Kostenloses eBook

LERNEN

firebase-database

Free unaffiliated eBook created from
Stack Overflow contributors.

#firebase-database

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: firebase-database

It is an unofficial and free firebase-database ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official firebase-database.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit der Firebase-Datenbank

Bemerkungen

Dieser Abschnitt bietet einen Überblick über die Firebase-Datenbank und warum ein Entwickler sie verwenden möchte.

Es sollte auch alle großen Themen in der Firebase-Datenbank erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation für die Firebase-Datenbank neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Versionen

Plattform-SDK	Ausführung	Veröffentlichungsdatum
Firebase JavaScript SDK	3.7.0	2017-03-01
Firebase C ++ SDK	3.0.0	2107-02-27
Firebase Unity SDK	3.0.0	2107-02-27
Firebase iOS SDK	3.14.0	2017-02-23
Firebase Android SDK	10.2	2017-02-15
Firebase Admin Node.js SDK	4.1.1	2017-02-14
Firebase Admin Java SDK	4.1.2	2017-02-14

Examples

Fügen Sie Firebase zu Ihrem Android-Projekt hinzu

Hier die Schritte, die zum Erstellen eines Firebase-Projekts und zum Verbinden mit einer Android-App erforderlich sind.

Fügen Sie Firebase Ihrer App hinzu

1. Erstellen Sie ein Firebase-Projekt in der Firebase-Konsole und klicken **Sie auf Neues Projekt erstellen** .
2. Klicken **Sie auf Firebase zu Ihrer Android-App hinzufügen** und folgen Sie den

Installationsschritten.

3. Wenn Sie dazu aufgefordert werden, geben Sie **den Paketnamen** Ihrer **App ein** .
   Es ist wichtig, den Paketnamen einzugeben, den Ihre App verwendet. Dies kann nur eingestellt werden, wenn Sie Ihrem Firebase-Projekt eine App hinzufügen.

4. Am Ende laden Sie eine `google-services.json` Datei herunter. Sie können diese Datei jederzeit erneut herunterladen. (Diese Datei befindet sich unter der Projekteinstellung in der Firebase-Konsole).

5. Wechseln Sie in die Android-Projektansicht, und fügen Sie die Datei google-service.json im App-Ordner ein

Im nächsten Schritt fügen Sie das SDK hinzu, um die Firebase-Bibliotheken in das Projekt zu integrieren.

Fügen Sie das SDK hinzu

Um die Firebase-Bibliotheken in eines Ihrer eigenen Projekte zu integrieren, müssen Sie einige grundlegende Aufgaben zur Vorbereitung Ihres Android Studio-Projekts ausführen. Möglicherweise haben Sie dies bereits beim Hinzufügen von Firebase zu Ihrer App getan.

1. Fügen Sie der `build.gradle` Datei `build.gradle` Regeln `build.gradle` , um das **google-services-Plugin aufzunehmen** :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.1.1'
    }
}
```

`app/build.gradle` dann in Ihrer Modul-Gradle-Datei (normalerweise `app/build.gradle`) die `app/build.gradle` Plugin-Zeile am Ende der Datei hinzu, um das Gradle-Plugin zu aktivieren:

```
apply plugin: 'com.android.application'

android {
  // ...
}

dependencies {
  // ...
  compile 'com.google.firebase:firebase-core:9.4.0'//THIS IS FOR ANALYTICS
  compile "com.google.firebase:firebase-database:11.0.2"
}

// BELOW STATEMENT MUST BE WRITTEN IN BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Anmerkungen:

- Daten können ohne Authentifizierung nicht gelesen / geschrieben werden. Wenn Sie es ohne Authentifizierung wünschen. Ändern Sie die Regeln in der Database Firebase Console.

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

- Fügen Sie die Internet-Berechtigung in Manifest hinzu
- Aktualisieren Sie die Google Play-Dienste und das Google Repository

Einfachen Wert in die Datenbank schreiben

Schließen Sie zuerst die Installation und das Setup ab , um Ihre App mit Firebase zu verbinden. Dann können Sie von überall in Ihrer Klasse schreiben:

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Es wird `Hello, Wold!` schreiben `Hello, Wold!` in den `message` , wie unten zu sehen:

```
"your-project-parent" : {
    "message" : "Hello, World!"
}
```

Erläuterung

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

Mit dem obigen Code wird die `FirebaseDatabase` Instanz dem `database` zur weiteren Verwendung zugewiesen.

```
DatabaseReference myRef = database.getReference("message");
```

Der `myRef` Code verweist auf das `myRef` Objekt in dem `myRef "message"` des übergeordneten Projekts (in diesem Beispiel ist es `"your-project-parent"`). Also ist es `"your-project-parent/message"`

```
myRef.setValue("Hello, World!");
```

Über dem Code wird `"Hello, World!"` in den von `myRef` referenzierten `myRef`

Ordnen Sie das benutzerdefinierte Modell automatisch der Datenstruktur zu

Nachdem Sie ein paar Daten für die Datenbank festgelegt haben, erhalten Sie eine Struktur, die aus mehreren Knoten wie dieser besteht.

```
"user" : {
    "-KdbKcU2ptfYF2xKb5aO" : {
      "firstName" : "Arthur",
      "lastName" : "Schopenhauer",
      "userName" : "AphorismMan",
      "phone" : "+9022-02-1778",
      "gender": "M",
      "age" : 25
    },
    "-KdbQFjs9BDviuqQVHjY" : {
      "firstName" : "Werner",
      "lastName" : "Heisenberg",
      "userName" : "whereAmI",
      "phone" : "+9005-12-1901",
      "gender": "M",
      "age" : 75
    }
  }
```

Sie können Datenstrukturen kategorisieren.

Klasse erstellen

Erstellen Sie eine Modellklasse, um die Datenbank festzulegen.

```
@IgnoreExtraProperties
public class User {
    public String firstName;
    public String lastName;
    public String userName;
    public String phone;
    public String gender;
    public int age;

    public User() {
    }

    public User(String firstName, String lastName, String userName, String phone, String
gender, int age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.userName = userName;
        this.phone = phone;
        this.gender = gender;
        this.age = age;
    }
}
```

Einige Dinge, die Sie beim Erstellen einer Modellklasse beachten sollten, die Sie Ihren Daten zuordnen möchten:

1. Sie müssen einen leeren Konstruktor haben
2. Der Umfang der Variablen / Felder muss öffentlich sein, damit der von der Firebase zurückgegebene DataSnapshot auf diese Felder zugreifen kann. Wenn Sie dies nicht tun, wenn Sie Daten abrufen möchten, kann DataSnapshot im Callback nicht auf Ihr Modell zugreifen. Dies führt zu einer Ausnahme.

3. Die Namen der Variablen / Felder sollten mit denen in Ihrer Datenstruktur übereinstimmen.

Senden an Firebase

Erstellen Sie ein Benutzerobjekt

```
User user = new User ( "Arthur","Schopenhauer","AphorismMan","+9022-02-1778","M",25)
```

und Referenz

```
DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference();
```

Jetzt haben Sie die Referenz Ihrer Datenbank. Erstellen Sie einen `user` mit `databaseReference.child("user")` . Wenn Sie `.push()` Ihre Modelle unter zufällig erstellten eindeutigen IDs wie oben `"-KdbKcU2ptfYF2xKb5aO", "-KdbQFjs9BDviuqQVHjY"` .

```
databaseReference.child("user").push().setValue(user, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError databaseError, DatabaseReference
databaseReference) {
                Toast.makeText(getActivity(), "User added.", Toast.LENGTH_SHORT).show();

            }
        });
```

Wenn Sie Ihre Daten unter Ihrem spezifischen Schlüssel `.child("yourSpecificKey")` , tun Sie dies mit `.child("yourSpecificKey")` anstelle von `.push()` .

```
databaseReference.child("user").child("yourSpecificKey").setValue(user,...
```

Erste Schritte mit der Firebase-Datenbank online lesen: https://riptutorial.com/de/firebase-database/topic/3044/erste-schritte-mit-der-firebase-datenbank

Kapitel 2: Daten lesen

Examples

Verstehen, welche Daten von getReference () referenziert werden

In diesem Beispiel verwenden wir diese Datenbank:

```
"your-project-name" : {
    "users" : {
        "randomUserId1" : {
            "display-name" : "John Doe",
            "gender" : "male"
        }
        "randomUserId2" : {
            "display-name" : "Jane Dae",
            "gender" : "female"
        }
    },
    "books" {
        "bookId1" : {
            "title" : "Adventure of Someone"
        },
        "bookId1" : {
            "title" : "Harry Potter"
        },
        "bookId1" : {
            "title" : "Game of Throne"
        }
    }
}
```

Wenn Sie die obige Datenbank verwenden, dann:

- `FirebaseDatabase.getInstance().getReference()`

  zeigt auf das übergeordnete Projekt Ihres Projekts, `"your-project-name"` . Der `dataSnapshot` Ihnen erfasste `dataSnapshot` enthält also alle darin enthaltenen Daten, einschließlich der Daten für `"users"` und `"books"` .

- `FirebaseDatabase.getInstance().getReference("users")` und `FirebaseDatabase.getInstance().getReference().child("users")`

  wird das gleiche Ergebnis haben und auf `"your-project-name/users"`

- `FirebaseDatabase.getInstance().getReference("users/randomUserId1")` und `FirebaseDatabase.getInstance().getReference().child("users/randomUserId1")` und `FirebaseDatabase.getInstance().getReference().child("users").child("randomUserId1")`

  hat das gleiche Ergebnis und zeigt auf `"your-project-name/users/randomUserId1"`

  Hinweis: Dieses Beispiel ist erforderlich, um vollständig zu verstehen, welche Daten sich im dataSnapshot-Objekt befinden

Verstehen, welche Daten sich im dataSnapshot-Objekt befinden

> Hinweis: Sie müssen zuerst wissen, welche Daten von getReference () referenziert werden, bevor Sie dieses Beispiel vollständig verstehen können.

Es gibt drei gängige Methoden, um Ihre Daten aus der Firebase-Echtzeitdatenbank abzurufen:

- `addValueEventListener()`
- `addListenerForSingleValueEvent()`
- `addChildEventListener()`

Wenn wir darüber sprechen, welche Daten sich im `dataSnapshot` Objekt befinden, sind `addValueEventListener()` und `addListenerForSingleValueEvent()` grundsätzlich gleich. Der einzige Unterschied besteht darin, dass `addValueEventListener()` die Änderungen in den referenzierten Daten `addListenerForSingleValueEvent()` jedoch nicht.

Angenommen, wir haben diese Datenbank:

```
"your-project-name" : {
    "users" : {
        "randomUserId1" : {
            "display-name" : "John Doe",
            "gender" : "male"
        }
        "randomUserId2" : {
            "display-name" : "Jane Dae",
            "gender" : "female"
        }
    },
    "books" {
        "bookId1" : {
            "title" : "Adventure of Someone"
        },
        "bookId1" : {
            "title" : "Harry Potter"
        },
        "bookId1" : {
            "title" : "Game of Throne"
        }
    }
}
```

Von addValueEventListener und addListenerForSingleValueEvent erzeugter DataSnapshot

`dataSnapshot` der von `addValueEventListener()` und `addListenerForSingleValueEvent()` wird, enthält den Wert der exakten Daten, auf die er verweist. Wie , wenn `ref` Punkt ist `"your-project-name"` dann `dataSnapshot` sollte sein:

```
... onDataChange(DataSnapshot dataSnapshot) {
    dataSnapshot.getKey(); // will have value of String: "your-project-name"
    for (DataSnapshot snapshot : dataSnapshot) {
        snapshot.getKey(); // will have value of String: "users", then "books"
        for (DataSnapshot deeperSnapshot : dataSnapshot) {
```

```
            snapshot.getKey();
            // if snapshot.getKey() is "users", this will have value of String:
"randomUserId1", then "randomUserId2"
            // If snapshot.getKey() is "books", this will have value of String: "bookId1",
then "bookId2"
        }
    }
}
```

Von addChildEventListener erzeugter DataSnapshot

`dataSnapshot` das von `addChildEventListener()` wird, enthält einen oder `addChildEventListener()` Datenwerte, die eine Ebene tiefer in den Daten liegen, auf die sie verweist. Wie in diesen Fällen:

Wenn `ref` auf `"your-project-name"` `dataSnapshot` sollte `dataSnapshot` lauten:

```
... onChildAdded(DataSnapshot dataSnapshot, String s) {
    dataSnapshot.getKey(); // will have value of String: "users", then "books"
    for (DataSnapshot snapshot : dataSnapshot) {
        snapshot.getKey();
        // if dataSnapshot.getKey() is "users", this will have value of String:
"randomUserId1", then "randomUserId2"
        // If dataSnapshot.getKey() is "books", this will have value of String: "bookId1",
then "bookId2"
        for (DataSnapshot deeperSnapshot : dataSnapshot) {
            snapshot.getKey();
            // if snapshot.getKey() is "randomUserId1" or "randomUserId1", this will have
value of String: "display-name", then "gender"
            // But the value will be different based on key
            // If snapshot.getKey() is "books", this will have value of String: "title", but
the value will be different based on key
        }
    }
}
// dataSnapshot inside onChildChanged, onChildMoved, and onChildRemoved will have the same
data as onChildAdded
```

Ich weiß, dass wir `.getValue()` anstelle von `getKey()` verwenden `.getValue()` . Aber hier verwenden wir `getKey` da es immer einen String enthält und nicht in ein benutzerdefiniertes Objekt oder eine Map oder ein anderes konvertiert werden muss. Wenn Sie wissen, in welchen Schlüssel `dataSnapshot` zeigt, können Sie leicht wissen, welchen Wert es enthält, und es in Ihr eigenes benutzerdefiniertes Objekt (oder irgendetwas) analysieren.

Daten lesen online lesen: https://riptutorial.com/de/firebase-database/topic/9242/daten-lesen

Kapitel 3: Firebase-Abfrage

Einführung

Firebase Query kann verwendet werden, um eine Sammlung von Daten basierend auf einigen Attributen zu ordnen sowie auf die große Liste von Elementen (für ähnliche Chat-Daten) beschränkt zu halten, bis zu einer Anzahl, die zur Synchronisierung mit dem Client geeignet ist.

Genau wie bei einer Referenz können Sie Daten von einer Abfrage mithilfe der on () -Methode empfangen. Sie erhalten nur Ereignisse und DataSnapshots für die Teilmenge der Daten, die Ihrer Abfrage entsprechen.

Examples

Firebase-Abfragebeispiel

```
private void loadData(){
        DatabaseReference dbRef = FirebaseDatabase.getInstance().getReference();

        Query dataQuery = dbRef.child("chat").orderByChild("id").equalTo("user1");
        dataQuery.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    // dataSnapshot is the "issue" node with all children with id 0
                    for (DataSnapshot issue : dataSnapshot.getChildren()) {
                        // do something with the individual "issues"
                    }
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }
```

Firebase-Abfrage online lesen: https://riptutorial.com/de/firebase-database/topic/10100/firebase-abfrage

Kapitel 4: Firebase-Echtzeitdatenbank mit Android

Examples

Integrieren Sie die Echtzeitdatenbank von Firebase in eine Android-Anwendung

So implementieren Sie die Firebase-Echtzeitdatenbank in Android-Anwendungen.

Setup / Installation:

1. Installieren Sie zuerst das Firebase-SDK (Anleitung).
2. Registrieren Sie Ihr Projekt über die Firebase- Konsole
3. Fügen Sie nach dem erfolgreichen Abschluss der beiden obigen Schritte die folgende Abhängigkeit in Ihrem Gradel der Anwendungsebene hinzu.

```
compile 'com.google.firebase:firebase-database:9.2.1'
```

4. [Optional] Konfigurieren Sie Ihre Datenbanksicherheitsregeln (Referenz).

Implementierungsbeispiel:

1. Deklarieren und initialisieren Sie die Datenbankreferenz

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");
```

Sie können später verschiedene Referenzen erstellen, um auf verschiedene Knoten zuzugreifen

6. Schreibe neue Daten in die Datenbank

```
myRef.setValue("Writing Demo");
```

7. Daten aus der Datenbank lesen

```
myRef.addValueEventListener(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.
    String value = dataSnapshot.getValue(String.class);
    Log.d(TAG, "Value is: " + value);
}
```

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

Firebase-Echtzeitdatenbank mit Android online lesen: https://riptutorial.com/de/firebase-database/topic/6341/firebase-echtzeitdatenbank-mit-android

Kapitel 5: Firebase-Echtzeit-Datenbank-Transaktionen

Einführung

Transaktionen bieten einen Mechanismus zum Koordinieren zwischen mehreren Parteien, die möglicherweise gleichzeitig auf dieselben Daten zugreifen. Diese "Parteien" können verschiedene Instanzen desselben Codes sein, wie verschiedene Benutzer, die dieselbe Anwendung oder Knoten in einem Server-Cluster ausführen, Teile des gleichen Programms oder ereignisabhängige Programme wie eine Administrationsanwendung, eine "Endbenutzer" -Anwendung und / oder " Backend "Serverlogik.

Examples

Ein verteilter Zähler

Stellen Sie sich vor, viele Benutzer führen alle eine Webanwendung aus, die versucht, einen Zähler in der Datenbank zu erhöhen. Jeder Benutzer muss den aktuellen Zählerstand lesen, einen addieren und den aktualisierten Wert ausschreiben. Um sicherzustellen, dass niemand den Zähler liest, während ein anderer einen Zähler hinzufügt, verwenden wir eine Transaktion:

```
ref.transaction(function(value){
  if (value === null) {
    // the counter doesn't exist yet, start at one
    return 1;
  } else if (typeof value === 'number') {
    // increment - the normal case
    return value + 1;
  } else {
    // we can't increment non-numeric values
    console.log('The counter has a non-numeric value: ' + value)
    // letting the callback return undefined cancels the transaction
  }
});
```

Firebase-Echtzeit-Datenbank-Transaktionen online lesen: https://riptutorial.com/de/firebase-database/topic/9612/firebase-echtzeit-datenbank-transaktionen

Kapitel 6: FirebaseRealtime-Datenbank mit Android

Examples

Fügen Sie die Echtzeitdatenbank in Android hinzu

1. Schließen Sie die Installation und das Setup ab , um Ihre App mit Firebase zu verbinden. Dadurch wird das Projekt in Firebase erstellt.

2. Fügen Sie der `build.gradle` Datei auf Modulebene die Abhängigkeit für die Firebase-Echtzeitdatenbank `build.gradle` :

```
compile 'com.google.firebase:firebase-database:9.2.1'
```

3. Konfigurieren Sie die Firebase-Datenbankregeln

Jetzt können Sie mit der Echtzeitdatenbank in Android arbeiten.

Beispielsweise schreiben Sie eine `Hello World` Nachricht unter dem `message` in die Datenbank.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Daten mit setValue speichern

Die `setValue()` -Methode überschreibt Daten an der angegebenen Position, einschließlich aller `setValue()` Knoten.

Sie können diese Methode verwenden, um:

1. Übergeben Sie Typen, die den verfügbaren JSON-Typen entsprechen, wie folgt:

- String
- Lange
- Doppelt
- Boolean
- Map <String, Object>
- Liste

2. Übergeben Sie ein benutzerdefiniertes Java-Objekt, wenn die Klasse, die es definiert, über einen Standardkonstruktor verfügt, der keine Argumente enthält und öffentliche Eigenschaften für die zuzuweisenden Eigenschaften vorliegen.

Dies ist ein Beispiel mit einem CustomObject.
Definieren Sie zuerst das Objekt.

```
@IgnoreExtraProperties
public class User {

    public String username;
    public String email;

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

    public User(String username, String email) {
        this.username = username;
        this.email = email;
    }
```

Rufen Sie dann die Datenbankreferenz ab und legen Sie den Wert fest:

```
   User user = new User(name, email);
   DatabaseReference mDatabase mDatabase = FirebaseDatabase.getInstance().getReference();
   mDatabase.child("users").child(userId).setValue(user);
```

Beispiel für das Einfügen von Daten oder das Abrufen von Daten aus Firebase

Bevor Sie verstehen, müssen Sie einige Einstellungen für die Projektintegration in Firebase beachten.

1. Erstellen Sie Ihr Projekt in Firebase Console und laden Sie die Datei google-service.json von der Konsole herunter und fügen Sie sie in das App-Level-Modul Ihres Projekts ein. Folgen Sie dem Link zum Erstellen von Projekten in der Konsole

2. Danach müssen wir einige Abhängigkeiten in unserem Projekt hinzufügen.

- Fügen Sie zunächst den Klassenpfad in unserem Projektlevel hinzu.

  ```
  classpath 'com.google.gms:google-services:3.0.0'
  ```

- Und danach Plugin in App Level Gradel anwenden, schreiben Sie es in den Abhängigkeitsbereich,

  ```
  apply plugin: 'com.google.gms.google-services
  ```

- Es gibt mehr Abhängigkeiten, die das Hinzufügen eines Grads der App-Ebene im Abhängigkeitsbereich erfordern

  ```
  compile 'com.google.firebase:firebase-core:9.0.2'

  compile 'com.google.firebase:firebase-database:9.0.2'
  ```

- Starten Sie nun das Einfügen von Daten in die Firebase-Datenbank. Zuerst müssen Sie eine Instanz von erstellen

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

Nach der Erstellung des FirebaseDatabase-Objekts erstellen wir unsere DatabaseReference zum Einfügen von Daten in die Datenbank.

```
DatabaseReference databaseReference = database.getReference().child("student");
```

Hier ist `student` der Tabellenname, wenn eine Tabelle in der Datenbank vorhanden ist, und fügt die Daten in die Tabelle ein. Ansonsten erstellen Sie eine neue mit dem Schülernamen. Danach können Sie Daten mit `databaseReference.setValue();` einfügen `databaseReference.setValue();` funktionieren wie folgt,

```
HashMap<String,String> student=new HashMap<>();

student.put("RollNo","1");

student.put("Name","Jayesh");

databaseReference.setValue(student);
```

Hier füge ich Daten als hasmap ein. Sie können aber auch als Modellklasse festlegen,

- Starten Sie den Abruf von Daten aus der Firebase. Wir verwenden hier addListenerForSingleValueEvent für den Lesewert aus der Datenbank.

```
  senderRefrence.addListenerForSingleValueEvent(new ValueEventListener() {
          @Override
          public void onDataChange(DataSnapshot dataSnapshot) {
              if(dataSnapshot!=null && dataSnapshot.exists()){
                  HashMap<String,String>
studentData=dataSnapshot.getValue(HashMap.class);
                  Log.d("Student Roll Num "," : "+studentData.get("RollNo"));
                  Log.d("Student Name "," : "+studentData.get("Name"));
              }
          }

          @Override
          public void onCancelled(DatabaseError databaseError) {

          }
      });
```

Holen Sie sich Werte von Firebase

1. Klasse erstellen und Importe hinzufügen, um Informationen zu analysieren:

```
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.IgnoreExtraProperties;

//Declaration of firebase references
private DatabaseReference mDatabase;

//Declaration of firebase atributtes
public String uID;
public String username;
public String email;
```

```
@IgnoreExtraProperties
public class User {

    //Default constructor
    public User() {

        //Default constructor required for calls to DataSnapshot.getValue(User.class)
        mDatabase = FirebaseDatabase.getInstance().getReference();

        //...
    }

    //...
}
```

2. Fügen Sie `addListenerForSingleValueEvent()` zu unserer Datenbankreferenz hinzu:

```
//Add new imports
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.ValueEventListener;

//...

public void getUser(String uID){

    //The uID it's unique id generated by firebase database
    mDatabase.child("users").child(uID).addListenerForSingleValueEvent(
            new ValueEventListener () {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                // ...
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // Getting Post failed, log a message
            }
    });
}
```

3. Aufblasen unserer Klasse mit Informationen zu Firebase im `onDataChange()` Ereignis:

```
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {

            //Inflate class with dataSnapShot
            Users user = dataSnapshot.getValue(Users.class);

            //...
        }
```

4. Schließlich können wir wie üblich verschiedene Attribute von der Firebase-Klasse erhalten:

```
//User inflated
Users user = dataSnapshot.getValue(Users.class);
```

```
//Get information
this.uID = user.uID;
this.username = user.username;
this.email = user.email;
```

Best Practices

1. Die Firebase unterstützt 32 verschiedene untergeordnete Ebenen. Dann ist es einfach, falsche Referenzen zu schreiben.

```
//Declaration of firebase references
//...
final private DatabaseReference userRef =
mDatabase.child("users").child("premium").child("normal").getRef();

//...

public void getUser(String uID){

    //Call our reference
    userRef.child(uID).addListenerForSingleValueEvent(
        new ValueEventListener () {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                // ...
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // Getting Post failed, log a message
            }
    });
}
```

2. Das `onCancelled()` Ereignis wird aufgerufen, wenn der Benutzer über Datenbankregeln keinen Zugriff auf diese Referenz hat. Fügen Sie erforderlichenfalls den entsprechenden Code hinzu, um diese Ausnahme zu steuern.

Weitere Informationen finden Sie in der offiziellen Dokumentation

FirebaseRealtime-Datenbank mit Android online lesen: https://riptutorial.com/de/firebase-database/topic/6220/firebaserealtime-datenbank-mit-android

Kapitel 7: Hallo Welt!

Examples

Hallo Welt in Android

1. Schließen Sie den Installations- und Setup-Teil ab . Dadurch wird das Projekt in der Firebase-Konsole erstellt und das Basis-SDK in Ihrer Android-App installiert.
2. Fügen Sie der `build.gradle` Datei auf App-Ebene die Abhängigkeit für die Firebase-Echtzeitdatenbank `build.gradle` :

```
compile 'com.google.firebase:firebase-database:9.4.0'
```

3. Schreiben Sie nun eine `Hello World` Nachricht unter dem `message` in die Datenbank.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

4. Prüfen Sie, ob die Nachricht in der `Realtime Database` .

Hallo Welt in IOS

1. Nachdem Sie Firebase für Ihr IOS-Projekt eingerichtet haben und die Dokumentation zum Einrichten von Firebase für IOS in einer anderen Dokumentation zu Firebase beschrieben haben, können Sie mit Firebase loslegen.

2. Wenn Sie den Datenbank-Pod noch nicht zu Ihrer Pod-Datei hinzugefügt haben, fügen Sie `pod Firebase/Database` , um die Datenbankfunktionalität für Firebase zu erhalten

3. Drücken Sie 'Hello World' in die Datenbank. Zuerst müssen Sie jedoch die Regeln für Ihre Datenbank im Dashboard Ihrer App in Firebase bearbeiten und diese ändern

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

   HINWEIS: Stellen Sie sicher, dass Sie diese Einstellung ändern, bevor Sie in die Produktion gehen, da dies bedeutet, dass jeder in Ihre Datenbank lesen und schreiben kann, was einen erheblichen Sicherheitsmangel darstellt

4. Jetzt müssen Sie nur Firebase in die Datei importieren, in der Sie Firbase verwenden, indem

Sie mit `import Firebase` die folgende Zeile schreiben. In der Datenbank sollte ein "Test" - Knoten angezeigt werden. Erweitern Sie ihn und er sollte ein untergeordnetes Element "Hello World" haben !!!!!!

```
FIRDatabase.database().reference().child("Test").setValue("Hello World")
```

Herzlichen Glückwunsch zu Ihrem ersten Datenbank-Push

Hallo Welt! online lesen: https://riptutorial.com/de/firebase-database/topic/8885/hallo-welt-

Kapitel 8: Regeln für die Echtzeitdatenbank von Firebase

Bemerkungen

Die Echtzeitdatenbankregeln von Firebase bestimmen, wer Lese- und Schreibzugriff auf Ihre Datenbank hat, wie Ihre Daten strukturiert sind und welche Indizes vorhanden sind. Diese Regeln leben auf den Firebase-Servern und werden zu jeder Zeit automatisch durchgesetzt. Jede Lese- und Schreibanfrage wird nur ausgeführt, wenn Ihre Regeln dies zulassen. Standardmäßig sind Ihre Regeln so festgelegt, dass nur authentifizierte Benutzer vollen Lese- und Schreibzugriff auf Ihre Datenbank haben. Dies dient zum Schutz Ihrer Datenbank vor Missbrauch, bis Sie Zeit haben, Ihre Regeln anzupassen oder die Authentifizierung einzurichten.

Firebase-Datenbankregeln haben eine JavaScript-artige Syntax und es gibt vier Arten:

Rule Types	
.read	Describes if and when data is allowed to be read by users.
.write	Describes if and when data is allowed to be written.
.validate	Defines what a correctly formatted value will look like, whether i and the data type.
.indexOn	Specifies a child to index to support ordering and querying.

Examples

Genehmigung

Die Identifizierung Ihres Benutzers ist nur ein Teil der Sicherheit. Sobald Sie wissen, wer sie sind, benötigen Sie eine Möglichkeit, den Zugriff auf die Daten in Ihrer Datenbank zu kontrollieren. Mit den Firebase-Datenbankregeln können Sie den Zugriff für jeden Benutzer steuern. Hier sind zum Beispiel Sicherheitsregeln, die es jedem erlauben, den Pfad `/foo/` zu lesen, aber niemand, der darauf schreiben kann:

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

```
}
```

`.read` und `.write` Regeln kaskadieren, so dass dieser Regelsatz Lesezugriff auf alle Daten unter Pfad / foo / sowie auf tiefere Pfade wie `/foo/bar/baz` gewährt. Beachten Sie, dass `.read` und `.write` Regeln, die den Zugriff zulassen, andere Regeln in der Datenbank überschreiben, die keinen Zugriff zulassen. Mit anderen Worten, alle anwendbaren Regeln `.read` und `.write` werden zusammen `.write` verknüpft. Lesezugriff auf `/foo/bar/baz` würde also in diesem Beispiel auch dann gewährt, wenn eine Regel im Pfad `/foo/bar/baz` als falsch ausgewertet wird.

Die Firebase-Datenbankregeln enthalten integrierte Variablen und Funktionen, mit denen Sie auf andere Pfade, serverseitige Zeitstempel, Authentifizierungsinformationen usw. zugreifen können. Hier ein Beispiel für eine Regel, die authentifizierten Benutzern Schreibzugriff auf `/users/<uid>/` . Dabei handelt es sich um die ID des Benutzers, die über die Firebase-Authentifizierung abgerufen wird.

```
{
  "rules": {
    "users": {
      "$uid": {
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

Datenvalidierung

Die Firebase-Echtzeitdatenbank ist ein Schema. Dies macht es einfach, die Dinge während der Entwicklung zu ändern. Sobald Ihre App jedoch bereit ist zu verteilen, ist es wichtig, dass die Daten konsistent bleiben. Die `.validate` enthält eine `.validate` Regel, mit der Sie die Validierungslogik mit den gleichen Ausdrücken wie `.read` und `.write` Regeln `.write` . Der einzige Unterschied besteht darin, dass alle relevanten Validierungsregeln auf true ausgewertet werden müssen, damit das Schreiben zulässig ist (mit anderen Worten, alle anwendbaren `.validate` Regeln werden UND-verknüpft, um ein Datenbankschreiben zu ermöglichen.

Diese Regel erzwingt, dass Daten, die in `/foo/` geschrieben werden, aus weniger als 100 Zeichen bestehen müssen:

```
{
  "rules": {
    "foo": {
      ".validate": "newData.isString() && newData.val().length < 100"
    }
  }
}
```

Überprüfungsregeln haben Zugriff auf dieselben integrierten Funktionen und Variablen wie die Regeln `.read` und `.write` . Sie können diese verwenden, um Validierungsregeln zu erstellen, die Daten an anderer Stelle in Ihrer Datenbank, die Identität Ihres Benutzers, die Serverzeit und vieles mehr kennen.

Datenbankindizes definieren

Die Firebase-Echtzeitdatenbank ermöglicht das Sortieren und Abfragen von Daten. Bei kleinen Datengrößen unterstützt die Datenbank Ad-hoc-Abfragen, sodass während der Entwicklung im Allgemeinen keine Indizes erforderlich sind. Bevor Sie jedoch Ihre App starten, müssen Sie unbedingt Indizes für alle Abfragen angeben, um sicherzustellen, dass sie mit dem Wachstum Ihrer App weiter funktionieren.

Indizes werden mit der Regel `.indexOn` angegeben. Hier ist ein Beispiel für eine Indexdeklaration, die die Höhen- und Längenfelder für eine Liste von Dinosauriern indizieren würde:

```
{
  "rules": {
    "dinosaurs": {
      ".indexOn": ["height", "length"]
    }
  }
}
```

Regeln für die Echtzeitdatenbank von Firebase online lesen: https://riptutorial.com/de/firebase-database/topic/3348/regeln-fur-die-echtzeitdatenbank-von-firebase

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit der Firebase-Datenbank	Abdul Wasae, Adarsh Madrecha, AtaerCaner , Community, ErstwhileIII, Gabriele Mariotti, koceeng, Nepster, Shiven, TwiterZX, Veeresh Charantimath
2	Daten lesen	koceeng
3	Firebase-Abfrage	Dhaval Solanki
4	Firebase-Echtzeitdatenbank mit Android	Dhaval Solanki, Krishna Kumar, ThunderStruct
5	Firebase-Echtzeit-Datenbank-Transaktionen	Mike
6	FirebaseRealtime-Datenbank mit Android	Dhaval Solanki, Gabriele Mariotti, Merlí Escarpenter Pérez, ThunderStruct
7	Hallo Welt!	noob, RyanM, ThunderStruct
8	Regeln für die Echtzeitdatenbank von Firebase	Adarsh Madrecha, mckoss