

Advanced Mobile Application Development

Week 15: Android and Firebase

Firebase

Firebase approaches all platforms – iOS, Android, and Web with similar patterns. The main difference is in the setup.

<https://firebase.google.com/docs/android/setup>

We're going to use the same Recipes Firebase project that we used for iOS. Since we won't be implementing authentication we need to make it public so our app can read and write.

In the Firebase console chose your database and then go into Authentication and in sign-in method disable all the sign-in providers.

Also back in Database go to the Rules tab and make sure they're public with read and write both set to true so our app can access the database.

Android Studio

New project called Recipes

Basic Activity template

RecipeMainActivity

Uncheck create fragment

Tools | Firebase to go into the Firebase Assistant

Open realtime database | Save and Retrieve data or

<https://firebase.google.com/docs/database/android/start/>

Connect to Firebase

Chose existing Recipes database or create a new one

Once connected go on to step 2 Add the Realtime Database to your app

This will add the follow dependencies and Google services plugin to your gradle files:

build.gradle (project-level)

```
Add Firebase Gradle buildscript dependency
classpath 'com.google.gms:google-services:3.0.0'
```

app/build.gradle

```
Add Firebase plugin for Gradle
apply plugin: 'com.google.gms.google-services'
```

```
build.gradle will include these new dependencies:
compile 'com.google.firebase:firebase-database:10.0.1'
```

You will need to add other dependencies for other Firebase functionality.

For now go into the Firebase console and under Authentication disable Google sign-in.

In Database rules make sure anyone can read or write to your database.

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Layout

Look at the activity_recipe_main.xml and look what the template has provided. Notice how it includes the layout content_recipe_main and look at that xml file.

We just need a listview to show our list of recipes so replace the textview in content_recipe_main.xml with a listview. Use any layout you want.

<ListView

```
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:id="@+id/listView"
    android:layout_marginStart="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    android:layout_marginEnd="8dp"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="8dp"
    android:layout_marginBottom="8dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Model class

Go into your project's Java folder and click on the top folder and add a new Java class called RecipeItem for our model.

```
public class RecipeItem {
    private String name;
    private String url;

    public RecipeItem(){
        // Default constructor required for calls to DataSnapshot.getValue(RecipeItem.class)
    }

    public RecipeItem(String newName, String newURL){
        name = newName;
        url = newURL;
    }

    public String getName(){
        return name;
    }

    public String geturl(){
        return url;
    }

    //used when writing to the database
    @Exclude
    public Map<String, Object> toMap(){
        HashMap<String, Object> result = new HashMap<>();
```

```

    result.put("name", name);
    result.put("url", url);
    return result;
}

//the string representation of a recipe name
public String toString(){
    return this.name;
}
}

```

Map and HashMap both represent HashMap objects but represent different interfaces. Map is a more general interface, there are different ways to create a Map and HashMap is one of those. So this would let you change the implementation without changing the interface.

We'll do the rest in RecipeMainActivity.java

Read from Firebase

<https://firebase.google.com/docs/database/android/read-and-write>

To access your Firebase database add the following references:

```

// Firebase database instance
FirebaseDatabase database = FirebaseDatabase.getInstance();
//Firebase database reference
DatabaseReference ref = database.getReference();

```

We also need an arraylist and an arrayadapter.

```

//array list of recipes
List recipes = new ArrayList<>();
//array adapter
ArrayAdapter<RecipeItem> listAdapter;

```

```

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;

```

Update onCreate() to set an event listener for any data changes in the database.

```

ListView recipeList = (ListView) findViewById(R.id.listView);
listAdapter = new ArrayAdapter<RecipeItem>(this, android.R.layout.simple_list_item_1, recipes);
recipeList.setAdapter(listAdapter);

```

```

// Read from the database
ValueEventListener firebaseListener = new ValueEventListener() {

```

@Override

```
public void onDataChange(DataSnapshot dataSnapshot) {  
    // This method is called once with the initial value and again  
    // whenever data at this location is updated.  
  
    //empty the arraylist  
    recipes.clear();  
    for(DataSnapshot snapshot : dataSnapshot.getChildren()){  
        //create new RecipeItem object  
        RecipeItem newRecipe = snapshot.getValue(RecipeItem.class);  
        //add new recipe to our array  
        recipes.add(newRecipe);  
        Log.d("data", "Value is: " + newRecipe.getName() + newRecipe.geturl());  
    }  
    //update adapter  
    ListAdapter.notifyDataSetChanged();  
}
```

@Override

```
public void onCancelled(DatabaseError error) {  
    // Failed to read value  
    Log.w("oncreate", "Failed to read value.", error.toException());  
}  
};
```

```
//add listener to our database reference  
ref.addValueEventListener(firebaseListener);
```

You should now be able to run your app and see all your Firebase data. If you make any changes through the console your app should automatically update.

Delete data

To delete items through your app we'll present a context menu on a long press.

```
@Override public void onCreateContextMenu(ContextMenu menu, View view,  
ContextMenu.ContextMenuInfo menuInfo){  
    super.onCreateContextMenu(menu, view, menuInfo);  
    //cast ContextMenu.ContextMenuInfo to AdapterView.AdapterContextMenuInfo since we're using an  
    adapter  
    AdapterView.AdapterContextMenuInfo adapterContextMenuInfo =  
    (AdapterView.AdapterContextMenuInfo) menuInfo;  
    //get recipe name that was pressed  
    String recipename = ((TextView) adapterContextMenuInfo.targetView).getText().toString();  
    //set the menu title  
    menu.setHeaderTitle("Delete " + recipename);  
    //add the choices to the menu  
    menu.add(1, 1, 1, "Yes");  
    menu.add(2, 2, 2, "No");  
}
```

```

@Override public boolean onOptionsItemSelected(MenuItem item){
    //get the id of the item
    int itemId = item.getItemId();
    if (itemId == 1) { //if yes menu item was pressed
        //get the position of the menu item
        AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
item.getMenuInfo();
        //get recipe name that was pressed
        String recipename = ((TextView) info.targetView).getText().toString();
        //remove the recipe
        recipes.remove(info.position);
        //refresh the list view
        listAdapter.notifyDataSetChanged();
        //delete from Firebase
        ref.child(recipename).removeValue();
    }
    return true;
}

```

Register the context menu in onCreate()
registerForContextMenu(recipeList);

Add data

Change the floating action button from email to add in activity_recipe_main.xml

```
app:srcCompat="@android:drawable/ic_input_add"
```

Android has a ton of built-in drawables that we can use in our applications.

<http://androiddrawables.com/> listed by version.

Create new layout resource for the dialog.

File name: dialog.xml

Root element: Linear Layout

Source set: main

Directory name: layout

Add string resources.

```

<string name="add_recipe_name">Recipe name</string>
<string name="add_recipe_url">http://</string>
<string name="adddialog_title">Add Recipe</string>

```

Add two EditTexts and a Button to dialog.xml.

```

<EditText
    android:id="@+id/editTextName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"

```

```

    android:layout_marginTop="10dp"
    android:ems="20"
    android:hint="@string/add_recipe_name">
    <requestFocus />
</EditText>

<EditText
    android:id="@+id/editTextURL"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginTop="10dp"
    android:ems="20"
    android:text="@string/add_recipe_url">
</EditText>

<Button
    android:id="@+id/addButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginBottom="10dp"
    android:text="@string/adddialog_title" />

```

In RecipeMainActivity we already have a listener all set up for this button so we'll use that. In onCreate() we'll replace the Snackbar with the Dialog and the logic to add a recipe.

```

fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        final Dialog dialog = new Dialog(RecipeMainActivity.this);
        dialog setContentView(R.layout.dialog);
        dialog.setTitle("Add Recipe");
        dialog.setCancelable(true);
        final EditText nameEditText = (EditText) dialog.findViewById(R.id.editTextName);
        final EditText urlEditText = (EditText) dialog.findViewById(R.id.editTextURL);
        Button button = (Button) dialog.findViewById(R.id.addButton);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String recipeName = nameEditText.getText().toString();
                String recipeURL = urlEditText.getText().toString();
                // create new recipe item
                RecipeItem newRecipe = new RecipeItem(recipeName, recipeURL);
                // add to the array list
                recipes.add(newRecipe);
                //refresh the list view
                listAdapter.notifyDataSetChanged();
            }
        });
    }
});

```

```

        //add to Firebase
        ref.child(recipeName).setValue(newRecipe);
        dialog.dismiss();
    }
    });
    dialog.show();
}
});

```

When we create the Dialog it's in an anonymous function so when implementing the listener, outer class RecipeMainActivity has to be specified to refer to the Activity instance and the keyword this in Java applies to the most immediate class being declared.

You should now be able to add recipes and see them in your listview as well as in Firebase through the console.

Load web page

When the user taps on a recipe we want to load the recipe url in an app that can load web pages like a browser. We're going to use an implicit intent so the user will be prompted to chose an app to load the web page if the device has more than one capable of doing so.

Add to onCreate()

```

//create listener
AdapterView.OnItemClickListener itemClickListener = new AdapterView.OnItemClickListener(){
    public void onItemClick(AdapterView<?> listView, View view, int position, long id){
        //get tapped recipe
        RecipeItem recipeTapped = (RecipeItem) recipes.get(position);
        //get the recipe url
        String recipeURL = recipeTapped.geturl();
        //create new intent
        Intent intent = new Intent(Intent.ACTION_VIEW);
        //add url to intent
        intent.setData(Uri.parse(recipeURL));
        //start intent
        startActivity(intent);
    }
};
recipeList.setOnItemClickListener(itemClickListener);

```

Now when you tap on a recipe its web page should open in a browser.