【Swift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/thbHUr

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_ios.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
- 問題を取り組む上で必要な開発環境は以下です
  - Mac OS X 10.10(Yosemite)以上
  - Xcode ver.7 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▼
Swift3.0
https://github.com/natsumo/SwiftFirstApp/archive/Swift3.0_Question.zip
Swift2.0
https://github.com/natsumo/SwiftFirstApp/archive/Swift2.0_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、`SwiftFirstApp.xcworkspace`をダブルクリックしてXcodeでプロジェクトを開きます
3. アプリを実行し、「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. 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「ランキングを見る」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
  - 次で使用します。

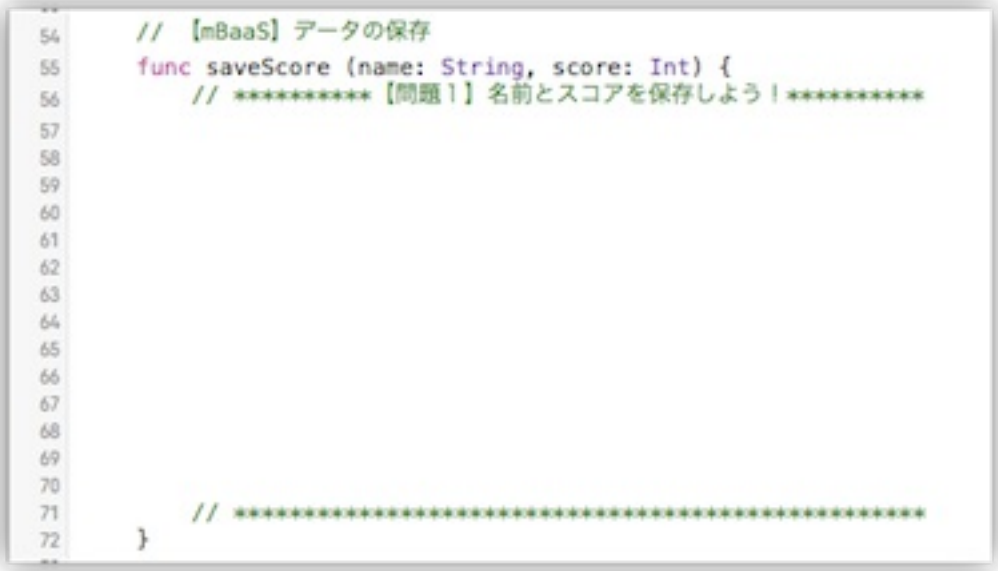
Xcode

- `AppDelegate.swift`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_NCMB_APPLICATION_KEY`と`YOUR_NCMB_CLIENT_KEY`に貼り付けます

- 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してみよう！

`GameViewController.swift` 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 mobile backend に保存する処理をコーディングしてください



※ 画像はSwift2.0 ver.です

- 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iOSドキュメントはObjective-Cで書かれていますので、Swiftに書き換えたものを用意しました

  http://qiita.com/natsumo/items/c00cf7a48e0f8cd8d236

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Xcode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
    - Swift3.0 の場合

```
let err = error as! NSError
// 保存に失敗した場合の処理
print("保存に失敗しました。エラーコード:\(err.code)")
```

    - Swift2.0 の場合

```
// 保存に失敗した場合の処理
print("エラーが発生しました。エラーコード:\(error.code)")
```

- 保存に成功した場合の処理を行う箇所に追記
    - Swift3.0 の場合

```
// 保存に成功した場合の処理
print("保存に成功しました。objectId:\(obj?.objectId)")
```

    - Swift2.0 の場合

```
// 保存に成功した場合の処理
print("保存に成功しました。objectId:\(obj.objectId)")
```

【作業1-2】

シ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。objectId:**********`」
    - 保存失敗時：「`エラーが発生しました。エラーコード:******`」

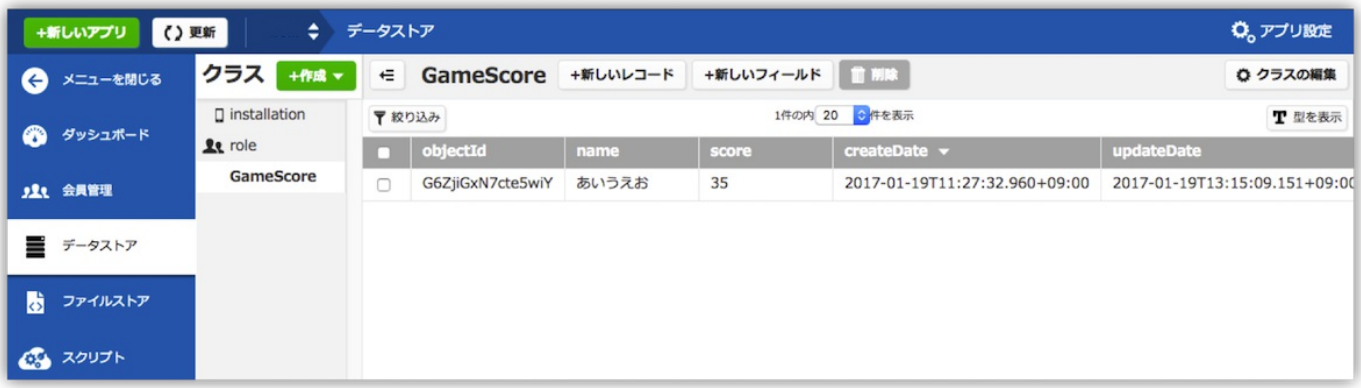
※エラーコードが出た場合はこちらで確認できます
http://mb.cloud.nifty.com/doc/current/rest/common/error.html#REST_APIのエラーコードについて

【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。



- 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

模範解答は以下です

Xcode

- Swift3.0 の場合

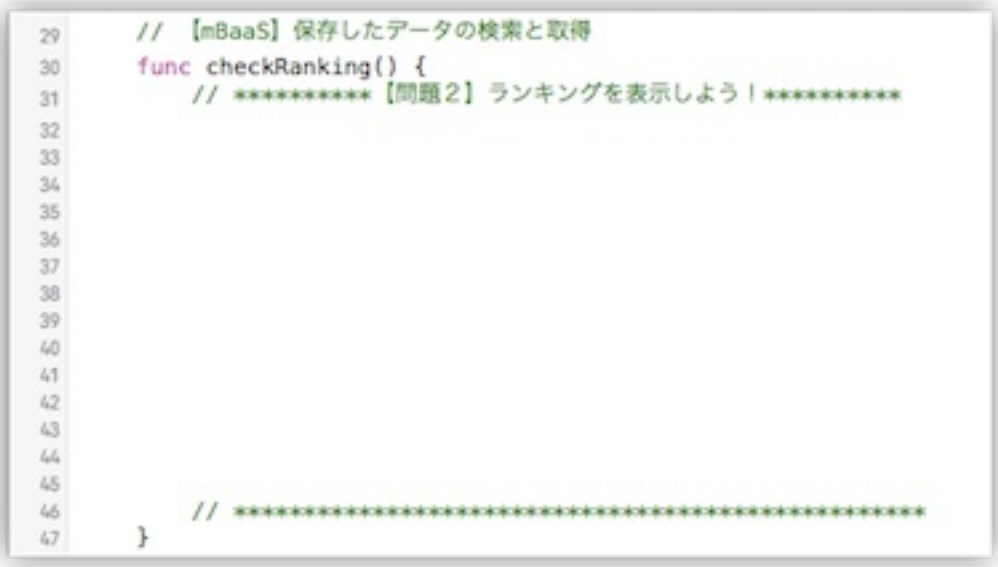
```
// **********【問題1】名前とスコアを保存しよう!**********
let obj = NCMBObject(className: "GameScore")
// 値を設定
obj?.setObject(name, forKey: "name")
obj?.setObject(score, forKey: "score")
// 保存を実施
obj?.saveInBackground({(error) in
    if error != nil {
        let err = error as! NSError
        // 保存に失敗した場合の処理
        print("保存に失敗しました。エラーコード:\(err.code)")
    }else{
        // 保存に成功した場合の処理
        print("保存に成功しました。objectId:\(obj?.objectId)")
    }
})
// ********************************************************
```

- Swift2.0 の場合

```
// **********【問題1】名前とスコアを保存しよう!**********
// 保存先クラスを作成
let obj = NCMBObject(className: "GameScore")
// 値を設定
obj.setObject(name, forKey: "name")
obj.setObject(score, forKey: "score")
// 保存を実施
obj.saveInBackgroundWithBlock{(error: NSError!) -> Void in
    if (error != nil) {
        // 保存に失敗した場合の処理
        print("保存に失敗しました。エラーコード:\(error.code)")
    }else{
        // 保存に成功した場合の処理
        print("保存に成功しました。objectId:\(obj.objectId)")
    }
}
// ********************************************************
```

【問題２】
ランキングを表示しよう！

`RankingViewController.swift`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



※画像はSwift2.0 ver.です

- 検索データ件数は５件とします
    - ただし、この値は「`rankingNumber`」としてフィールドに設定しているため、「`5`」の代わりに「`Int32(rankingNumber)`」を使用して設定してください

ヒント

- ニフティクラウドmobile backendのiOSドキュメントはObjective-Cで書かれていますので、Swiftに書き換えたものを用意しました

    http://qiita.com/natsumo/items/25074fa1ce209033e98e

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Xcode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
    - Swift3.0 の場合

```
let err = error as! NSError
// 検索に失敗した場合の処理
print("検索に失敗しました。エラーコード:\(err.code)")
```

    - Swift2.0 の場合

```
// 検索に失敗した場合の処理
print("検索に失敗しました。エラーコード:\(error.code)")
```

- 保存に成功した場合の処理を行う箇所に追記
    - Swif3.0 の場合

```
// 検索に成功した場合の処理
print("検索に成功しました。")
```

    - Swift2.0 の場合

```
// 検索に成功した場合の処理
print("検索に成功しました。")
```

【作業2-2】

シミュレーターで実行し、「ランキングを見る」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、「ランキングを見る」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

```
// 取得したデータを格納
self.rankingArray = objects as! Array
// テーブルビューをリロード
self.rankingTableView.reloadData()
```

【作業2-4】

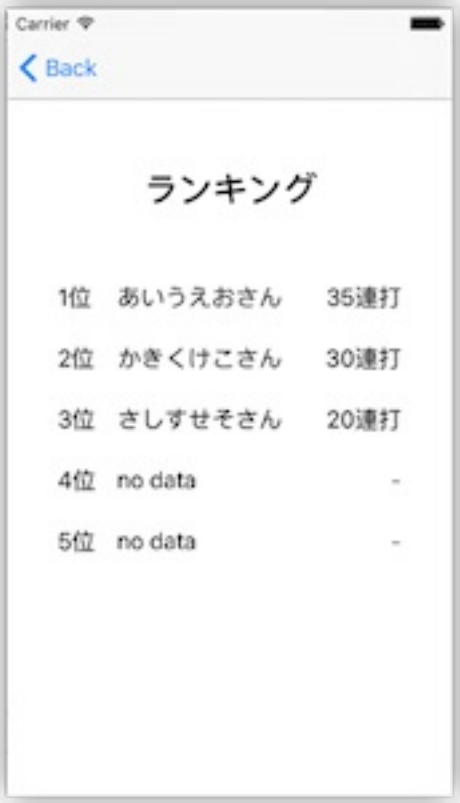
シミュレーターで実行、「ランキングを見る」ボタンを押します

- 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「ランキングを見る」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

模範解答は以下です

Xcode

- Swift3.0 の場合

```
// **********【問題2】ランキングを表示しよう!**********
// GameScoreクラスを検索するクエリを作成
let query = NCMBQuery(className: "GameScore")
// scoreの降順でデータを取得するように設定する
query?.addDescendingOrder("score")
// 検索件数を設定
query?.limit = Int32(rankingNumber)
// データストアを検索
query?.findObjectsInBackground({(objects, error) in
    if error != nil {
        let err = error as! NSError
        // 検索に失敗した場合の処理
        print("検索に失敗しました。エラーコード:\(err.code)")
    } else {
        // 検索に成功した場合の処理
        print("検索に成功しました。")
        // 取得したデータを格納
        self.rankingArray = objects as! Array
        // テーブルビューをリロード
        self.rankingTableView.reloadData()
    }
})
// ***************************************************
```

- Swift2.0 の場合

```
// **********【問題2】ランキングを表示しよう!**********
// GameScoreクラスを検索するクエリを作成
let query = NCMBQuery(className: "GameScore")
// scoreの降順でデータを取得するように設定する
query.addDescendingOrder("score")
// 検索件数を設定
query.limit = Int32(rankingNumber)
// データストアを検索
query.findObjectsInBackgroundWithBlock { (objects: [AnyObject]!, error: NSError!) -> Void in
    if error != nil {
        // 検索に失敗した場合の処理
        print("検索に失敗しました。エラーコード:\(error.code)")
    } else {
        // 検索に成功した場合の処理
        print("検索に成功しました。")
        // 取得したデータを格納
        self.rankingArray = objects as! Array
        // テーブルビューをリロード
        self.rankingTableView.reloadData()
    }
}
// ***************************************************
```

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▼
Swift3.0
https://github.com/natsumo/SwiftFirstApp/archive/Swift3.0_AnswerProject.zip
Swift2.0
https://github.com/natsumo/SwiftFirstApp/archive/Swift2.0_AnswerProject.zip

- APIキーを設定してご利用ください