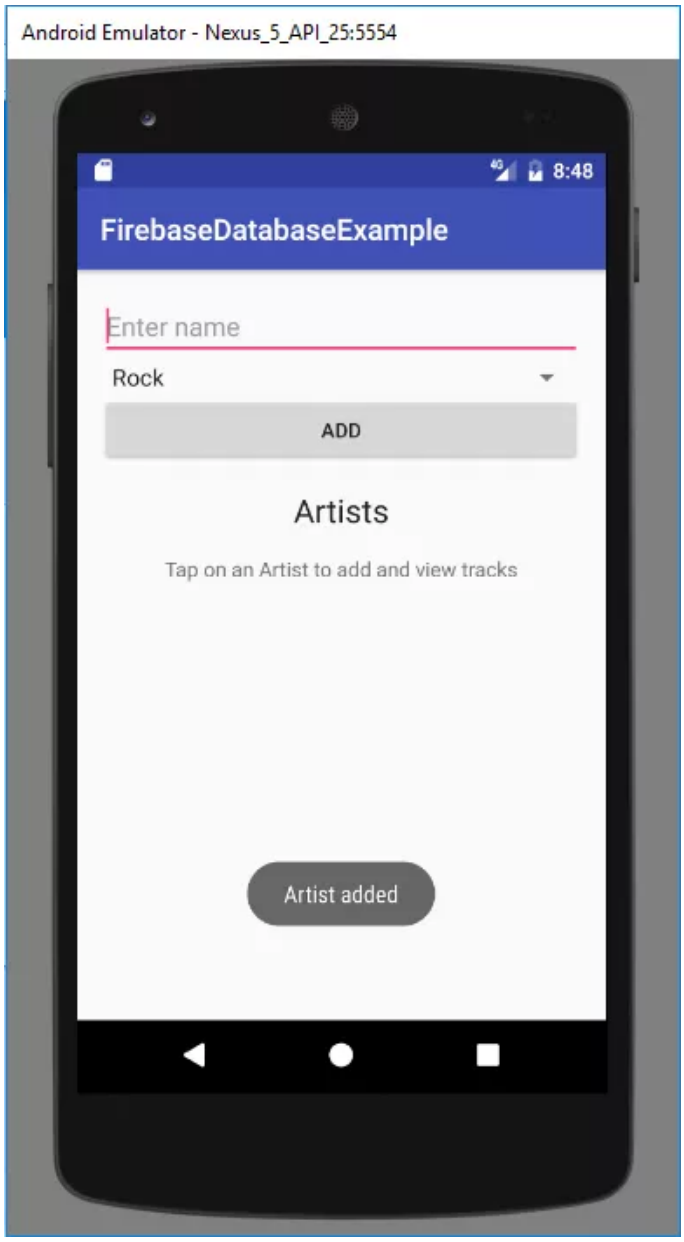
```
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    //we will use these constants later to pass the artist name and id to another activity
    public static final String ARTIST_NAME = "net.simplifiedcoding.firebasedatabaseexample.artistname";
    public static final String ARTIST_ID = "net.simplifiedcoding.firebasedatabaseexample.artistid";

    //view objects
    EditText editTextName;
    Spinner spinnerGenre;
    Button buttonAddArtist;
    ListView listViewArtists;

    //a list to store all the artist from firebase database
    List<Artist> artists;

    //our database reference object
    DatabaseReference databaseArtists;

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

        //getting the reference of artists node
        databaseArtists = FirebaseDatabase.getInstance().getReference("artists");

        //getting views
        editTextName = (EditText) findViewById(R.id.editTextName);
        spinnerGenre = (Spinner) findViewById(R.id.spinnerGenres);
        listViewArtists = (ListView) findViewById(R.id.listViewArtists);

        buttonAddArtist = (Button) findViewById(R.id.buttonAddArtist);

        //list to store artists
        artists = new ArrayList<>();


        //adding an onclicklistener to button
        buttonAddArtist.setOnClickListener(new View.OnClickListener() {
            @Override
```

SIMPLIFIED CODING

```
        });
    }

    /*
    * This method is saving a new artist to the
    * Firebase Realtime Database
    * */
    private void addArtist() {
        //getting the values to save
        String name = editTextName.getText().toString().trim();
        String genre = spinnerGenre.getSelectedItem().toString();

        //checking if the value is provided
        if (!TextUtils.isEmpty(name)) {

            //getting a unique id using push().getKey() method
            //it will create a unique id and we will use it as the Primary Key for our Artist
            String id = databaseArtists.push().getKey();

            //creating an Artist Object
            Artist artist = new Artist(id, name, genre);

            //Saving the Artist
            databaseArtists.child(id).setValue(artist);

            //setting edittext to blank again
            editTextName.setText("");

            //displaying a success toast
            Toast.makeText(this, "Artist added", Toast.LENGTH_LONG).show();
        } else {
            //if the value is not given displaying a toast
            Toast.makeText(this, "Please enter a name", Toast.LENGTH_LONG).show();
        }
    }
}
```

SIMPLIFIED CODING

- Now try running the application.



- The toast is fine now lets check the database.

SIMPLIFIED CODING

artistId: "-Ke41CVpb9KxCS1sdx4
artistName: "Atif Aslam

- So we the artist is getting save. Everytime you will save an artist a new child node inside artists will be created with a unique id.

Retrieving Artists

- Now we will fetch all the artists. We already have the **ListView** to display all the artists with genre. But before proceeding in retrieving the artists in ListView we will create a Layout File and a Custom Adapter for the list.
- So first create a new layout file named **layout_artist_list.xml** it will contain two **TextView**.

layout_artist_list.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:text="Atif Aslam"
        android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"
        android:id="@+id/textViewName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:text="Rock"
        android:textAppearance="@style/Base.TextAppearance.AppCompat.Medium"
        android:id="@+id/textViewGenre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />


</LinearLayout>
```

- Now create one more class named **ArtistList** and write the following code.

ArtistList.java Java

SIMPLIFIED CODING

```
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.List;

/**
 * Created by Belal on 2/26/2017.
 */

public class ArtistList extends ArrayAdapter<Artist> {
    private Activity context;
    List<Artist> artists;

    public ArtistList(Activity context, List<Artist> artists) {
        super(context, R.layout.layout_artist_list, artists);
        this.context = context;
        this.artists = artists;
    }


    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = context.getLayoutInflater();
        View listViewItem = inflater.inflate(R.layout.layout_artist_list, null, true);

        TextView textViewName = (TextView) listViewItem.findViewById(R.id.textViewName);
        TextView textViewGenre = (TextView) listViewItem.findViewById(R.id.textViewGenre);

        Artist artist = artists.get(position);
        textViewName.setText(artist.getArtistName());
        textViewGenre.setText(artist.getArtistGenre());

        return listViewItem;
    }
}
```

- Now lets retrieve all the artists. For this we will attach a **ValueEventListener** to the database reference object, inside **onStart()** method of MainActivity.

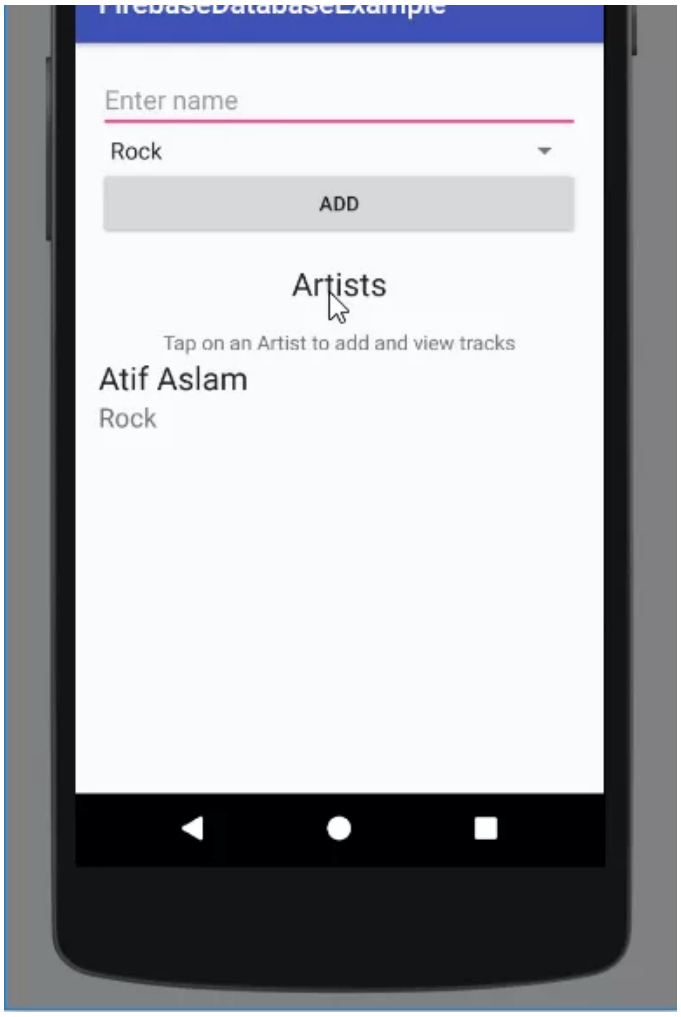
Java

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

```
                artists.clear();

                //iterating through all the nodes
                for (DataSnapshot postSnapshot : dataSnapshot.getChildren()) {
                    //getting artist
                    Artist artist = postSnapshot.getValue(Artist.class);
                    //adding artist to the list
                    artists.add(artist);
                }

                //creating adapter
                ArtistList artistAdapter = new ArtistList(MainActivity.this, artists);
                //attaching adapter to the listview
                listViewArtists.setAdapter(artistAdapter);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }
```


- Now again run the application.

SIMPLIFIED CODING



- So the Artists are getting fetched. Also if you will add new Artist it will be automatically added inside the **ListView** as the method **onDataChange()** will be executed on any data change.
- Now we will add **Tracks** to a particular Artist. For this we will open a new Activity when an Artist is selected from the ListView. For this we have to attach an **OnItemClickListener** to the **ListView**.
- So add the following code inside **onCreate()** method.

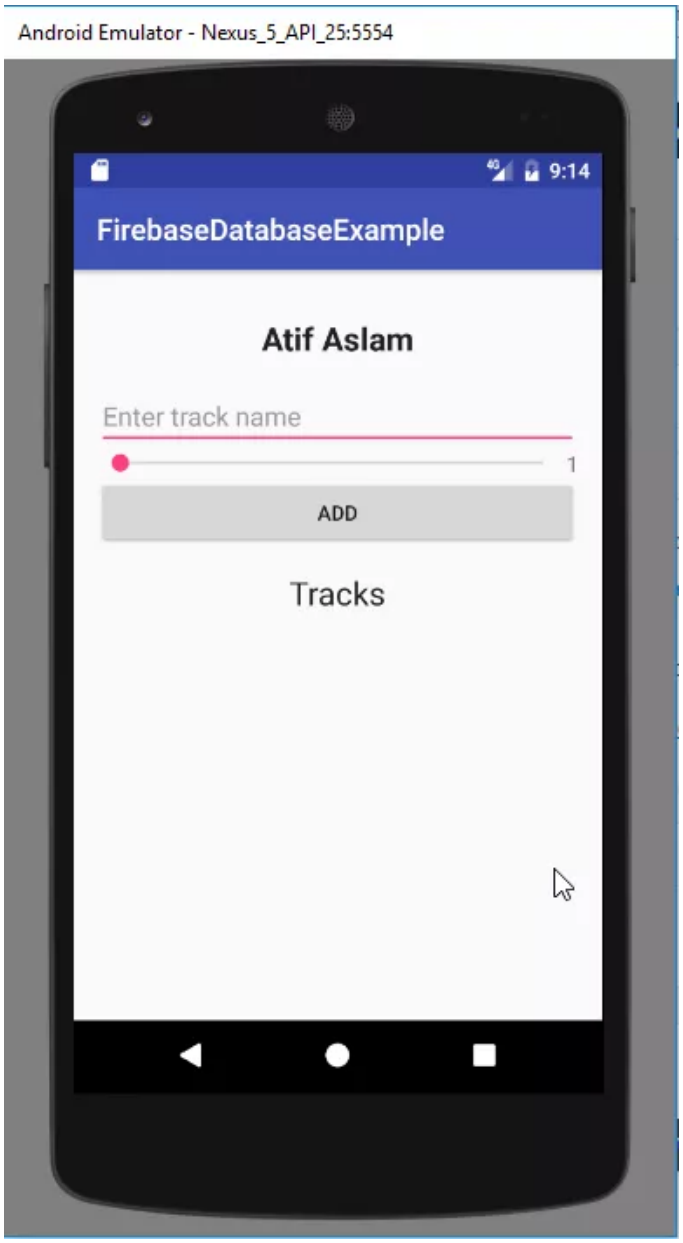
Java

```
    //attaching listener to listview
      listViewArtists.setOnItemClickListener(new AdapterView.OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
              //getting the selected artist
              Artist artist = artists.get(i);

              //creating an intent
              Intent intent = new Intent(getApplicationContext(), ArtistActivity.class);
```

SIMPLIFIED CODING

```
                startActivity(intent);
            }
        });
```

- Now run your application and select an Artist from the List. A new activity will open displaying the artist name.



- Now lets proceed in adding and retrieving tracks. But before we need to create one more class for the adapter of Track List.
- So create a new class named **TrackList** and write the following code.

```
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.List;

/**
 * Created by Belal on 2/26/2017.
 */

public class TrackList extends ArrayAdapter<Track> {
    private Activity context;
    List<Track> tracks;

    public TrackList(Activity context, List<Track> tracks) {
        super(context, R.layout.layout_artist_list, tracks);
        this.context = context;
        this.tracks = tracks;
    }


    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = context.getLayoutInflater();
        View listViewItem = inflater.inflate(R.layout.layout_artist_list, null, true);

        TextView textViewName = (TextView) listViewItem.findViewById(R.id.textViewName);
        TextView textViewRating = (TextView) listViewItem.findViewById(R.id.textViewGenre);

        Track track = tracks.get(position);
        textViewName.setText(track.getTrackName());
        textViewRating.setText(String.valueOf(track.getRating()));

        return listViewItem;
    }
}
```

- Now lets add and retrieve tracks.

Adding and Retrieving Tracks from Firebase Realtime Database

- Come inside **ArtistActivity.java** and write the following code. The process is same as we did above that is why I am not explaining the codes with comments.

SIMPLIFIED CODING

```
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class ArtistActivity extends AppCompatActivity {

    Button buttonAddTrack;
    EditText editTextTrackName;
    SeekBar seekBarRating;
    TextView textViewRating, textViewArtist;
    ListView listViewTracks;

    DatabaseReference databaseTracks;

    List<Track> tracks;

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

        Intent intent = getIntent();

        /*
        * this line is important
        * this time we are not getting the reference of a direct node
        * but inside the node track we are creating a new child with the artist id
        * and inside that node we will store all the tracks with unique ids
        * */
        databaseTracks = FirebaseDatabase.getInstance().getReference("tracks").child(intent.getStri
```

SIMPLIFIED CODING

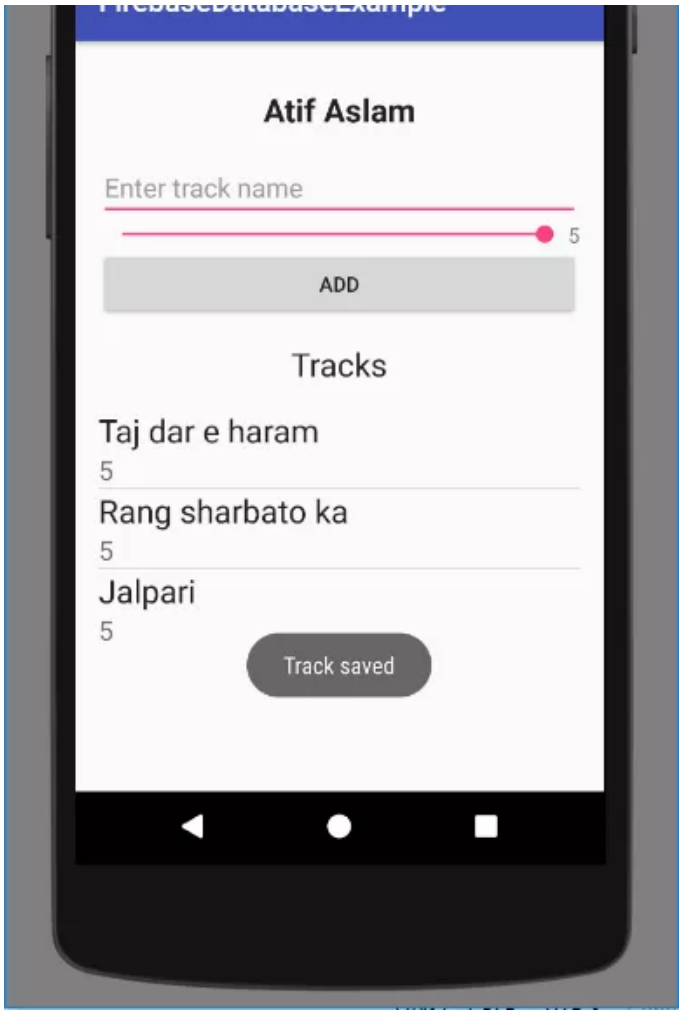
```
        listViewTracks = (ListView) findViewById(R.id.listViewTracks);

        tracks = new ArrayList<>();

        textViewArtist.setText(intent.getStringExtra(MainActivity.ARTIST_NAME));

        seekBarRating.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
                textViewRating.setText(String.valueOf(i));
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        buttonAddTrack.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                saveTrack();
            }
        });
    }

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

        databaseTracks.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                tracks.clear();
                for (DataSnapshot postSnapshot : dataSnapshot.getChildren()) {
                    Track track = postSnapshot.getValue(Track.class);
                    tracks.add(track);
                }
                TrackList trackListAdapter = new TrackList(ArtistActivity.this, tracks);
                listViewTracks.setAdapter(trackListAdapter);
```

SIMPLIFIED CODING

```
        });
    }

    private void saveTrack() {
        String trackName = editTextTrackName.getText().toString().trim();
        int rating = seekBarRating.getProgress();
        if (!TextUtils.isEmpty(trackName)) {
            String id  = databaseTracks.push().getKey();
            Track track = new Track(id, trackName, rating);
            databaseTracks.child(id).setValue(track);
            Toast.makeText(this, "Track saved", Toast.LENGTH_LONG).show();
            editTextTrackName.setText("");
        } else {
            Toast.makeText(this, "Please enter track name", Toast.LENGTH_LONG).show();
        }
    }
}
```

- Again run your application and try adding tracks.

SIMPLIFIED CODING



- As you can see tracks are getting saved. Now lets see updating the Artist.

Updating Artist in Firebase Database

- For updating an existing artist name or genre we will show a Dialog to enter new detail. Dialog will be opened on long pressing an Artist from the list.
- So first create a new layout file named **update_dialog.xml** and write the following xml code. The same dialog will be used for deleting the artist as well.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="@dimen/activity_horizontal_margin">
```

SIMPLIFIED CODING

```
        android:hint="Enter name" />

    <Spinner
        android:id="@+id/spinnerGenres"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/editTextName"
        android:entries="@array/genres"></Spinner>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/buttonUpdateArtist"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Update" />

        <Button
            android:id="@+id/buttonDeleteArtist"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Delete" />

    </LinearLayout>


</LinearLayout>
```

- As you can see we have two buttons one to update the artist and other one to delete the artist.
- Now come inside **MainActivity.java** and define a method **updateArtist()**.

Java

```
   private boolean updateArtist(String id, String name, String genre) {
        //getting the specified artist reference
        DatabaseReference dR = FirebaseDatabase.getInstance().getReference("artists").child(id);

        //updating artist
        Artist artist = new Artist(id, name, genre);
        dR.setValue(artist);
```

SIMPLIFIED CODING

and on update button click we are calling the above method **updateArtist()** to update the artist.

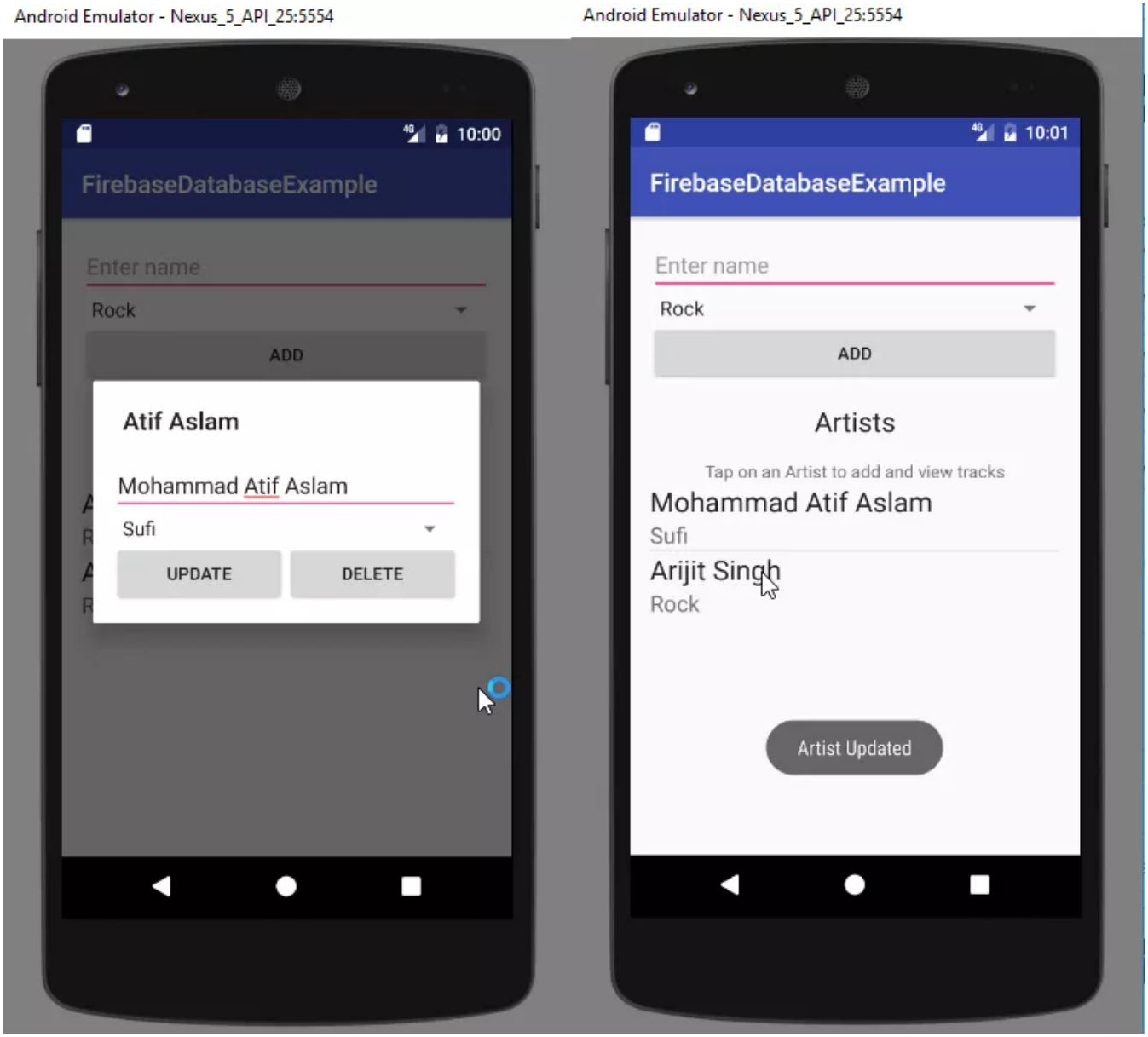
Java

```java
    private void showUpdateDeleteDialog(final String artistId, String artistName) {

        AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);
        LayoutInflater inflater = getLayoutInflater();
        final View dialogView = inflater.inflate(R.layout.update_dialog, null);
        dialogBuilder.setView(dialogView);

        final EditText editTextName = (EditText) dialogView.findViewById(R.id.editTextName);
        final Spinner spinnerGenre = (Spinner) dialogView.findViewById(R.id.spinnerGenres);
        final Button buttonUpdate = (Button) dialogView.findViewById(R.id.buttonUpdateArtist);
        final Button buttonDelete = (Button) dialogView.findViewById(R.id.buttonDeleteArtist);

        dialogBuilder.setTitle(artistName);
        final AlertDialog b = dialogBuilder.create();
        b.show();


        buttonUpdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String name = editTextName.getText().toString().trim();
                String genre = spinnerGenre.getSelectedItem().toString();
                if (!TextUtils.isEmpty(name)) {
                    updateArtist(artistId, name, genre);
                    b.dismiss();
                }
            }
        });


        buttonDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                /*
                * we will code this method to delete the artist
                * */

            }
        });
    }
```

SIMPLIFIED CODING

```
            @Override
            public boolean onItemLongClick(AdapterView<?> adapterView, View view, int i, long l) {
                Artist artist = artists.get(i);
                showUpdateDeleteDialog(artist.getArtistId(), artist.getArtistName());
                return true;
            }
        });
```

- Now again try running the application. Long press on an Artist and try updating it.



- So the update is working fine. Now only the delete operation is remaining. So lets delete an Artist.

SIMPLIFIED CODING

Java

```java
    private boolean deleteArtist(String id) {
        //getting the specified artist reference
        DatabaseReference dR = FirebaseDatabase.getInstance().getReference("artists").child(id);

        //removing artist
        dR.removeValue();

        //getting the tracks reference for the specified artist
        DatabaseReference drTracks = FirebaseDatabase.getInstance().getReference("tracks").child(id

        //removing all tracks
        drTracks.removeValue();
        Toast.makeText(getApplicationContext(), "Artist Deleted", Toast.LENGTH_LONG).show();

        return true;
    }
```

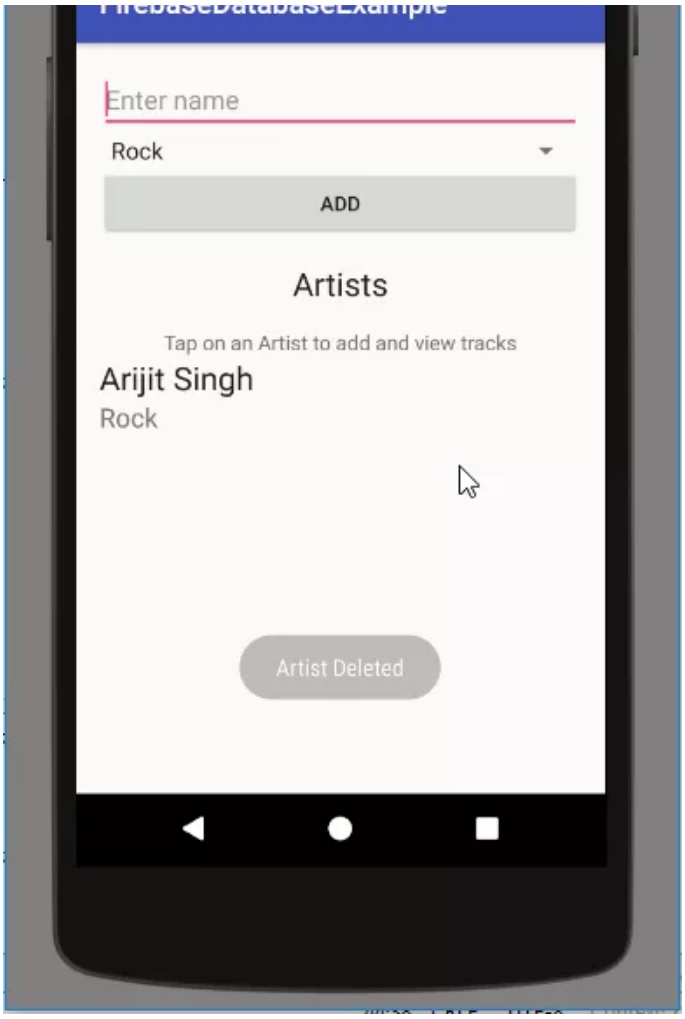
- Now call this method inside Click Listener of Delete Artist Button which is inside **showUpdateDeleteDialog()**.

Java

```java
        buttonDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                deleteArtist(artistId);
                b.dismiss();
            }
        });
```

- Now run your application and try deleting an Artist.

SIMPLIFIED CODING



- Bingo! the delete is working fine as well. If you are having troubles about this **Firebase Realtime Database Tutorial** you can get my source code from below.

So thats all for this Firebase Realtime Database tutorial. We covered all the operations like **Creating, Reading, Updating and Deleting (CRUD) data in Firebase Realtime Database**. If you are having any queries and confusions then lets discuss in comment section.

SIMPLIFIED CODING

Related

Android Firebase Tutorial - User Registration with Authentication
July 15, 2016
In "Android Advance"

Firebase Storage Example - Uploading and Retrieving Files
February 24, 2017
In "Android Advance"

Firebase Cloud Messaging for Android using PHP and MySQL
November 5, 2016
In "Android Advance"

Some more tutorials for you

1. **Login With Facebook Android Studio using Facebook SDK 4**
2. **Android Login Example Using PHP, MySQL and Volley**
3. **Android Paypal Integration Tutorial**
4. **Retrofit Upload File Tutorial – Uploading and Downloading Images**
5. **Android Firebase Tutorial – User Registration with Authentication**
6. **Firebase Storage Example – Uploading and Retrieving Files**

SIMPLIFIED CODING

Filed Under: Android Advance, Android Application Development
Tagged With: firebase, firebase android, firebase database, firebase realtime database

About Belal Khan

I am Belal Khan, I am currently pursuing my MCA. In this blog I write tutorials and articles related to coding, app development, android etc.

Comments

Waqas Khalid says
March 2, 2017 at 6:44 am

Thanks a lot for the wonderful tutorial. Great Explanation. You made this look really easy.

Reply

Avinash Manohar says

SIMPLIFIED CODING

Reply

abdolati ali says
March 25, 2017 at 12:15 pm

Thank you.

Reply

Kıvanç says
March 26, 2017 at 6:30 pm

Thanks for tutorial Belal Khan.
I have a question: I want to make a search (android.support.v7.widget.SearchView) on this realtime database and I'm new on android development. I tried this;

```
@Override
@TargetApi(Build.VERSION_CODES.N_MR1)
public boolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.option_search, menu);

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN) {
SearchManager manager = (SearchManager) getSystemService(Context.SEARCH_SERVICE);
SearchView search = (SearchView) menu.findItem(R.id.search).getActionView();
search.setSearchableInfo(manager.getSearchableInfo(getComponentName()));

search.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
@Override
public boolean onQueryTextSubmit(String query) {
return false;
}
```

```
//attaching adapter to the listview
listViewArtists.setAdapter(artistAdapter);
artistAdapter.getFilter().filter(newText);

return false;
}
});

}
return true;
}
```

but this wasn't succeed. Could you lead me about, how can we make query on our database? Thank you.

Reply

Simon says
April 12, 2017 at 10:39 pm

Hello I also interested in search option. Did you find some solution?

Reply

Ruben says
March 27, 2017 at 9:57 am

Fantastic, thanks for sharing!!! Congratulations.

Reply

Dayakar says

SIMPLIFIED CODING

fad says
May 7, 2017 at 6:01 pm

thank you for the tutorial. I want to ask a question. how can we query an artist and show it a textview (equivalent of select * from artist whe name="atif aslam")?

Reply

Monil says
May 24, 2017 at 10:13 am

Hi Belal,

I have done the same steps but databaseReference is throwing null exception.
I have posted the question in stackoverflow with code and screenshot.
Please suggest the reason
https://stackoverflow.com/questions/44155101/firebase-databasereference-is-taking-null-value-on-initialization

Reply

arnel says
July 25, 2017 at 2:05 pm

Thanks it was a great tutorial.

Reply

SIMPLIFIED CODING

Reply

Jaime says
August 30, 2017 at 1:58 pm

Your artistID es different that the node ID. Why is that?

Reply

Andre says
October 23, 2017 at 8:29 am

Hi,
Is really necessary to repeat the Key as an attribute on the object?

Reply

Saad says
November 4, 2017 at 9:54 am

can you tell me about custom login in firebase using android with source code.

Reply

Noah says
November 6, 2017 at 1:57 pm

SIMPLIFIED CODING

Reply

Utsav says
November 24, 2017 at 11:20 pm

Excellent Tutorial.

Reply

arunkuamr says
December 14, 2017 at 10:08 am

nice work,real help to me

Reply

Android Developer says
December 16, 2017 at 12:06 pm

Thanks for the wonderful tutorial!!

I was wondering what if i need to store album cover (i.e., and image) as well??? It would be a great help if you can guide.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

SIMPLIFIED CODING

Name *

Email *

CAPTCHA Code *

POST COMMENT

Notify me of follow-up comments by email.

Notify me of new posts by email.

SEARCH

Search this website ...

SIMPLIFIED CODING

ABOUT ME

SIMPLIFIED CODING

CONNECT WITH ME

Belal Khan

Follow

1,432 followers

Follow @codesimplified

Simplified Coding

Like Page

Be the first of your friends to like this

Simplified Coding

Follow

POPULAR TUTORIALS

Android JSON Parsing – Retrieve From MySQL Database

Android Login and Registration Tutorial with PHP MySQL

Android Volley Tutorial – Fetching JSON Data from URL

Android Upload Image to Server using Volley Tutorial

Android TabLayout Example using ViewPager and Fragments

Android Upload Image to Server Using PHP MySQL

SIMPLIFIED CODING

Retrieve Data From MySQL Database in Android using Volley

Android Push Notification using GCM Tutorial

ABOUT SIMPLIFIED CODING

Simplified Coding is a blog for all the students learning programming. We are providing various tutorials related to programming and application development. You can get various nice and simplified tutorials related to programming, app development, graphics designing and animation. We are trying to make these things simplified and entertaining. We are writing text tutorial and creating video and visual tutorials as well. You can check about the admin of the blog here and check out our sitemap

QUICK LINKS

Advertise Here

Privacy Policy

Disclaimer

About

Contact Us

Write for Us

CATEGORIES

Android Advance Android Application Development Android Beginners Android Intermediate Ionic Framework Tutorial JavaScript

SIMPLIFIED CODING