iOS クイックスタート

このページでは、mobile backendをiOSアプリと連携させる手順を紹介します

目次

アプリの新規作成

mBaaS
ダッシュボード

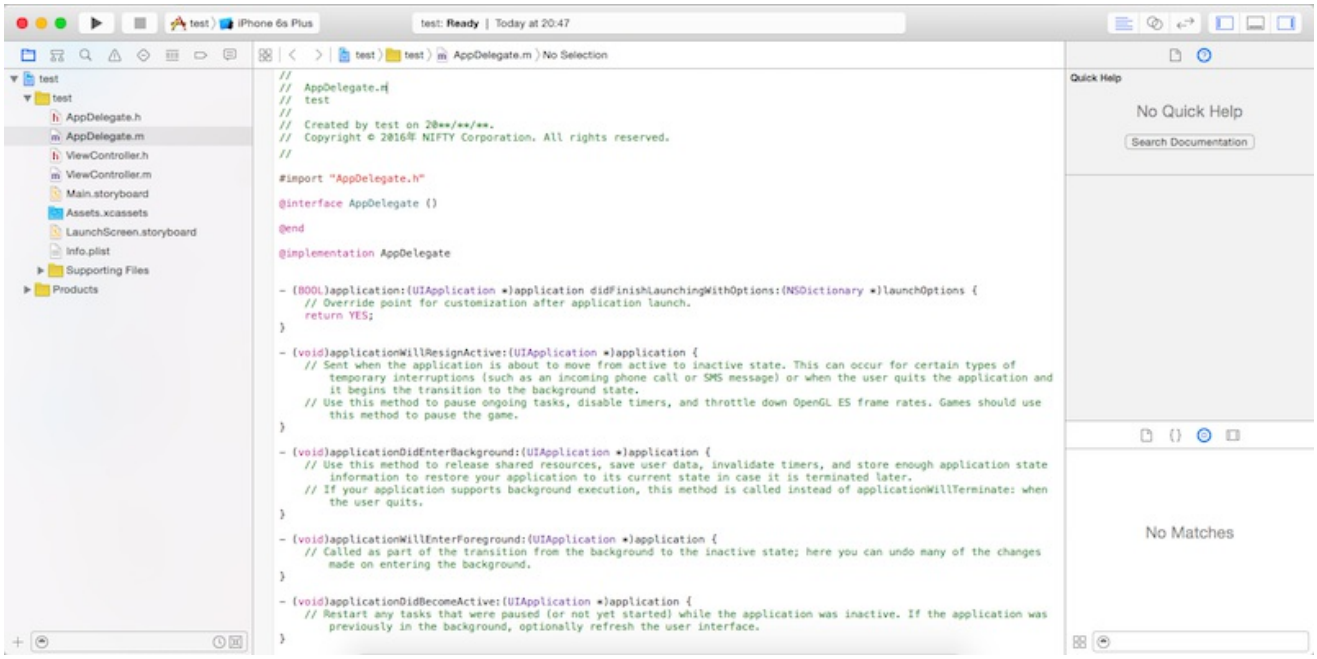
- ニフティクラウドmobile backendにログインします
- ダッシュボードが表示されたら、「アプリの新規作成」を行います
    - すでに別のアプリを作成済みの場合は、ヘッダーの「＋新しいアプリ」をクリックします

- 「アプリ名」を入力し「新規作成」をクリックすると、APIキー（アプリケーションキーとクライアントキー）が発行されます

- APIキーは後ほどXcodeアプリで使います

Xcode

- Xcodeでプロジェクトを作成します
    - 既存のプロジェクトを利用する場合はこの作業は不要です



- プロジェクトは一度閉じておきます

SDKをインストールする

Carthageを利用する方法

ターミナル

- Homebrewなどを利用してCarthageをインストールする必要があります

  i. iOS SDKを利用するプロジェクトのディレクトリへ移動します

  ```
  $ cd 'プロジェクトパス'
  ```

  ii. touchコマンドでCartfileを作成します

  ```
  $ touch Cartfile
  ```

  iii. Cartfileを編集します

  ```
  github "NIFTYCloud-mbaas/ncmb_ios"
  ```

  iv. carthageコマンドを使ってiOS SDKをビルドします

  ```
  $ carthage update --platform iOS
  ```

  - プロジェクト内のCarthage/Build/iOS/ に NCMB.framework が作成されます

  v. プロジェクトに NCMB.Framework を追加します

Xcode

- xcodeでプロジェクトを開きます
- プロジェクト設定のGeneralタブから、「Linked Frameworks and Library」にある「＋」ボタンを押して、「Add Other...」を押して、「Carthage/Build/iOS/NCMB.framework」を選択します
- NCMB.Frameworkが表示されたら、Build Phasesタブに移動し＋ボタンを押して、「New Run Script Phase」を選択します

- 「shell/bin/sh」 と書かれている下のところに以下内容を記述します

```
$ /usr/local/bin/carthage copy-frameworks
```

- その下にある「input Files」の「＋」ボタンを押して、NCMB.frameworkに情報を記述すれば、CarthageでのSDKインストールは完了です

```
$(SRCROOT)/Carthage/Build/iOS/NCMB.framework
```

CocoaPodsを利用する方法

ターミナル

（1）CocoaPodsをインストールする

- CocoaPodsがすでにインストールされている方はこちらの作業は不要です
- cocoaPodsをインストールを行います

```
$ sudo gem install cocoapods
```

- cocoaPodsのセットアップを行います

```
$ pod setup
```

- バージョン情報が表示されればインストール完了です

```
$ pod --version
```

（2）SDKライブラリのインストール

1. Xcodeプロジェクト内にある「プロジェクト名.xcodeproj」と同じディレクトリに移動します

```
$ cd 'プロジェクトパス'
```

2. Podfile(インストールするライブラリを指定するファイル)を作成します

```
$ pod init
```

3. podfileを開いて、以下の内容に書き換えてください

```
# Uncomment this line to define a global platform for your project
platform :ios, '8.0'
# Uncomment this line if you're using Swift
# use_frameworks!

target "YOUR_APP_TARGET" do
    pod 'NCMB', :git => 'https://github.com/NIFTYCloud-mbaas/ncmb_ios.git'
end
```

    - 「`YOUR_APP_TARGET`」の部分は、作成しているXcodeプロジェクトのプロジェクト名に書き換えてください

4. 編集したpodfileを保存をします

5. podfileに書いたSDKをインストールします

```
$ pod install
```

    - 基本は上記コマンドでインストールを行いますが、短時間でインストールが必要な場合は、`$ pod install --no-repo-update`が利用可能です

6. 「プロジェクト名.xcworkspace」が作成されます

    - 注意：元々ある「プロジェクト名.xcodeproj」からXcodeアプリを開いても、SDKが読み込まれませんので、必ず「プロジェクト名.xcworkspace」から開いて編集を行ってください

参考：SDKのアップデートについて

- コマンドSDKのアップデートが可能です

```
$ pod update
```

    - Cocoapodsを利用して導入したSDKの場合は上記コマンドの実行だけで更新可能です。

- ローカルに置いたSDKのリポジトリを指定していた場合は以下の方法で更新できます

```
# Uncomment this line to define a global platform for your project
platform :ios, '8.0'
# Uncomment this line if you're using Swift
# use_frameworks!

target "YOUR_APP_TARGET" do
#  以下のようにローカルのpathを指定していた場合はPodfileを変更する
#  pod 'NCMB', :path =>'your directory path'
#
#  変更後のpod指定方法
   pod 'NCMB', :git => 'https://github.com/NIFTYCloud-mbaas/ncmb_ios.git'
end
```

    - 「`YOUR_APP_TARGET`」の部分は、作成しているXcodeプロジェクトのプロジェクト名に書き換えてください

- 上記のようにpodfileを編集（GitHubリポジトリを指定）して、下記コマンドを実行します

```
$ pod update
```

SDKをダウンロードして利用する方法

（1）SDKをダウンロードする

- GitHubのiOS SDKページで「Clone or download ▼」＞「Download ZIP」をクリックし、masterブランチのzipファイルをダウンロードします
- ダウンロードしたzipファイルを解凍してフォルダを開きます
    - フォルダの中には「NCMB」というフォルダがあります。その中のファイルがSDKです。

（2）SDKをインストールする

Xcode

- Xcodeプロジェクトを開きます
- （1）で確認した「NCMB」フォルダをXcodeプロジェクトのターゲットグループ直下（AppDelegateクラスと同じ階層）にコピーします
- フォルダをコピーするときに、Xcodeでポップアップが開くので、次の様に設定します
    - 「Destination」の項目で「Copy items if needed」にチェックを入れる
    - 「Added folders」の項目で「Create groups」を選択する
    - 「Add to targets」の項目でSDKを利用するターゲットを選択する

参考：SDKのアップデートについて

- 最新のSDKをダウンロードし、同様の操作で「NCMB」フォルダを置き換えることで、SDKのアップデートが可能です

参考：ARCが無効な環境でSDKを利用する場合

- ARCが無効な環境でSDKを利用する場合は、以下の手順でSDKのみARCを有効にする設定を行います
    - ターゲットの一覧から対象のターゲットを選択
    - 「Build Phases」のタブにある「Compile Sources」を開く
    - ニフティクラウド mobile backendのiOS SDKを構成する全ファイルを選択
    - ダブルクリックして「Compiler Flags」に「-fobjc-arc」を設定

SDKの読み込み

Xcode

- `AppDelegate.m`の冒頭に次のコードを追記して、インストールしたSDKを読み込みます

    - 追記するコードは、SDKのインストール方法によって異なります

```
// CocoaPodsを利用する方法
#import <NCMB/NCMB.h>

// SDKをダウンロードして利用する方法
#import "NCMB/NCMB.h"
```

APIキーの設定とSDKの初期化

Xcode

- コードを書いていく前に、必ずmBaaSで発行されたAPIキーの設定とSDKの初期化を行う必要があります
- `AppDelegate.m` の `application:didFinishLaunchingWithOptions:` メソッドに次のコードを書きます

```
// APIキーの設定とSDK初期化
[NCMB setApplicationKey:@"YOUR_APPLICATION_KEY" clientKey:@"YOUR_CLIENT_KEY"];
```

mBaaS
ダッシュボード

- 上の「`YOUR_APPLICATION_KEY`」と「`YOUR_CLIENT_KEY`」は、mBaaSのダッシュボードで「アプリの新規作成」を行ったときに発行されたAPIキーに置き換えます
    - アプリ作成時のAPIキー発行画面を閉じてしまった場合は、「アプリ設定」＞「基本」で確認できます。
    - 「コピー」ボタンを使用してコピーしてください。

- これで連携作業は完了です！サンプルコードを書いて実際にmBaaSを使ってみましょう

サンプルコードの実装

Xcode

- `AppDelegate.m` の `application:didFinishLaunchingWithOptions:` メソッド内に書いた処理は、アプリの起動時に実行されます
- APIキーの設定とSDK初期化コードの下にサンプルコードを書くと、すぐに動作確認が可能です

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
// APIキーの設定とSDK初期化
[NCMB setApplicationKey:@"YOUR_APPLICATION_KEY" clientKey:@"YOUR_CLIENT_KEY"];
// ↓ここにサンプルコードを実装↓


return YES;
}
```

サンプルコード（データストア）

- 次のコードはmBaaSのデータストアに保存先の「TestClass」というクラスを作成し、「message」というフィールドへ「Hello, NCMB!」というメッセージ（文字列）を保存するものです

```
// クラスのNCMBObjectを作成
NCMBObject *object = [NCMBObject objectWithClassName:@"TestClass"];
// オブジェクトに値を設定
[object setObject:@"Hello, NCMB!" forKey:@"message"];
// データストアへの登録
[object saveInBackgroundWithBlock:^(NSError *error) {
    if (error){
        // 保存に失敗した場合の処理

    } else {
        // 保存に成功した場合の処理

    }
}];
```

アプリを実行してmBaaSのダッシュボードを確認する

- アプリを実機またはシュミレーターで実行します

mBaaS
ダッシュボード

- アプリが起動されたら、mBaaSのダッシュボードで「データストア」から、データが保存されていることを確認できます