com.google.firebase

Classes
FirebaseApp: Firebase SDK의 entry point
FIrebaseOptions: Firebase option의 Configuration Class
FirebaseOptions.Builder: Firebase option의 제작을 위한 Builder Class

Exceptions
FirebaseApiNotAvailabeException: 요청한 API가 유효하지 않음을 나타냅니다.
FirebaseException: Firebase Exception의 Base Class
FirebaseNetworkException: Firebase service에 대한 요청이 Network Error 때문에 Fail 한 경우 throws Exception. Device의 Network 연결 상태를 확인하거나 조금 기다렸다가 시도해보는 것이 좋습니다.
FirebaseTooManyRequestException: Firebase service에 대한 요청이 같은 기기로부터 계속해서 너무 많은 요청이 올 때 block되며 throws Exception. 조금 뒤에 다시 요청을 시도하는 것이 좋습니다.

FirebaseApp
Firebase SDK의 entry point이다. 일반적인 FIrebase API의 설정값이나 상태를 나타낸다.
Firebase App과 직접적으로 상호작용하지 않는다면 대부분의 App에서 사용할 필요가 없다.
Firebase API은 defult로 주어진 default FirebaseApp을 사용하는데,
FirebaseFoo.getInstance(firebaseApp)으로부터 온 API와 전혀 다르지 않다.
default app instance는 시작될 때 FirebaseInitPorvider에 의해서 초기화된다. 또한 default app instance는 Gradle manifest merging에 의해 app의 manifest로 추가된다. 만약 app이 서로 다른 build system을 사용한다면 provider는 app manifest에 추가된 내용을 필요로 한다.
initializeApp(Context, FirebaseOptions)가 App의 default app instance로 대신해서 초기화될 수 있으며, 이 method는 Application으로부터 적용된다. 이 method는 또한 이 class가 application의 main process 밖에서 사용될 경우 반드시 대신해서 초기화가 필요하다.

> ***Constant Summary***
- String DEFAULT_APP_NAME

> ***Public Method Summary***
- boolean equls(Object o)
- Context getApplicationContext()
- static List<FirebaseApp> getApps(Context context): FirebaseApps의 모든 다수의 list를 return한다.
- static FirebaseApp getInstance(String name)
- static FirebaseApp getInstance()
- String getName()
- FirebaseOptions getOptions()
- int hashCode()
- static FirebaseApp initializeApp(Context context, FirebaseOptions options, String name): FirebaseApp의 factory method initializer.
- static FirebaseApp initializeApp(Context context, FIrebaseOptions options):

FirebaseApp의 default instance initializer
- String toString()

> ***Inherited Method Summary***
- From class java.lang.Object

> ***Exceptions***
- public static FirebaseApp getInstance(String name) l public static FirebaseApp getInstance()
- Throws IllegalStatieException: FirebaseApp이 initializeApp(Context, FirebaseOptions, String)으로 초기화되지 않은 경우 발생한다.

- public FirebaseApp initializeApp(Context context, FirebaseOptions options, String name)
- Throws IllegalStateException: App의 이름은 같지만 다른 parameter에서 이미 initialized된 경우 발생한다.

FirebseOptions
public final class FirebaseOptions extends Object

> ***Nested Class Summary***
- class FirebseOptions.Builder: FirebaseOptions의 construct class

> ***Public Method Summary***
- boolean equals(Object o)
- static FirebaseOptions fromResource(Context context): FirebaseOptions에 있는 (string resource로부터 부여된) instance를 생성한다.
- String getApiKey(): app으로부터 인증을 요청할 때 사용되는 API key를 return한다. Google server가 app을 식별하는데 사용한다.
- String getApplicationID(): app에 유일하게 식별할 수 있는 Google API를 return한다.
- String getDatabaseUril(): database의 root URL을 return한다.
- String getGcmSenderId(): Google Developer의 console로부터 온 Project Number 이다. GCM(Google Cloud Messaging)을 설정하는데 사용한다.
- String getStorageBucket(): Google Cloud Storage의 bucket name을 return한다.
- int hashCode()
- String toString()

> ***Inherited Method Summary***
- From class java.lang.Object

com.google.firebase.database

Interfaces
- **ChildEventListener**
  Class들은 이 interface를 구현하면 DatabaseReference ref로부터 주어진 child의

location으로부터 데이터의 변경에 대한 receive events를 받을 수 있습니다. Listener를 location에 부착해서 사용할 수 있으며 Listener가 바뀌었을 때 trigger 되도록 하는 적절한 method입니다.

> ***Public Method Summary***

- abstract void **onCancelled**(DatabaseError error)
  이 method는 server에서 listener로 전송되는 모든 event 안에서 trigger되거나 FIrebase의 rule과 security의 결과에 따라서 제거될 때 trigger됩니다.
- abstract void **onChildAdded**(DataSnapshot snapshot, String previousChildName)
  이 method는 새로운 child가 이 listener가 부착된 location에 추가되었을 때 trigger 됩니다.
- abstract void **onChildChanged**(DataSnapshot snapshot, String previousChildName)
  이 method는 child location의 data가 변경되었을 때 trigger됩니다.
- abstract void **onChildMoved**(DataSnapshop snapshot, String previousChildName)
  이 method는 child location의 권한이 변경되었을 때 trigger됩니다.
- abstract void **onChildRemoved**(DataSnapshop snapshot)
  이 method는 부착된 listener의 location에서 child가 제거될 때 trigger됩니다.

- **DatabaseReference.CompletionListener**
  이 interface를 Database server로부터 명령에 대한 acknowledge를 받았을 때 notify받을 수 있습니다. 또한 완료된 것으로 간주할 수도 있습니다.

  > ***Public Method Summary***

  - abstract void **onComplete**(DatabaseError error, DatabseReference ref)
    이 method는 명령이 전달되었을 때 성공과 실패 여부에 상관 없이 trigger됩니다.

- **Logger**
  이 interface는 Firebase Database의 setup logging으로 사용됩니다.

  > ***Nested Class Summary***

  - enum **Logger.Level**
    Firebase Database library의 log level로 사용됩니다.

- **Transaction.Handler**
  Object를 이 interface를 구현할 경우 transaciton의 실행과 모든 transaction의 결과에 대해서 notify될 수 있습니다.

  > ***Public Method Summary***

  - abstract void **doTransaction**(MutableData currentData)
    이 method는 이 location에서 현재 data와 함께 호출됩니다. 여러 번 호출될 수 있습니다.
  - abstract void **onComplete**(DatbaseError error, boolean committed, DataSnapshot currentData)
    이 method는 transaction의 result와 함께 한번만 불립니다.

- **ValueEventListener**
  Class들이 이 interface를 구현할 경우 location의 data change에 대한 receive events를 사용할 수 있습니다.

  > ***Public Method Summary***
  - abstract void **onCancelled**(DatabaseError error)
    이 method는 server에서 listener로 전송되는 이벤트에서 trigger되거나 Firebase의 Rule이나 Security의 결과에 따라서 제거될 때 trigger됩니다.
  - abstract void **onDataChange**(DataSnapshot snapshot)
    이 method는 이 location에서 data의 snapshot과 함께 호출됩니다.

Classes

- **DatabaseError**
  DatabaseError의 instance들은 operation이 실패했을 때 callback으로 넘겨집니다. 발생한 error에 대한 세부정보 또한 포함되어 넘겨집니다.

  > ***Content Summary***
  - int **DATA_STATE**
    내부 사용 변수
    Constant Value = -1
  - int **DISCONNECTED**
    operation이 Nework disconnect 때문에 aborted된 경우.
    Constant Value = -4
  - int **EXPIRED_TOKEN**
    제공된 auth token이 만료된 경우.
    Constant Value = -6
  - int **INVALID_TOKEN**
    특정 authentication token이 유효하지 않은 경우. token이 malformed하거나, 만료되었거나, 생성될 때 revoked된 경우 발생할 수 있습니다.
    Constant Value = -7
  - int **MAX_RETRIES**
    transaction이 너무 많은 재시도(재요청)를 하는 경우.
    Constant Value = -8
  - int **NETWORK_ERROR**
    operation이 network error 때문에 preformed될 수 없는 경우.
    Constant Value = -24
  - int **OPERATION_FAILED**
    server가 operation이 failed 되었다고 나타내는 경우.
    Constant Value = -2
  - int **OVERRIDDEN_BY_SET**
    transaction이 subsequent set에 의해서 override된 경우.
    Constant Value = -9
  - int **PERMISSION_DENIED**
    client가 해당 operation을 수행하는데에 대한 permission을 가지고 있지 않은 경우.
    Constant Value = -3
  - int **UNAVAILABLE**
    service가 unavailable한 경우.

Constant Value = -10
  - int **UNKNOWN_ERROR**
    알 수 없는 error가 발생한 경우.
    Constant Value = -999
  - int **USER_CODE_EXCEPTION**
    user code 안에서 exception이 발생한 경우.
    Constant Value = -11
  - int **WRITE_CANCELED**
    write가 local에서 실패한 경우.
    Constant Value = -25

> ***Public Method Summary***
  - static DatabaseError **fromException**(Throwable e)
  - public int **getCode**() { **return** error에 따른 status code중 하나를 return합니다. }
  - public String **getDetails**() { **return** error와 부가 정보를 사람이 읽을 수 있는 형태로 return합니다. }
  - public String **getMessage**() { **return** error와 부가 정보를 사람이 읽을 수 있는 형태로 return합니다. }
  - public DatabaseException **toException**()
    FirebaseDatabase에서 third party로 Exception을 통합하려는 경우 사용할 수 있습니다.
  - public String **toString**()

- **DatabaseReference**
  Database location에 읽기나 쓰기를 할 때 사용하는 가장 대표적인 Firebase의 reference입니다. 이 class는 모든 Database operation들의 starting point입니다. URL과 함께 initailize한 뒤부터 data를 읽거나 쓸 수 있으며, 새로운 DatabaseReference를 생성할 수도 있습니다.

  > ***Nested Class Summary***
    - interface **DatabaseReference.CopletionListener**
      이 interface는 Database server로부터 operation이 acknowledge된 것을 notify 받거나 완료된 것으로 간주할 때 사용하는 method입니다.

  > ***Public Method Summary***
    - DatabaseRefence **child**(Stirng pathString) { **return** 주어진 path로부터 새로운 DatabaseReference를 return합니다. }
      pathStirng과 연관된 location으로부터 reference를 얻어옵니다.
    - boolean **equals**(Object object)
    - FirebaseDatabase **getDatabase**(Stirng pathString) { **return** 이 reference를 위한 Database object를 return합니다. }
      pathStirng과 연관된 location으로부터 reference를 얻어옵니다.
    - String **getKey**() { **return** location에서 이 reference로부터 lost token을 나타냅니다. }
    - DatabaseReference **getParent**() { **return** parent location의 DatabaseReference를 return합니다. 이 reference의 instance가 root location인 경우 null을 return합니다. }
    - DatabaseReference **getRoot**() { **return** 이 Firebase Database의 root

location의 reference를 return합니다. }

- static void **goOffline**()
  Firebase Database의 client가 server로부터 연결을 수동으로 해제하는 방법입니다. 또한 automatic reconnect를 비활성화합니다.
  Note: 이 *method*는 모든 *Firebase Databse*의 *connection*에 영향을 줍니다.
- static void **goOnline**()
  Firebase Database의 client가 server로부터 연결을 수동으로 연결하는 방법입니다. 또한 automatic reconnect를 활성화합니다.
  Note: 이 *method*는 모든 *Firebase Databse*의 *connection*에 영향을 줍니다.
- int **hashCode**()
- OnDisconnect **onDisconnect**() { **return** 이 location에서 disconnect operation들을 관리하기 위한 object를 return합니다. }
  이 location에서의 disconnect operation에 접근을 제공합니다.
- DatabaseReference **push**() { **return** 새로운 location을 가리키는 DatabaseReference를 return합니다. }
  child location에서 자동적으로 생성된 reference를 생성합니다. child key는 client side에서 생성되며, 정렬을 목적으로 server's time의 추정치를 포함합니다 (incorporates an estimate of the server's time for sorting purpose). single client에서 gerated된 location은 생성되도록 정렬되며, 거의 대부분 client들 쪽으로 정렬됩니다.
- Task<Void> **removeValue**()
  이 location의 value를 'null'로 설정합니다.
- void **removeValue**(DatabaseReference.CompletionListener listener)
  이 location의 value를 'null'로 설정합니다.
- void **runTransaction**(Transaction.Handler handler, boolean fireLocalEvents)
  이 location의 transaction을 실행합니다.
- void **runTransaction**(Transaction.Handler handler)
  이 location의 transaction을 실행합니다.
- void **setPriority**(Object priority, DatabaseReference.CompletionListener listener)
  이 Databse location에서 data의 priority를 설정합니다.
- Task<Void> **setPriority**(Object priority)
  이 Databse location에서 data의 priority를 설정합니다.
- void **setValue**(Object value, Object priority, DatabaseReference.CompletionListener listener)
  parameter로 주어진 value에 대한 data와 priority를 설정합니다.
- void **setValue**(Object value, DatabaseReference.CompletionListener listener)
  parameter로 주어진 value에 대한 data를 설정합니다.
- Task<Void> **setValue**(Object value, Object priority)
  parameter로 주어진 value에 대한 data와 priority를 설정합니다.
- Task<Void> **setValue**(Object value)
  parameter로 주어진 value에 대한 data를 설정합니다.
- String **toString**()
- Task<Void> **updateChildren**(Map<String, Object> update)
  특정한 value들을 특정한 key들로 Update합니다.
- void **updateChildren**(Map<String, Object> update,

DatabaseReference.CompletionListener listener)
특정한 value들을 특정한 key들로 Update합니다.

> ***Inherited Method Summary***
  - From class com.google.firebase.database.Query
  - From class java.lang.Object

- DataSnapshot: DataSnapshot의 instance는 Firebase Database location의 data를 가지고 있습니다.
- FirebaseDatabase: Firebase Database에 접근할 수 있는 entry point입니다.
- GenericTypeIndicator<T>: JAVA가 type에 상관없는 generics를 구현하는 방법이 있기 때문에 조금 더 복잡한 방법으로 runtime에 generic collection으로 method의 property type을 결정하도록 사용할 수 있는 Class입니다.
- MutableData: location에서 encapsulate된 data와 priority의 Instance입니다.
- OnDisconnect: OnDisconnect Class는 이 client가 연결이 끊어질 경우 서버로부터 실행되는 명령을 관리할 때 사용됩니다.
- Query: Query Class(그리고 subclass인 DatabaseReference)는 Data를 읽는 데 사용됩니다.
- ServerValue: Firebase Database에 Placeholer values를 포함한 data를 기록할 때 사용합니다.
- Transaction: Transaction Class는 location에서 data를 전송할 때 기술적으로 encapsulate 할 수 있는 Class입니다.
- Transaction.Result: 이 Class는 대표적으로 doTransaction이라는 method가 Transaction의 실행 중에 유일하게 밖으로 전송되기를 원합니다.

Enums

- Logger.Level: Firebase Databse의 library에서 Log의 Level로 사용됩니다.

Exceptions

- DatabaseException: 이 error는 Firebase Database의 library가 주어진 명령에 대해서 input하지 못 할 경우 throw됩니다.