Mobile Application Lab 9

Instructors: Legand Burge, Asad Ullah Naweed
Date: 09/18/2018
Topics: Firebase Realtime Database

Be sure to click on the links in this doc to view more information about certain topics. These may or may not help you in your lab, but can always provide some extra context and information.

Initial Setup

You'll need to use your solution to Lab 5 as a starting point. We'll be using the Firebase Realtime Database as a source of questions for your quiz app instead of hardcoding these questions in a resource file.

Create a new project, making sure that your application's package name is `com.techexchange.mobileapps.lab9`. Copy-paste the relevant pieces of code from Lab 5 into this project.

Adding Firebase

Core Firebase Setup

There are a few things you need to add to your project's gradle build files in order to be able to use Firebase in your Android app. Look at the "Add the SDK" section on this web page, and follow the instructions there.

Then, download the google-services.json file, and add it to your project's `app` directory.

Firebase Database Dependency

Now, add the following dependency to your app's build.gradle file:

```
dependencies {
  // ...
  implementation 'com.google.firebase:firebase-core:16.0.1'
  implementation 'com.google.firebase:firebase-database:16.0.1'
}
```

Before proceeding, build and run your app, and make sure that it compiles.

Obtaining data from the Firebase Database

We will now modify the code to obtain questions from Firebase, and update the list of questions available in real time as the database is modified.

The way to do this is to attach a listener for data to a particular node in the Firebase Database. The listener callback methods are invoked whenever new data is available from the database. This means that:

- The callback is invoked at least once to get the initial data state.
- The callback is invoked every single time something in the database changes.

This is a pretty powerful and simple technique for apps to listen for changes to a shared database state, and then adjust accordingly. For this lab, you will all be communicating with a single Firebase Database that has been set up for you already. The google-service.json file contains the information your app needs to communicate with that particular database, so all of that is abstracted away for you.

Basic boilerplate code

Make the following modifications to the `onCreateView` method inside `QuizFragment`:

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
  FirebaseApp app = FirebaseApp.initializeApp(getActivity());
  FirebaseDatabase database = FirebaseDatabase.getInstance(app);
  database.goOnline();

  // Existing code...
)
```

This is broilerplate code for initializing a FirebaseDatabase reference. Once you have a reference to the database, you can get a `DatabaseReference` for a specific node in the database and attach a listener to it.

You also need to add the following permission in your `AndroidManifest.xml` file:

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.techexchange.mobileapps.lab9">
    <uses-permission android:name="android.permission.INTERNET" />
    <!-- Existing code for application... -->
</manifest>
```

Before proceeding, build and run your app, and make sure that it still works.

Attaching the data listener

Now, do the following:

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
  FirebaseApp app = FirebaseApp.initializeApp(getActivity());
  FirebaseDatabase database = FirebaseDatabase.getInstance(app);
  database.goOnline();

  DatabaseReference ref = database.getReference();
  ref = ref.child("lab9").child("questions");

  ref.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
      onQuestionListChanged(dataSnapshot);
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
      Log.d(TAG, "The database transaction was cancelled",
          databaseError.toException());
    }
  });

  // Existing code here...

  ~~questionList = initQuestionList();~~
}

private void onQuestionListChanged(DataSnapshot dataSnapshot) {
  // To be implemented...
}
```

```
~~private void initQuestionList() {~~
~~    // All existing code...~~
~~}~~
```

What's happening here is pretty simple: You're attaching a listener to the node at path `/lab9/questions/`. The callback will get invoked once, and then again every single time the data gets changed, so your code can cater for that.

Build and run your app. It should crash. Identify the bug and fix it. Logcat messages are your friends.

Implementing onQuestionListChanged

Take a look at the documentation for `DataSnapshot`. A single instance of `DataSnapshot` is immutable, and represents the data present at the node when the callback was invoked.

Now take a look at the Firebase Database structure on the screen, and implement the method. Your method should:

1. Create a new list of questions with data from the data snapshot.
2. Update the UI.