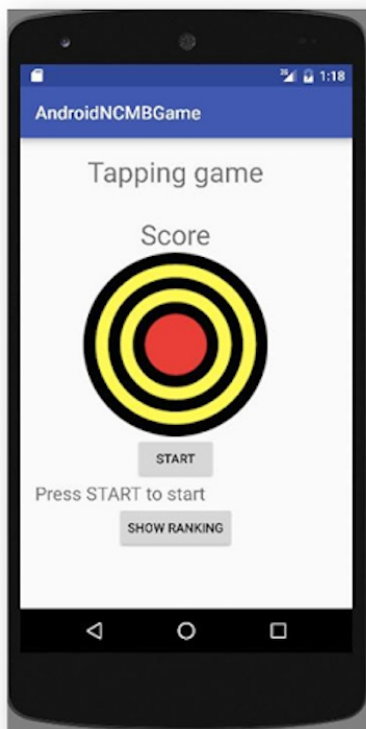


【Android問題集】 オンラインランキング機能を作 ってみよう！ 「連打ゲーム」

2017/05/16作成 (2017/05/19修正)



GitHub

<https://goo.gl/zZY5RA>

コンテンツ概要

- ニフティクラウドmobile backend の機能『データストア』を学習するための問題集です
 - ニフティクラウドmobile backend の利用登録（無料）が必要です。
- 問題用プロジェクトにはオンラインランキング機能が実装されていない状態の「連打ゲーム」です
 - 既に実装済みのニフティクラウドmobile backend を利用するための準備（SDK導入など）方法の詳細はこちらをご覧ください。

http://mb.cloud.nifty.com/doc/current/introduction/quickstart_android.html

問題について

- 問題は2問あります
- 2問クリアすると「連打ゲーム」にオンラインランキング機能を実装したアプリが完成します
- 問題に取り組む上で必要な開発環境は以下です
 - AndroidStudio ver.1.4 以上

問題に取り組む前の準備

プロジェクトのダウンロード

▼問題用プロジェクト▼

<https://github.com/natsumo/AndroidFirstApp/archive/Question.zip>

1. 上記リンクをクリックしてzipファイルをローカルに保存します
2. zipファイルを解凍して、AndroidStudioを開き、「Open an existing AndroidStudio project」を選択し、解凍したプロジェクトを開きます。
3. プロジェクトをビルドし、「連打ゲーム」で遊んでみましょう

「連打ゲーム」の操作方法

1. 「START」ボタンをタップします
2. 「3」, 「2」, 「1」とカウントダウンし、「スタート!」からタイムアップ!の10秒間「◎」の部分がタップできるようになります
3. 10秒間に何回タップできるかを競う単純なゲームです
4. 10秒経つと名前を入力するアラートが表示されますので、入力し「OK」をタップします
5. 「SHOW RANKING」ボタンをタップすると、画面に名前とスコアが表示されます。

※ **注意**：問題に取り組む前の状態では「SHOW RANKING」ボタンをタップしてもランキングは表示されません

アプリの新規作成とAPIキーの設定

mBaaS ダッシュボード

- ニフティクラウドmobile backend にログインしアプリの新規作成を行います
 - アプリ名はわかりやすいものにしましょう。例) 「renda」
- アプリが作成されるとAPIキーが2種類（アプリケーションキーとクライアントキー）発行されます
 - 次で使います。

Android Studio

- MainActivity.java の onCreate を編集します
- 先程ニフティクラウドmobile backend のダッシュボード上で確認したAPIキーを、それぞれ YOUR_APPLICATION_KEY と YOUR_CLIENT_KEY に貼り付けます

APIキー	
アプリケーションキー	3d4349360c600eb74b8013c615534c866b631139e02b61 <input type="button" value="コピー"/>
クライアントキー	384a065ef48f43fd251792c41b2aa38d20b969948bb31fda <input type="button" value="コピー"/>

重要
APPLICATION_KEYと
CLIENT_KEYは
mobile backendの
管理画面から
コピーボタンで
コピーして使います！

```
/****** mBaaS初期化 *****/  
NCMB.initialize(this.getContext(), "YOUR_APPLICATION_KEY", "YOUR_CLIENT_KEY");
```

- このとき、ダブルクォーテーション (") を消さないように注意してください！

【問題 1】

名前とスコアの保存を試みよう！

MainActivity.java を開きます。下図の **saveScore** メソッドを編集し、引数の **name**（アラートで入力した名前）と **score**（連打ゲームでタップした回数）の値をmBaaSに保存する処理をコーディングしてください

```
/*
 * mBaaSデータを保存
 */
public void saveScore(String name, int score) {

    // *****【問題1】名前とスコアを保存しよう！*****

    // *****

}
```

- データストアに保存先クラスを作成します
 - クラス名は「**GameScore**」としてください
- **name** を保存するフィールドを「**name**」、**score** を保存するフィールドを「**score**」として保存してください

ヒント

- ニフティクラウドmobile backend のキュメントのドキュメントページをご活用ください

http://mb.cloud.nifty.com/doc/current/datastore/basic_usage_android.html

コーディング後の作業

問題1のコーディングが完了したら、下記の作業を行います

【作業1-1】

それぞれ該当する箇所に以下の処理を追記して、実行時にAndroidStudio上にログを表示できるようにします

- 保存に失敗した場合の処理を行う箇所に追記

```
//保存が失敗した場合の処理  
Log.e("NCMB", "保存に失敗しました。エラー:" + e.getMessage());
```

- 保存に成功した場合の処理を行う箇所に追記

```
//保存が成功した場合の処理  
Log.i("NCMB", "保存に成功しました。");
```

【作業1-2】

エミュレーターで実行、「START」ボタンを押してゲームを遊びます

- 名前を入力し、「OK」がタップされると【問題1】で作成した `saveScore` メソッドが呼ばれ、データが保存されます
- このとき下記のいずれかのログが出力されます
 - 保存成功時：「保存に成功しました。」
 - 保存失敗時：「保存に失敗しました。エラー :*****」

※ エラーコードが出た場合はこちらで確認できます

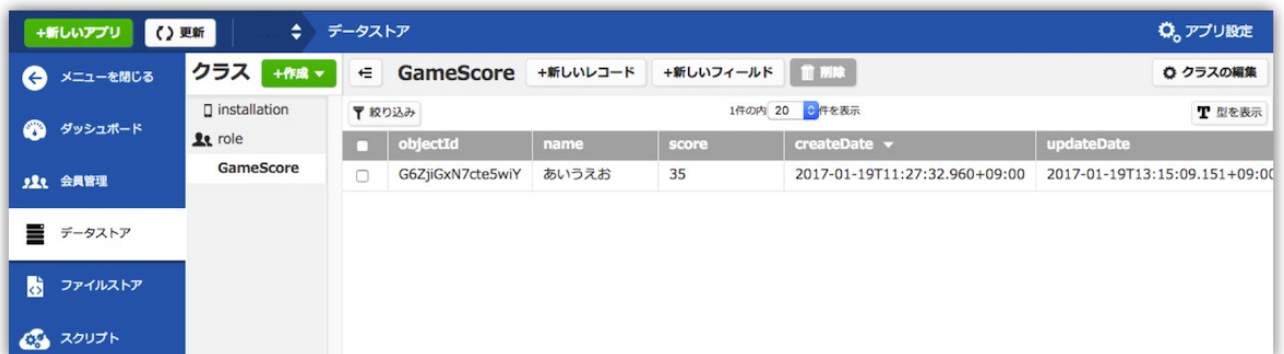
http://mb.cloud.nifty.com/doc/current/rest/common/error.html#REST_API
Iのエラーコードについて

【問題1】 答え合わせ

ニフティクラウドmobile backend上での確認

mBaaS ダッシュボード

- 保存されたデータを確認しましょう
 - 「データストア」をクリックすると、「GameScore」クラスにデータが登録されていることが確認できます。



The screenshot shows the mBaaS dashboard interface. On the left is a sidebar with navigation links: 'メニューを開く', 'ダッシュボード', '会員管理', 'データストア', 'ファイルストア', and 'スクリプト'. The 'データストア' (Data Store) link is selected. The main area displays the 'GameScore' class. Below the class name, there are tabs for 'installation', 'role', and 'GameScore'. The 'GameScore' tab is active, showing a table of data. The table has columns: 'objectId', 'name', 'score', 'createDate', and 'updateDate'. One record is visible with 'objectId' G6ZjiGxN7cte5wiY, 'name' あいうえお, 'score' 35, 'createDate' 2017-01-19T11:27:32.960+09:00, and 'updateDate' 2017-01-19T13:15:09.151+09:00.

objectId	name	score	createDate	updateDate
G6ZjiGxN7cte5wiY	あいうえお	35	2017-01-19T11:27:32.960+09:00	2017-01-19T13:15:09.151+09:00

- 上図はスコアが35連打で名前を「あいうえお」とした場合の例です。

コードの答え合わせ

Android Studio

- 模範解答は以下です

```
// *****【問題1】名前とスコアを保存しよう!*****  
//保存するインスタンスを作成  
NCMBObject obj = new NCMBObject("GameScore");  
//値を設定  
obj.put("name", name);  
obj.put("score", score);  
//保存を実施  
obj.saveInBackground(new DoneCallback() {  
    @Override  
    public void done(NCMBException e) {  
        if (e != null) {  
            //保存が失敗した場合の処理  
            Log.e("NCMB", "保存に失敗しました。エラー:" + e.getMessage());  
        } else {  
            //保存が成功した場合の処理  
            Log.i("NCMB", "保存に成功しました。");  
        }  
    }  
});  
// *****
```


【問題2】

ランキングを表示しよう！

RankingActivity.java を開きます。下図の checkRanking メソッドを編集し、データストアの GameScore クラスに保存した name と score のデータを score の降順（スコアの高い順）で検索・取得する処理をコーディングしてください

```
protected void checkRanking() {
```

```
    // ***** 【問題2】ランキングを表示しよう！*****
```

```
    // *****
```

```
}
```

- 検索データ件数は5件とします

ヒント

- ニフティクラウドmobile backend のキュメントのドキュメントページをご活用ください

http://mb.cloud.nifty.com/doc/current/datastore/basic_usage_android.html

http://mb.cloud.nifty.com/doc/current/datastore/ranking_android.html

コーディング後の作業

問題2のコーディングが完了したら、下記の作業を行います

【作業2-1】

該当する箇所に以下の処理を追記して、実行時にAndroidStudio上にログを表示できるようにします

- 検索に失敗した場合の処理を行う箇所に追記

```
//エラー時の処理
Log.e("NCMB", "検索に失敗しました。エラー:" + e.getMessage());
```

- 検索に成功した場合の処理を行う箇所に追記

```
//成功時の処理
Log.i("NCMB", "検索に成功しました。");
```

【作業2-2】

エミュレーターで実行し、「SHOW RANKING」ボタンをタップします

- 画面起動後、`checkRanking` メソッドが呼ばれ、【問題1】で保存されたデータが検索・取得されます
- このとき下記のいずれかのログが出力されます
 - 検索成功時：「検索に成功しました。」
 - 検索失敗時：「検索に失敗しました。エラーコード：*****」

※ エラーコードが出た場合はこちらで確認できます

http://mb.cloud.nifty.com/doc/current/rest/common/error.html#REST_APIのエラーコードについて

- 検索の状態（成功・失敗）に関係なく、「SHOW RANKING」ボタンをタップしても、まだランキングは表示されません

【作業2-3】

検索に成功したら、該当する箇所に以下の処理を追記して、取得した値から必要なデータを取り出し、ランキング画面へ反映させます

- 検索に成功した場合の処理を行う箇所に追記

```
//ListViewオブジェクトの取得
ListView lv = (ListView)findViewById(R.id.lstRanking);
// ループカウンタ
ArrayAdapter<String> adapter
    = new ArrayAdapter<String>(RankingActivity.this, android.R.layout.simple_list_item_1);

for (int i = 0, n = objects.size(); i < n; i++) {
    NCMBObject o = objects.get(i);
    Log.i("NCMB", o.getString("name"));
    // 処理
    String name = o.getString("name");
    Integer score = o.getInt("score");
    adapter.add(name + " さん : " + score.toString() + " (point)");
}

lv.setAdapter(adapter);
```

【作業2-4】

エミュレーターで実行、「SHOW RANKING」ボタンを押します

- 先ほどのスコアが表示されれば完成です！おめでとうございます★

【問題2】 答え合わせ

ランキング画面の確認

- ランキング画面を確認しましょう
 - アプリで「SHOW RANKING」をタップすると以下のようにランキングが表示されます



- 上図は3回遊んだ場合の例です。複数回遊んで、ランキングが表示されることを確認しましょう！

コードの答え合わせ

Android Studio

- 模範解答は以下です

```
// *****【問題2】ランキングを表示しよう!*****  
//HighScoreクラスを検索するクエリを作成  
NCMBQuery<NCMBObject> query = new NCMBQuery<NCMBObject>("GameScore");  
//Scoreフィールドの降順でデータを取得  
query.addOrderByDescending("score");  
//検索件数を5件に設定  
query.setLimit(5);  
//データストアでの検索を行う  
query.findInBackground(new FindCallback<NCMBObject>() {  
    @Override  
    public void done(List<NCMBObject> objects, NCMBException e) {  
        if (e != null) {  
            //エラー時の処理  
            Log.e("NCMB", "検索に失敗しました。エラー:" + e.getMessage());  
        } else {  
            //成功時の処理  
            Log.i("NCMB", "検索に成功しました。");  
            //ListViewオブジェクトの取得  
            ListView lv = (ListView)findViewById(R.id.lstRanking);  
            // ループカウンタ  
            ArrayAdapter<String> adapter  
                = new ArrayAdapter<String>(RankingActivity.this, android.R.layout.simple_list_item_1);  
  
            for (int i = 0, n = objects.size(); i < n; i++) {  
                NCMBObject o = objects.get(i);  
                Log.i("NCMB", o.getString("name"));  
                // 処理  
                String name = o.getString("name");  
                Integer score = o.getInt("score");  
                adapter.add(name + " さん : " + score.toString() + " (point)");  
            }  
  
            lv.setAdapter(adapter);  
        }  
    }  
});  
// *****
```

参考

- 問題の解答を実装した完全なプロジェクトをご用意しています

▼完成版プロジェクト▼

<https://github.com/natsumo/AndroidFirstApp/archive/AnswerProject.zip>

- APIキーを設定してご利用ください