



eBook Gratuit

APPRENEZ firebase-database

eBook gratuit non affilié créé à partir des
contributeurs de Stack Overflow.

#firebase-
database

Table des matières

À propos	1
Chapitre 1: Démarrer avec firebase-database	2
Remarques	2
Versions	2
Exemples	2
Ajouter Firebase à votre projet Android	2
Ajouter Firebase à votre application	2
Ajouter le SDK	3
Ecrire une valeur simple dans la base de données	4
Mapper automatiquement le modèle personnalisé à la structure de données	4
Chapitre 2: Base de données FirebaseRealtime avec Android	7
Exemples	7
Ajouter la base de données en temps réel dans Android	7
Utiliser setValue pour enregistrer des données	7
Exemple d'insertion de données ou de récupération de données à partir de Firebase	8
Obtenir la valeur / s de la base de feu	9
Chapitre 3: Bonjour le monde!	12
Exemples	12
Hello World en Android	12
Bonjour tout le monde dans IOS	12
Chapitre 4: Firebase Real-Time Database avec Android	14
Exemples	14
Intégrer la base de données Firebase Real-Time à une application Android	14
Chapitre 5: Lecture des données	16
Exemples	16
Comprendre quelles données référencées par getReference ()	16
Comprendre quelles données se trouvent dans l'objet dataSnapshot	17
Chapitre 6: Règles de base de données en temps réel Firebase	19
Remarques	19
Exemples	19

Autorisation.....	19
La validation des données.....	20
Définition des index de base de données.....	21
Chapitre 7: Requête Firebase.....	22
Introduction.....	22
Exemples.....	22
Exemple de requête Firebase.....	22
Chapitre 8: Transactions de base de données en temps réel Firebase.....	23
Introduction.....	23
Exemples.....	23
Un compteur distribué.....	23
Crédits.....	24

À propos

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Chapitre 1: Démarrer avec firebase-database

Remarques

Cette section fournit une vue d'ensemble de la base de données firebase et des raisons pour lesquelles un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans la base de données firebase, et établir un lien avec les sujets connexes. La documentation de la base de données firebase étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Platform SDK	Version	Date de sortie
Firebase JavaScript SDK	3.7.0	2017-03-01
Kit de développement logiciel Firebase C ++	3.0.0	2107-02-27
SDK Firebase Unity	3.0.0	2107-02-27
Firebase iOS SDK	3.14.0	2017-02-23
Firebase Android SDK	10.2	2017-02-15
SDK Admin Node.js de Firebase	4.1.1	2017-02-14
Firebase Admin Java SDK	4.1.2	2017-02-14

Exemples

Ajouter Firebase à votre projet Android

Voici les étapes nécessaires pour créer un projet Firebase et pour le connecter à une application Android.

Ajouter Firebase à votre application

1. Créez un projet Firebase dans la [console Firebase](#) et cliquez sur **Créer un nouveau projet** .
2. Cliquez sur **Ajouter Firebase à votre application Android** et suivez les étapes de configuration.
3. Lorsque vous y êtes invité, entrez **le nom du package de votre application** .

Il est important de saisir le nom du package utilisé par votre application. Cela ne peut être défini que lorsque vous ajoutez une application à votre projet Firebase.

4. À la fin, vous allez télécharger un fichier `google-services.json`. Vous pouvez télécharger ce fichier à tout moment. (ce fichier est situé sous le paramètre de projet dans la console Firebase).
5. Passer android studio View à Project et coller le fichier `google-service.json` sous le dossier `app`

L'étape suivante consiste à ajouter le SDK pour intégrer les bibliothèques Firebase dans le projet.

Ajouter le SDK

Pour intégrer les bibliothèques Firebase dans un de vos propres projets, vous devez effectuer quelques tâches de base pour préparer votre projet Android Studio. Vous l'avez peut-être déjà fait dans le cadre de l'ajout de Firebase à votre application.

1. Ajoutez des règles à votre fichier `build.gradle` niveau `build.gradle` pour inclure le **plug-in google-services** :

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

Ensuite, dans le fichier Gradle de votre module (généralement `app/build.gradle`), ajoutez la ligne de plug-in `apply` au bas du fichier pour activer le plug-in Gradle:

```
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'
```

Remarques:

- Les données ne peuvent pas être en lecture / écriture sans authentification. Si vous le voulez sans authentification. Modifier les règles dans la console de base de données

Firebase.

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

- Ajouter une autorisation Internet dans Manifest
- Mettre à niveau les services Google Play et Google Repository

Ecrire une valeur simple dans la base de données

D'abord, terminez l' [installation et la configuration](#) pour connecter votre application à Firebase. Ensuite, de n'importe où dans votre classe, vous pouvez écrire:

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

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

Il va écrire `Hello, World!` dans le nœud du `message`, comme ci-dessous:

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

Explication

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

Le code ci-dessus affectera l'instance `FirebaseDatabase` dans l'objet de `database` pour une utilisation ultérieure.

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

Le code ci-dessus fera référence à l'objet `myRef` dans l'enfant `"message"` du parent de votre projet (dans cet exemple, il s'agit de `"your-project-parent"`). C'est donc `"your-project-parent/message"`

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

Le code ci-dessus définira `"Hello, World!"` dans le chemin référencé par `myRef`

Mapper automatiquement le modèle personnalisé à la structure de données

Après avoir défini quelques données sur la base de données et avoir une structure composée de plusieurs nœuds comme celui-ci;

```
"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
    }
}

```

vous pouvez classer les structures de données.

Création de classe

Créez une classe de modèle à définir sur la base de données.

```

@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;
    }
}

```

Quelques points à retenir lors de la création d'une classe de modèle que vous souhaitez mapper sur vos données:

1. Vous devez avoir un constructeur vide
2. L'étendue des variables / champs doit être publique, de sorte que le [DataSnapshot](#) renvoyé par la base de données puisse accéder à ces champs. Si vous ne le faites pas, lorsque vous souhaitez obtenir des données, DataSnapshot ne peut pas accéder à votre modèle en rappel et cela entraînera une exception.
3. Les noms des variables / champs doivent correspondre à ceux de votre structure de données.

Envoi à Firebase

Créer un objet utilisateur

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

et référence

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

Vous avez maintenant la référence de votre base de données. Créez un nœud `user` avec `databaseReference.child("user")`. Si vous faites `.push()` vos modèles trouveront des identifiants uniques créés aléatoirement comme ci-dessus, `"-KdbKcU2ptfYF2xKb5a0"`, `"-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();
    }
});
```

Si vous souhaitez définir vos données sous votre clé spécifique, faites-le avec `.child("yourSpecificKey")` au lieu de `.push()`.

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

Lire Démarrer avec firebase-database en ligne: <https://riptutorial.com/fr/firebase-database/topic/3044/demarrer-avec-firebase-database>

Chapitre 2: Base de données FirebaseRealtime avec Android

Exemples

Ajouter la base de données en temps réel dans Android

1. Terminez l' [installation et la configuration](#) pour connecter votre application à Firebase. Cela créera le projet dans Firebase.
2. Ajoutez la dépendance pour la base de données Firebase Realtime à votre fichier `build.gradle` niveau du `build.gradle` :

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

3. Configurer les [règles de base de données Firebase](#)

Vous êtes maintenant prêt à travailler avec la base de données Realtime sous Android.

Par exemple, vous écrivez un message `Hello World` dans la base de données sous la clé de `message` .

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

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

Utiliser `setValue` pour enregistrer des données

La méthode `setValue()` remplace les données à l'emplacement spécifié, y compris les nœuds enfants.

Vous pouvez utiliser cette méthode pour:

1. Les types de transfert correspondant aux types JSON disponibles sont les suivants:
 - Chaîne
 - Longue
 - Double
 - Booléen
 - Carte `<String, Object>`
 - liste
2. Transmettez un objet Java personnalisé, si la classe qui le définit a un constructeur par défaut qui ne prend aucun argument et possède des getters publics pour les propriétés à

affecter.

Ceci est un exemple avec un CustomObject.
Définissez d'abord l'objet.

```
@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;
    }
}
```

Ensuite, obtenez la référence de la base de données et définissez la valeur:

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

Exemple d'insertion de données ou de récupération de données à partir de Firebase

Avant de commencer, il est nécessaire de suivre certaines configurations pour l'intégration du projet avec Firebase.

1. Créez votre projet dans Firebase Console et téléchargez le fichier google-service.json à partir de la console et placez-le dans le module de niveau application de votre projet. [Suivez le lien pour Create Project dans la console.](#)

2. Après cela, nous avons besoin d'ajouter une dépendance dans notre projet,

- Tout d'abord, ajoutez le chemin de classe dans notre niveau de projet,

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

- Et après avoir appliqué le plug-in au niveau de l'application, écrivez-le ci-dessous de la section sur les dépendances,

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

- Il y a pour plus de dépendance qui nécessitent d'ajouter dans le niveau de l'application gradle dans la section de dépendance

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

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

- Maintenant, commencez à insérer des données dans la base de données firebase, vous devez d'abord créer une instance de

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

après la création de l'objet FirebaseDatabase, nous allons créer notre DatabaseReference pour les données d'insertion dans la base de données,

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

Ici, `student` est le nom de la table si la table existe dans la base de données, puis insérer les données dans la table sinon en créer une nouvelle avec le nom de l'étudiant, après cela vous pouvez insérer des données en utilisant `databaseReference.setValue()`; fonction comme suit,

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

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

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

```
databaseReference.setValue(student);
```

Ici, j'insère des données en tant que hasmap Mais vous pouvez définir comme classe de modèle également,

- Commencez par récupérer les données de Firebase, nous utilisons ici `addListenerForSingleValueEvent` pour la valeur de lecture de la base de données,

```
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) {

    }

});
```

Obtenir la valeur / s de la base de feu

1. Créez une classe et ajoutez des importations pour analyser les informations:

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

//Declaration of firebase references
private DatabaseReference mDatabase;
```

```

//Declaration of firebase attributtes
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. Ajoutez `addListenerForSingleValueEvent()` à notre référence de base de données:

```

//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. Gonflez notre classe avec des informations `onDataChange()` dans l' `onDataChange()` :

```

@Override
public void onDataChange(DataSnapshot dataSnapshot) {

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

    //...
}

```

4. Enfin, nous pouvons obtenir différents attributs à partir de la classe Firebase:

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

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

Les meilleures pratiques

1. La base de données prend en charge 32 niveaux d'enfant différents, puis il est facile d'écrire des références erronées, pour éviter de créer des références privées finales:

```
//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. L' `onCancelled()` est appelé lorsque l'utilisateur n'a pas accès à cette référence par des règles de base de données. Ajoutez le code pertinent pour contrôler cette exception si vous en avez besoin.

Pour plus d'informations visitez [la documentation officielle](#)

Lire Base de données FirebaseRealtime avec Android en ligne: <https://riptutorial.com/fr/firebase-database/topic/6220/base-de-donnees-firebaserealtime-avec-android>

Chapitre 3: Bonjour le monde!

Exemples

Hello World en Android

1. Complétez la partie [Installation et configuration](#) . Cela créera le projet dans la console Firebase et installera également le SDK de base dans votre application Android.
2. Ajoutez la dépendance pour la base de données Firebase Realtime à votre fichier `build.gradle` niveau de l' `build.gradle` :

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

3. Maintenant, écrivez un message `Hello World` dans la base de données sous la clé de `message` .

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

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

4. Vérifiez si le message apparaît dans la `Realtime Database` .

Bonjour tout le monde dans IOS

1. Une fois que vous avez configuré Firebase sur votre projet IOS en suivant la documentation de configuration de Firebase pour IOS dans d'autres documents relatifs à FireBase, vous pouvez utiliser Firebase.
2. Si vous n'avez pas encore ajouté le module de base de données à votre Podfile, faites-le maintenant en ajoutant `pod Firebase/Database` afin d'obtenir la fonctionnalité de base de données pour Firebase
3. Poussez «Hello World» dans la base de données, mais vous devez d'abord modifier les règles de votre base de données dans le tableau de bord de votre application dans Firebase et la remplacer par

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

REMARQUE: Assurez-vous de modifier ce point avant de passer en production, car cela signifie que tout le monde peut lire et écrire dans votre base de données, ce qui

constitue une faille de sécurité majeure.

4. Maintenant, tout ce que vous avez à faire est d'importer Firebase dans le fichier que vous utilisez en utilisant `import Firebase` écrivez la ligne suivante et vous devriez voir un nœud 'Test' dans la base de données, développez-le et il devrait avoir un fils de "Hello World" !!!!!

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

CONGRATS SUR VOTRE PREMIER BASE DE DONNÉES

Lire **Bonjour le monde!** en ligne: <https://riptutorial.com/fr/firebase-database/topic/8885/bonjour-le-monde->

Chapitre 4: Firebase Real-Time Database avec Android

Exemples

Intégrer la base de données Firebase Real-Time à une application Android

Comment implémenter la base de données Firebase Real-Time dans les applications Android.

Configuration / installation:

1. Tout d'abord, installez le SDK Firebase ([guide](#))
2. Enregistrez votre projet à l'aide de la [console](#) Firebase
3. Après avoir effectué les deux étapes ci-dessus, ajoutez la dépendance suivante dans votre niveau d'application.

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

4. [Facultatif] Configurez vos règles de sécurité de base de données ([référence](#)).

Exemple d'implémentation:

1. Déclarez et initialisez la référence de base de données

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

Vous pourrez ensuite créer différentes références pour accéder à différents nœuds

6. Ecrire de nouvelles données dans la base de données

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

7. Lire les données de la base de données

```
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());
}
});
```

Lire Firebase Real-Time Database avec Android en ligne: <https://riptutorial.com/fr/firebase-database/topic/6341/firebase-real-time-database-avec-android>

Chapitre 5: Lecture des données

Exemples

Comprendre quelles données référencées par `getReference ()`

Dans cet exemple, nous utilisons cette base de données:

```
"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"
    }
  }
}
```

Si vous utilisez la base de données ci-dessus, alors:

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

pointera vers le parent de votre projet, les données `"your-project-name"`. Ainsi, le `dataSnapshot` vous avez acquis contiendra toutes les données qu'il contient, y compris toutes `"users"` données `"users"` et les `"books"`.

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

aura le même résultat, pointant vers `"your-project-name/users"`

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

aura le même résultat, pointant vers `"your-project-name/users/randomUserId1"`

Remarque: cet exemple est nécessaire pour bien comprendre [quelles données se trouvent dans l'objet dataSnapshot](#)

Comprendre quelles données se trouvent dans l'objet dataSnapshot

Remarque: Vous devez d'abord [connaître les données référencées par `getReference\(\)`](#) avant de pouvoir comprendre pleinement cet exemple.

Il existe trois méthodes courantes pour obtenir vos données à partir de la base de données Firebase Realtime:

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

Lorsque nous parlons des données `dataSnapshot` objet `dataSnapshot`, `addValueEventListener()` et `addListenerForSingleValueEvent()` sont fondamentalement les mêmes. La seule différence est que `addValueEventListener()` garde les modifications apportées aux données référencées, alors que `addListenerForSingleValueEvent()` ne l'est pas.

Alors considérez que nous avons cette base de données:

```
"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"
    }
  }
}
```

DataSnapshot produit par `addValueEventListener` et `addListenerForSingleValueEvent`

`dataSnapshot` produit par `addValueEventListener()` et `addListenerForSingleValueEvent()` contiendra la ou les valeurs des données exactes dans `addListenerForSingleValueEvent()` il est référencé. Comme quand `ref` est `dataSnapshot` sur `"your-project-name"` alors `dataSnapshot` devrait être:

```
... 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"
        }
    }
}

```

DataSnapshot produite par addChildEventListener

dataSnapshot produit par addChildEventListener() contiendra la ou les valeurs de données d'un niveau plus profond dans les données dataSnapshot il est référencé. Comme dans ces cas:

Quand ref est dataSnapshot sur "your-project-name" alors dataSnapshot devrait être:

```

... 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

```

Je sais que nous voudrions probablement utiliser `.getValue()` au lieu de `getKey()`. Mais ici, nous utilisons `getKey` car il contiendra toujours une chaîne et pas besoin de convertir en objet personnalisé, en carte ou autre. Fondamentalement, lorsque vous savez sur quelle clé dataSnapshot pointe, vous pouvez facilement savoir quelle valeur il contient et l'analyser dans votre propre objet personnalisé (ou n'importe quoi)

Lire Lecture des données en ligne: <https://riptutorial.com/fr/firebase-database/topic/9242/lecture-des-donnees>

Chapitre 6: Règles de base de données en temps réel Firebase

Remarques

Les règles de base de données Firebase Realtime déterminent qui a un accès en lecture et en écriture à votre base de données, la structure de vos données et les index existants. Ces règles résident sur les serveurs Firebase et sont appliquées automatiquement à tout moment. Chaque demande de lecture et d'écriture ne sera complétée que si vos règles le permettent. Par défaut, vos règles sont définies pour autoriser uniquement les utilisateurs authentifiés à accéder en lecture et en écriture à votre base de données. Cela permet de protéger votre base de données contre les abus jusqu'à ce que vous ayez le temps de personnaliser vos règles ou de configurer l'authentification.

Les règles de base de données Firebase ont une syntaxe de type JavaScript et sont de quatre types:

Rule Types	
<code>.read</code>	Describes if and when data is allowed to be read by users.
<code>.write</code>	Describes if and when data is allowed to be written.
<code>.validate</code>	Defines what a correctly formatted value will look like, whether it is a string, number, or object, and the data type.
<code>.indexOn</code>	Specifies a child to index to support ordering and querying.

Exemples

Autorisation

Identifier votre utilisateur n'est qu'une partie de la sécurité. Une fois que vous savez qui ils sont, vous avez besoin d'un moyen de contrôler leur accès aux données de votre base de données. Les règles de base de données Firebase vous permettent de contrôler les accès pour chaque utilisateur. Par exemple, voici un ensemble de règles de sécurité permettant à quiconque de lire le chemin `/foo/`, mais personne ne lui écrit:

```
{
  "rules": {
    "foo": {
      ".read": true,
```

```

    ".write": false
  }
}
}

```

`.read` règles `.read` et `.write` sont en cascade, donc cet ensemble de règles accorde un accès en lecture à toutes les données de path `/foo/` ainsi que des chemins plus profonds tels que `/foo/bar/baz`. Notez que les règles `.read` et `.write` qui autorisent l'accès remplaceront les autres règles de la base de données qui ne permettent pas l'accès; En d'autres termes, toutes les règles applicables, `.read` et `.write` sont ORed ensemble). Ainsi, l'accès en lecture à `/foo/bar/baz` serait toujours accordé dans cet exemple, même si une règle du chemin `/foo/bar/baz` évaluée à `false`.

Les règles de base de données Firebase incluent des variables et des fonctions intégrées qui vous permettent de vous référer à d'autres chemins, à des horodatages côté serveur, à des informations d'authentification, etc. Voici un exemple de règle qui accorde un accès en écriture aux utilisateurs authentifiés à `/users/<uid>/`, où est l'ID de l'utilisateur obtenu via l'authentification Firebase.

```

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

```

La validation des données

La base de données Firebase Realtime est schemaless. Cela permet de changer les choses au fur et à mesure de votre développement, mais une fois que votre application est prête à être distribuée, il est important que les données restent cohérentes. Le langage de règles inclut une règle `.validate` qui vous permet d'appliquer la logique de validation en utilisant les mêmes expressions que celles utilisées pour les règles `.read` et `.write`. La seule différence est que toutes les règles de validation pertinentes doivent être évaluées à `true` pour que l'écriture soit autorisée (en d'autres termes, toutes les règles `.validate` applicables sont regroupées pour permettre une écriture de base de données).

Ces règles imposent que les données écrites dans `/foo/` doivent être une chaîne de moins de 100 caractères:

```

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

```

Les règles de validation ont accès à toutes les mêmes fonctions et variables `.read` que les règles

`.read` et `.write`. Vous pouvez les utiliser pour créer des règles de validation prenant en compte les données se trouvant ailleurs dans votre base de données, l'identité de votre utilisateur, l'heure du serveur, etc.

Définition des index de base de données

La base de données en temps réel Firebase permet de commander et d'interroger des données. Pour les petites tailles de données, la base de données prend en charge les requêtes ad hoc, de sorte que les index ne sont généralement pas requis lors du développement. Cependant, avant de lancer votre application, il est important de spécifier des index pour toutes les requêtes que vous avez pour vous assurer qu'elles continuent à fonctionner à mesure que votre application se développe.

Les index sont spécifiés à l'aide de la règle `.indexOn`. Voici un exemple de déclaration d'index qui indexerait les champs de hauteur et de longueur pour une liste de dinosaures:

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

Lire Règles de base de données en temps réel Firebase en ligne: <https://riptutorial.com/fr/firebase-database/topic/3348/regles-de-base-de-donnees-en-temps-reel-firebase>

Chapitre 7: Requête Firebase

Introduction

Firebase Query peut être utilisé pour ordonner une collection de données basée sur certains attributs et restreinte à la grande liste d'éléments (comme les données de discussion) jusqu'à un nombre approprié pour la synchronisation avec le client.

Tout comme avec une référence, vous pouvez recevoir des données d'une requête en utilisant la méthode `on()`. Vous ne recevrez que des événements et des `DataSnapshots` pour le sous-ensemble des données correspondant à votre requête.

Exemples

Exemple de requête Firebase

```
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) {

        }
    });
}
```

Lire Requête Firebase en ligne: <https://riptutorial.com/fr/firebase-database/topic/10100/requete-firebase>

Chapitre 8: Transactions de base de données en temps réel Firebase

Introduction

Les transactions fournissent un mécanisme de coordination entre plusieurs parties pouvant accéder simultanément aux mêmes données. Ces "parties" peuvent être différentes instances du même code comme différents utilisateurs exécutant la même application ou des nœuds dans un cluster de serveurs, des parties du même programme ou des programmes différents comme une application d'administration, une application "utilisateur final" et / ou " backend "logique du serveur.

Exemples

Un compteur distribué

Imaginez que de nombreux utilisateurs exécutent tous une application Web qui tente d'incrémenter un compteur dans la base de données. Chaque utilisateur doit lire le compte actuel, en ajouter un et écrire la valeur mise à jour. Pour vous assurer que personne ne lit le compteur pendant que quelqu'un en ajoute un, nous utilisons une transaction:

```
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
  }
});
```

Lire Transactions de base de données en temps réel Firebase en ligne:

<https://riptutorial.com/fr/firebase-database/topic/9612/transactions-de-base-de-donnees-en-temps-reel-firebase>

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec firebase-database	Abdul Wasae , Adarsh Madrecha , AtaerCaner , Community , ErstwhileIII , Gabriele Mariotti , koceeng , Nepster , Shiven , TwiterZX , Veeresh Charantimath
2	Base de données FirebaseRealtime avec Android	Dhaval Solanki , Gabriele Mariotti , Merlí Escarpenter Pérez , ThunderStruct
3	Bonjour le monde!	noob , RyanM , ThunderStruct
4	Firebase Real-Time Database avec Android	Dhaval Solanki , Krishna Kumar , ThunderStruct
5	Lecture des données	koceeng
6	Règles de base de données en temps réel Firebase	Adarsh Madrecha , mckoss
7	Requête Firebase	Dhaval Solanki
8	Transactions de base de données en temps réel Firebase	Mike