

Firestore 雲端服務

James Chen

jameschen@hust.edu.tw

Firebase 主要服務

- **Cloud Messaging (FCM/GCM)**
 - Send/receive notifications
- **Realtime Database (DataStore)**
 - JSON tree, NoSQL
 - <https://youtu.be/U5aeM5dvUpA>
- **Authentication**
 - Integrate with identity providers or email
 - Ex: Google, Twitter, Facebook,

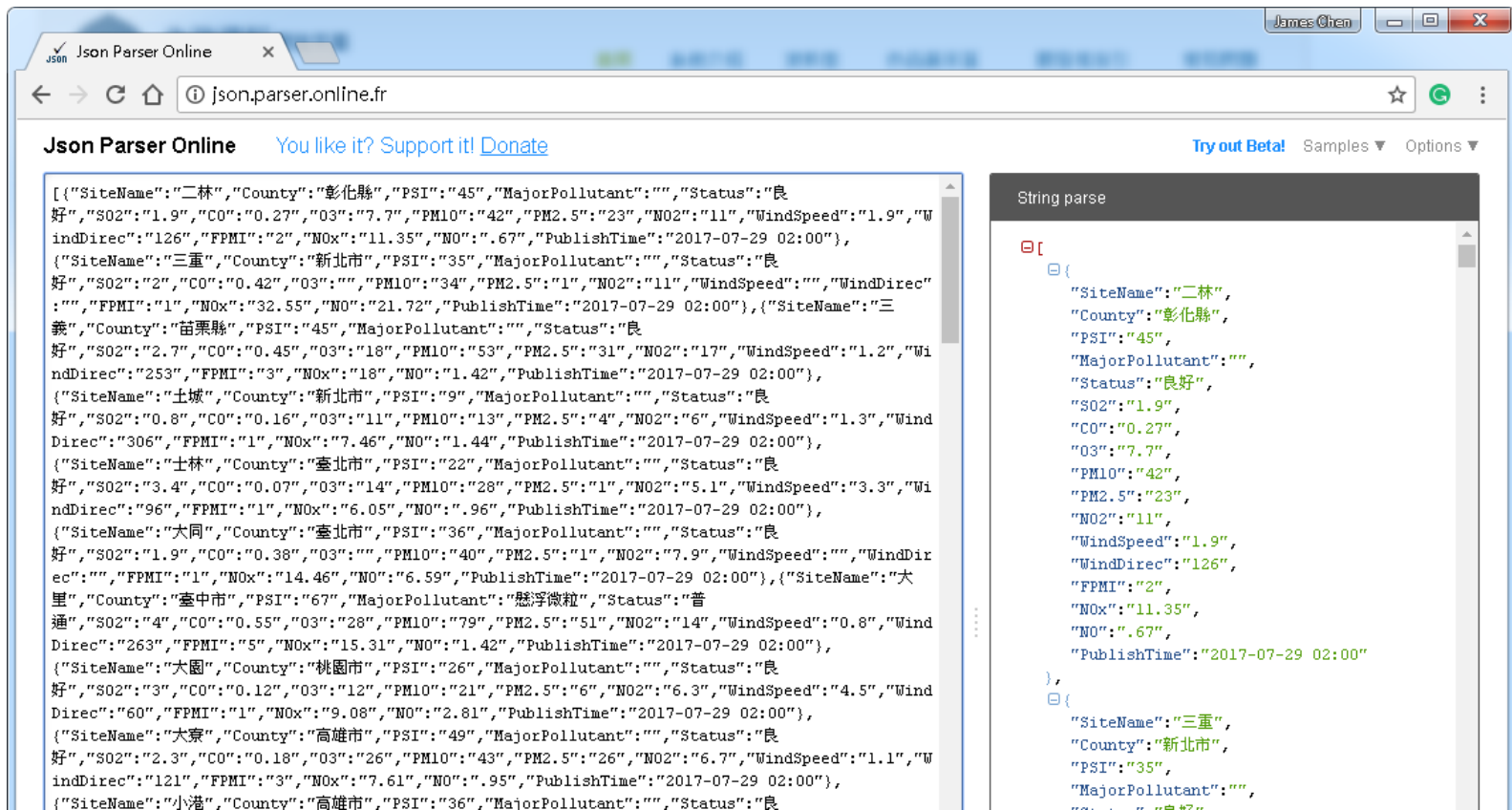
Firestore Database

The screenshot shows the Firebase Realtime Database console interface. The main content area displays a JSON tree structure for a user profile. The root node is 'bn105999ai2', which contains two child nodes: 'S123456' and 'S12345678'. The 'S123456' node contains a list of properties: 'grade: 4', 'height: 170.', 'isStudent: fals', 'loc', 'name: "James Che"', and 'password: 12345'. The 'loc' node is further expanded to show 'lat: 24.1234!' and 'lng: 120.6677!'. The 'S12345678' node contains a single child node 'led'.

A red cloud-shaped callout box is overlaid on the right side of the JSON tree, containing the text: **JSON Tree**
Key → values(Objects)

利用線上工具來解析 JSON 格式

- JSON Parser, JSON Editor, ...



The screenshot shows a web browser window with the URL `json.parser.online.fr`. The page title is "Json Parser Online" and it includes a navigation bar with "You like it? Support it! Donate", "Try out Beta!", "Samples", and "Options".

The main content area is split into two panels:

- Left Panel (Raw JSON):** Contains a large JSON array of objects. Each object represents an air quality station with fields like "SiteName", "County", "PSI", "MajorPollutant", "Status", "S02", "CO", "O3", "PM10", "PM2.5", "NO2", "WindSpeed", "WindDirec", "FPMI", "NOx", and "NO".
- Right Panel (String parse):** Shows the JSON array converted into a structured object format, with the same fields as the raw JSON, but with proper indentation and quotes for readability.

```
[{"SiteName": "二林", "County": "彰化縣", "PSI": "45", "MajorPollutant": "", "Status": "良好", "S02": "1.9", "CO": "0.27", "O3": "7.7", "PM10": "42", "PM2.5": "23", "NO2": "11", "WindSpeed": "1.9", "WindDirec": "126", "FPMI": "2", "NOx": "11.35", "NO": ".67", "PublishTime": "2017-07-29 02:00"}, {"SiteName": "三重", "County": "新北市", "PSI": "35", "MajorPollutant": "", "Status": "良好", "S02": "2", "CO": "0.42", "O3": "", "PM10": "34", "PM2.5": "11", "NO2": "11", "WindSpeed": "", "WindDirec": "", "FPMI": "1", "NOx": "32.55", "NO": "21.72", "PublishTime": "2017-07-29 02:00"}, {"SiteName": "三義", "County": "苗栗縣", "PSI": "45", "MajorPollutant": "", "Status": "良好", "S02": "2.7", "CO": "0.45", "O3": "18", "PM10": "53", "PM2.5": "31", "NO2": "17", "WindSpeed": "1.2", "WindDirec": "253", "FPMI": "3", "NOx": "18", "NO": "1.42", "PublishTime": "2017-07-29 02:00"}, {"SiteName": "土城", "County": "新北市", "PSI": "9", "MajorPollutant": "", "Status": "良好", "S02": "0.8", "CO": "0.16", "O3": "11", "PM10": "13", "PM2.5": "4", "NO2": "6", "WindSpeed": "1.3", "WindDirec": "306", "FPMI": "1", "NOx": "7.46", "NO": "1.44", "PublishTime": "2017-07-29 02:00"}, {"SiteName": "士林", "County": "臺北市", "PSI": "22", "MajorPollutant": "", "Status": "良好", "S02": "3.4", "CO": "0.07", "O3": "14", "PM10": "28", "PM2.5": "11", "NO2": "5.1", "WindSpeed": "3.3", "WindDirec": "96", "FPMI": "1", "NOx": "6.05", "NO": ".96", "PublishTime": "2017-07-29 02:00"}, {"SiteName": "大同", "County": "臺北市", "PSI": "36", "MajorPollutant": "", "Status": "良好", "S02": "1.9", "CO": "0.38", "O3": "", "PM10": "40", "PM2.5": "1", "NO2": "7.9", "WindSpeed": "", "WindDirec": "", "FPMI": "1", "NOx": "14.46", "NO": "6.59", "PublishTime": "2017-07-29 02:00"}, {"SiteName": "大里", "County": "臺中市", "PSI": "67", "MajorPollutant": "懸浮微粒", "Status": "普通", "S02": "4", "CO": "0.55", "O3": "28", "PM10": "79", "PM2.5": "51", "NO2": "14", "WindSpeed": "0.8", "WindDirec": "263", "FPMI": "5", "NOx": "15.31", "NO": "1.42", "PublishTime": "2017-07-29 02:00"}, {"SiteName": "大園", "County": "桃園市", "PSI": "26", "MajorPollutant": "", "Status": "良好", "S02": "3", "CO": "0.12", "O3": "12", "PM10": "21", "PM2.5": "6", "NO2": "6.3", "WindSpeed": "4.5", "WindDirec": "60", "FPMI": "1", "NOx": "9.08", "NO": "2.81", "PublishTime": "2017-07-29 02:00"}, {"SiteName": "大寮", "County": "高雄市", "PSI": "49", "MajorPollutant": "", "Status": "良好", "S02": "2.3", "CO": "0.18", "O3": "26", "PM10": "43", "PM2.5": "26", "NO2": "6.7", "WindSpeed": "1.1", "WindDirec": "121", "FPMI": "3", "NOx": "7.61", "NO": ".95", "PublishTime": "2017-07-29 02:00"}, {"SiteName": "小港", "County": "高雄市", "PSI": "36", "MajorPollutant": "", "Status": "良
```

Firestore 即時資料庫的特點

1. An GUI based Management Console;
2. On the cloud for any devices;
3. Smart **authentication** provides secure access;
4. Being a real-time database;
5.

<https://console.firebase.google.com>

Firebase 主控台 - 建立專案

- <https://console.firebase.google.com> (Gmail帳號)



Firebase 主控台 - 專案總覽

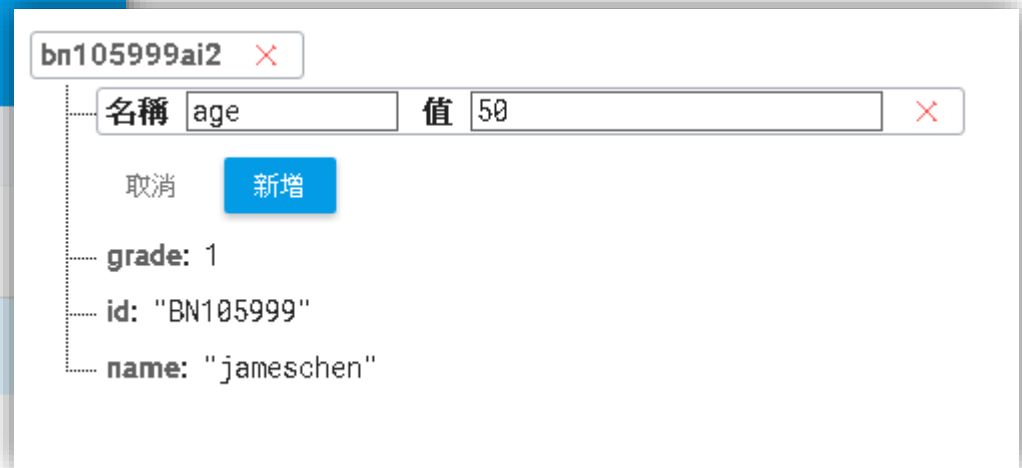
The image shows a screenshot of the Firebase console interface. The browser address bar displays `https://console.firebase.google.com/u/1/project/bn105999ai2/overview`. The main content area is titled "Realtime Database" and includes a navigation menu with options: "資料", "規則", "備份", and "使用情況". A notification banner at the top of the content area reads "★ 預設安全性規則要求使用者進行驗證" with links for "瞭解詳情" and "關閉". Below the notification, there is a data entry field showing "bn105999ai2: null" with "+" and "x" icons. At the bottom of the content area, there is a server icon and the text "為所有連結的用戶端即時儲存及同步處理資料", along with links for "瞭解詳情" and "查看文件".

The left sidebar contains a navigation menu with the following items: Overview, Analytics, 開發, Authentication, Database (highlighted with a red arrow), Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, 拓展, Notifications, Remote Config, Dynamic Links, and Spark (with subtext "免費，每個月 \$0 美元" and a "升級" button).

The browser tabs show "BN105999AI2 - 總覽 - F x" and "BN105999AI2 - Realtime x". The browser address bar for the second tab shows `https://console.firebase.google.com/u/1/project/bn105999ai2/database/data`.

線上維護 Firebase 的資料

- **Firestore 資料是以 key, value 方式儲存 (JSON)**
- **新增:** 點擊 **[+]** → 輸入對應的 key 與 value, 再按 **【新增】**
- 例如: 加入你自己的個人資料; 類似以下內容:
 - name(姓名), id(學號), grade(年級), age(年紀), ...



請手動輸入測試資料 ...

```
1.  {
2.  "S123456":
3.  {
4.      "name": "S123456",
5.      "password": "123456",
6.      "grade": 4,
7.      "height": 170.5,
8.      "weight": 75.5,
9.      "isStudent": false,
10.     "loc": { "lat":24.123456, "lng":120.667788}
11.  }
12. }
```

一件重要的事情 ... 設定權限

- 至少開放 read 權限：改為 true → 無須登入授權

The image consists of two screenshots of the Realtime Database console interface, illustrating the steps to update permissions.

Left Screenshot: Shows the '規則' (Rules) tab. A red circle highlights the '規則' tab, labeled with a '1' in a box. Below it, a code editor shows the following JSON structure:

```
1 {
2   "rules": {
3     ".read": "auth != null",
4     ".write": "auth != null"
5   }
6 }
```

A red box highlights the values "auth != null" for both ".read" and ".write", labeled with a '2' in a box.

Right Screenshot: Shows the '規則' (Rules) tab. A blue box at the top contains the instruction: "先將雙引號內部改為 true 或 'true' 最後選擇 [發佈(Publish)]". A black arrow points from this box to a '發佈' (Publish) button, which is circled in red and labeled with a '3' in a box. Below the button, the code editor shows the updated JSON structure:

```
1 {
2   "rules": {
3     ".read": "true",
4     ".write": "true"
5   }
6 }
```

A red box highlights the updated values "true" for both ".read" and ".write".

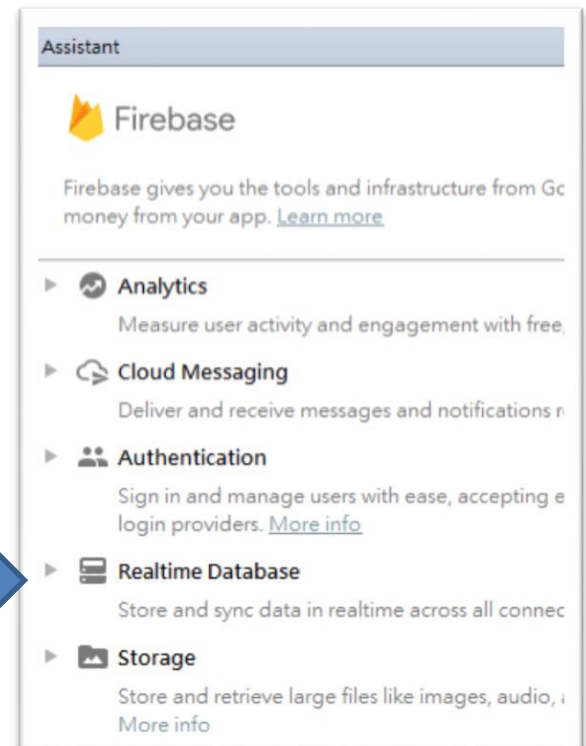
終端設備的存取 ...

- **Arduino/ESP8266/...**

- <https://github.com/firebase/firebase-arduino>

- **Android App ...**

- Studio : [Tools] > [Firebase]



設定 Android Studio專案

- 修改專案 **build.gradle**

```
dependencies {  
    classpath 'com.android.tools.build:gradle:3.1.4'  
    // ...  
    classpath 'com.google.gms:google-services:4.2.0'  
}
```

- 修改 **app/build.gradle**

- 加入外掛 **Firebase plugin** for Gradle

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

- 加入程式庫相關 (dependencies):

```
compile 'com.google.firebase:firebase-database:17.0.0'
```

存取 Firebase 基本指令

◆ 取得資料庫參考物件 (**DatabaseReference**)

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

• 儲存資料到Firebase 資料庫 : **setValue(...), updateChildren(...)**

// 單純 key(ref) → value

```
myRef.setValue("theValue");
```

// 有下一階資料 (內含多組 key:value → HashMap)

```
myRef.updateChildren( hashmap, aCompletionListener );
```

• 讀取資料(異動) : 使用 **ValueEventListener** , **ChildEventListener**

```
myRef.addValueEventListener(new ValueEventListener() {  
    public void onDataChange(DataSnapshot dataSnapshot) {  
        String value = dataSnapshot.getValue(String.class);  
    }  
    public void onCancelled(DatabaseError error) {  
        Log.w(TAG, "Failed to read value.", error.toException( ));  
    }  
});
```

寫入資料

```
// datatype: String, Long, Double, Boolean,
```

```
// datatype: Map<String, Object>, List<Object>
```

1. *FirebaseDatabase* *database* =

```
FirebaseDatabase.getInstance( );
```

2. *DatabaseReference* *myRef* =

```
database.getReference("S123456/regid");
```

3. *myRef.setValue*("1234567890");

讀取資料

```
1. DatabaseReference myRef = database.getReference("S123456");
2. myRef.addValueEventListener(new  ValueEventListener( ) {
3.     @Override
4.     public void onDataChange(DataSnapshot dataSnapshot) {
5.         // Called once: (1) initialize ; (2) data at this location is updated.
6.         String value = dataSnapshot.getValue(String.class);
7.         Log.d(TAG, "Value is: " + value);
8.     }

9.     @Override
10.    public void onCancelled(DatabaseError error) {
11.        // Failed to read value
12.        Log.w(TAG, "Failed to read value.", error.toException());
13.    }
14. });
```

即時監看Firebase 資料變化

(***ChildEventListener***)

```
Firestore.setAndroidContext(this);  
String url = "https://yourUrl.firebaseio.com/user";  
new Firebase(url).addChildEventListener(...);
```

```
DatabaseReference myRef = FirebaseDatabase.getInstance( ).getReference();  
myRef.addChildEventListener ( new ChildEventListener() {  
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {  
        // (String) dataSnapshot.child("name").getValue();  
    }  
    public void onChildRemoved(DataSnapshot dataSnapshot) {  
        // (String) dataSnapshot.child("name").getValue();  
    }  
    public void onChildMoved(DataSnapshot dataSnapshot, String s) { /* ... */ }  
    public void onCancelled(FirebaseError firebaseError) { }  
    public void onChildChanged(DataSnapshot dataSnapshot, String s) { }  
}  
);
```


Configure ProGuard (*)

1. # Add this global rule

-keepattributes Signature

2. # This rule will properly ProGuard all the model classes in

3. # the package **com.yourcompany.models**. Modify to fit the structure

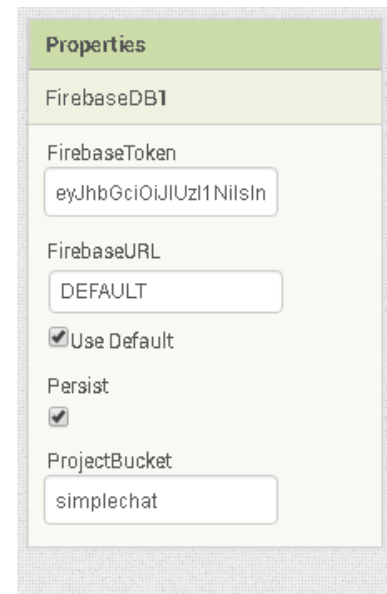
4. # of your app.

```
-keepclassmembers class com.yourcompany.models.** {  
    *;  
}
```

利用 App Inventor 存取 Firebase 資料

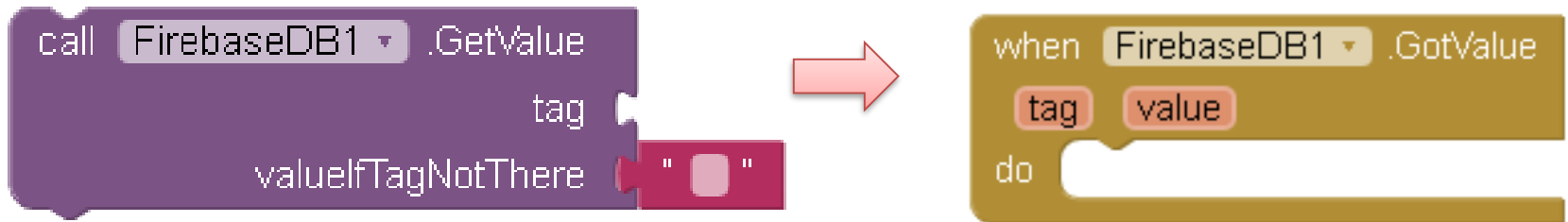
- **實驗元件 FirebaseDB** : 方便AI2 存取 Firebase 即時資料庫
 - 模擬器無法支援!! 必須安裝到實機、進行測試。
- **屬性**
 - FirebaseToken : 自動產生，不需更改
 - FirebaseURL : 必須在設計時就給定網址; 預設的測試網址: **DEFAULT**
 - FirebaseBucket : 一般是第一層名稱(key)
- **方法**
 - 讀取**標籤**值(GetValue)、儲存資料(StoreValue)、增加標籤資料(AppendValue)
 - 讀取並移除首項資料(RemoveFirst)
 - 讀取標籤清單(GetTagList)、清除指定標籤(ClearTag)
- **事件**
 - 取得數值(GotValue)
 - 資料改變(DataChanged)
 - 資料庫錯誤(FirebaseError)
 - 取得首項資料(FirstRemoved)
 - 取得標籤清單(TagList)

<https://xxx.firebaseio.com>



Firestore 主要積木使用時機

- 讀取資料：非同步取得資料



- 儲存資料：

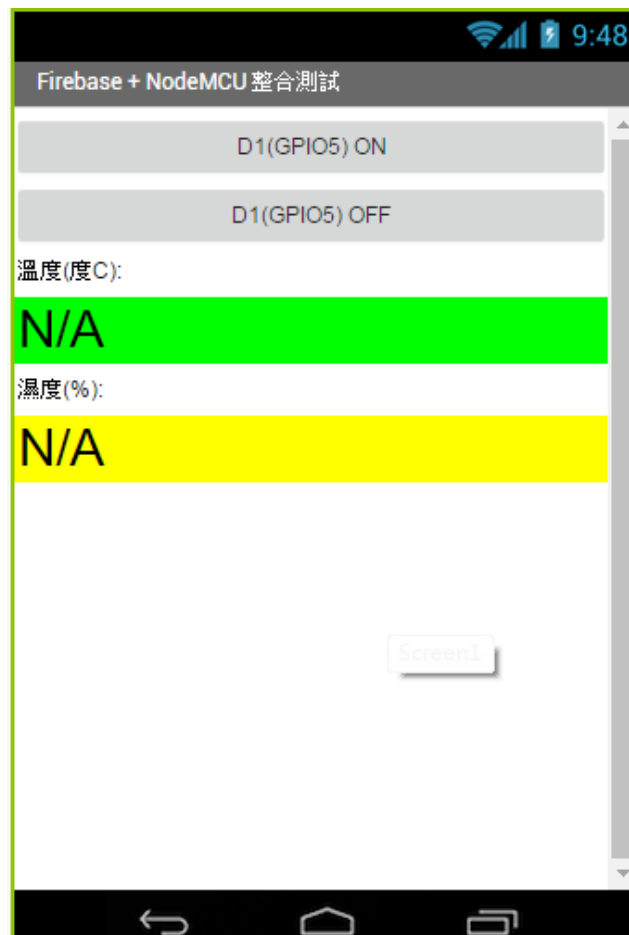


- 當資料被改變、或錯誤發生：主動通知



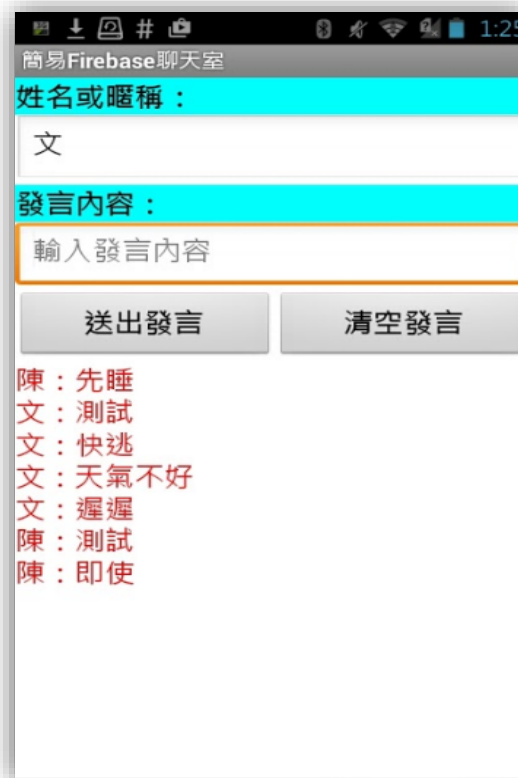
FirebaseTest 即時資料庫體驗

- 匯入專案 : Lab0308_FirebaseTest.aia



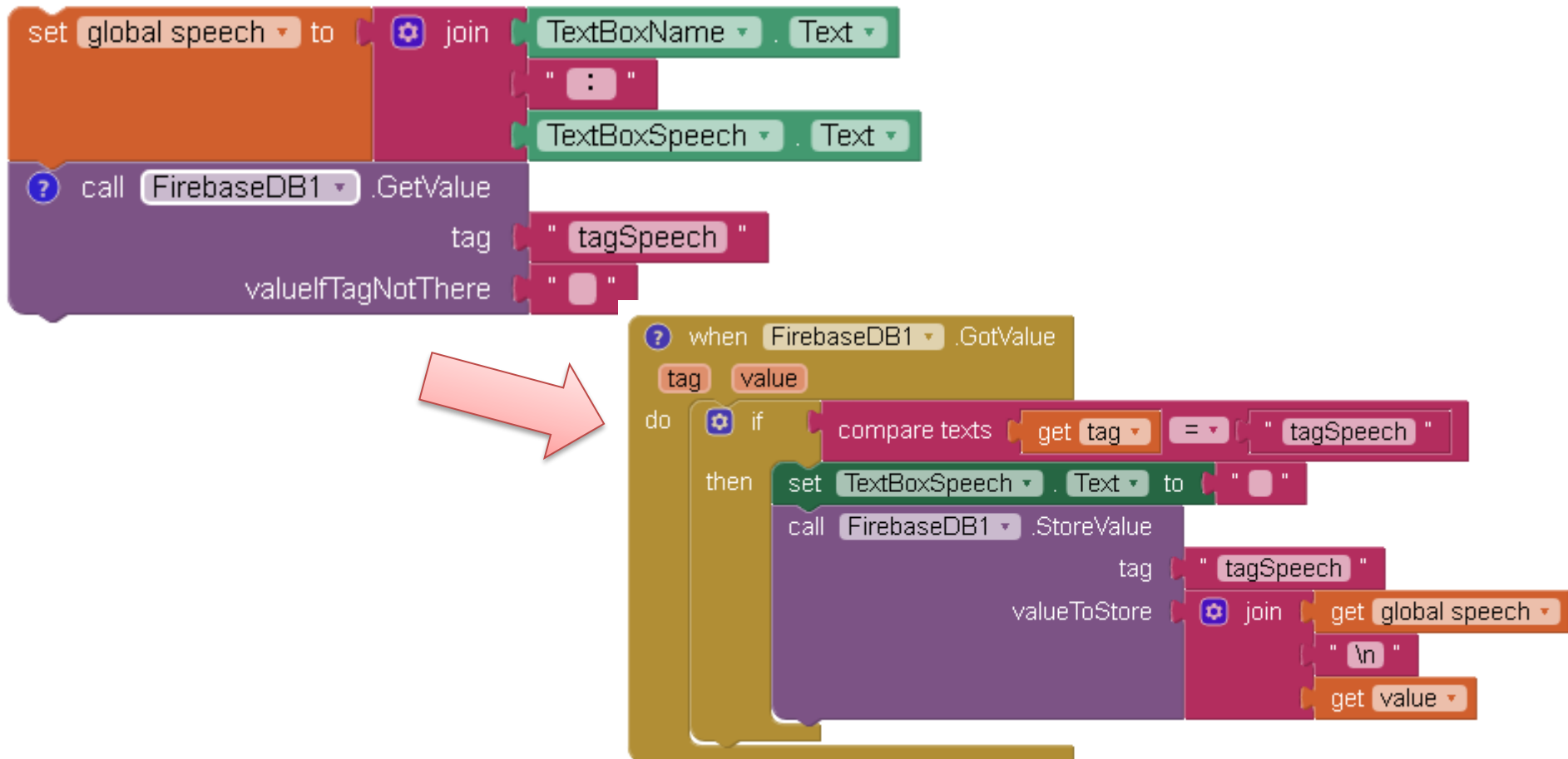
FirestoreChatroom 聊天室體驗

- 匯入專案 : Lab0308_FirebaseChatroom.aia
- 一起聊天的使用者必須安裝同一App



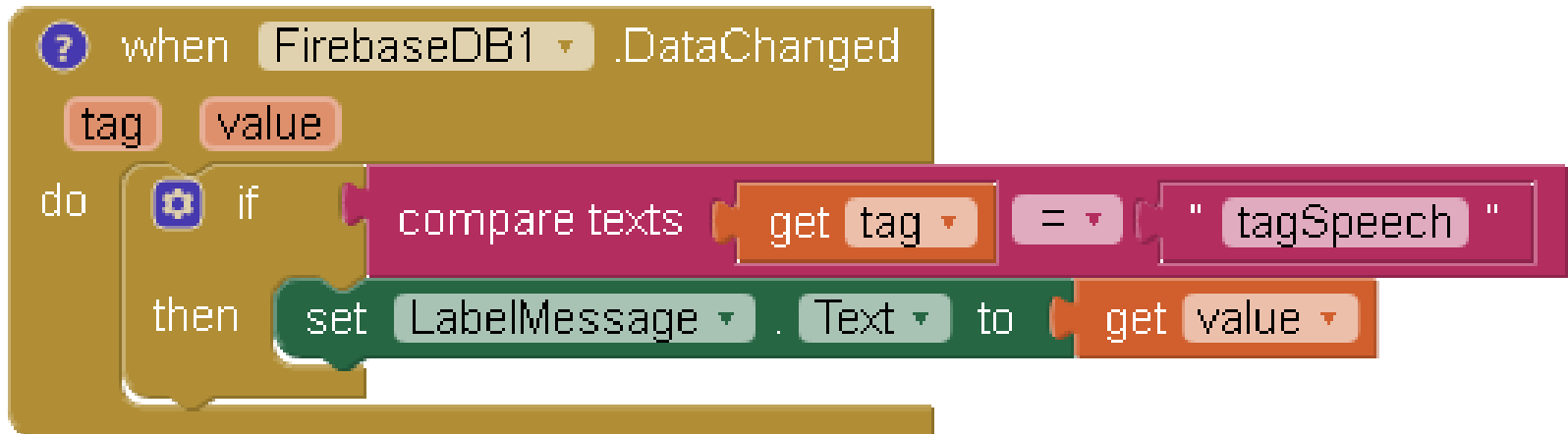
Firestore 聊天室的基本架構-1

- 發言時：
 - 確認必要資料都輸入完成後，先讀取遠端Firestore的聊天室內容
 - 取得最新的聊天室內容後，加上自己的發言，再回寫到遠端Firestore



Firestore 聊天室的基本架構-2

- 聊天室內容有更新時(會主動通知)：
 - 判斷為聊天室內容之後，更新到畫面顯示區



```
when FirestoreDB1 .DataChanged
  tag value
do
  if compare texts get tag = "tagSpeech"
  then set LabelMessage . Text to get value
```

The image shows a Scratch-style code block for handling Firestore data changes. The main block is a 'when FirestoreDB1 .DataChanged' event. It has two input fields: 'tag' and 'value'. Inside a 'do' loop, there is an 'if' condition: 'compare texts' with 'get tag' and 'tagSpeech' compared using an equals sign. If true, the 'then' block is executed: 'set LabelMessage . Text to get value'.