クイックスタートについて

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

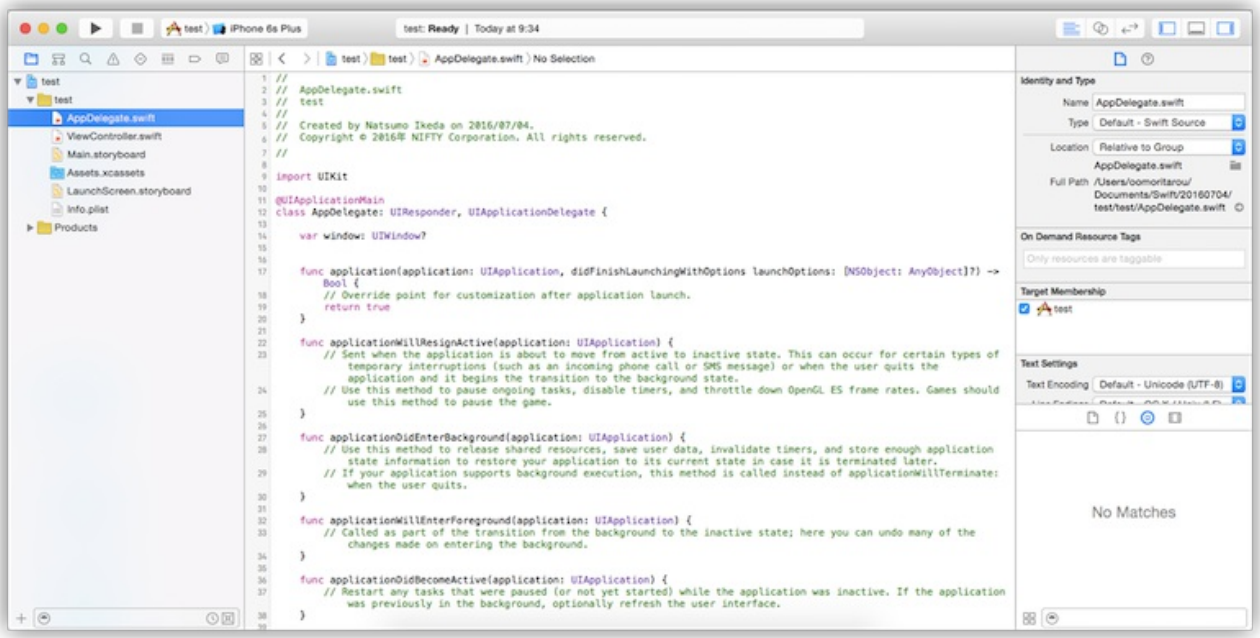
- ニフティクラウドmobile backendにログインします
- ダッシュボードが表示されたら、「アプリの新規作成」を行います
    - すでに別のアプリを作成済みの場合は、ヘッダーの「＋新しいアプリ」をクリックします

- 「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

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

- 「`$ sudo gem install cocoapods`」コマンドでcocoaPodsをインストールを行います

- 「`$ pod setup`」コマンドでセットアップを行います
- 「`$ pod --version`」コマンドでバージョン情報が表示されればインストール完了です

（2）SDKライブラリのインストール

1. 「`$ cd`」コマンドで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
2. 「`$ pod init`」コマンドで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

1. podfileを開いて、下記(※1)に記載しているの内容に書き換えてください

    - **Xcode7以上の場合**（`use_frameworks!`を使用する方法）

```
# 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
```

    - **Xcode7未満または上記方法が利用できない場合**

```
# 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
```

    ※いずれの場合も「`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

2. 編集したpodfileを保存をします
3. 「`$ pod install --no-repo-update`」コマンドでpodfileに書いたSDKをインストールします
4. 「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

- 「`$ pod update`」コマンドでSDKのアップデートが可能です
  - 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

  - `use_framework!`が使用できない環境の場合はコメントアウトしてください

```
# 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
#  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、「`$ pod update`」コマンドを実行します

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

CocoaPodsを利用する方法

Xcode7以上の場合（`use_frameworks!`を使用する方法）

- `AppDelegate.swift`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
    - 他のファイルでもSDKを使用する場合は都度追記が必要です

```
import NCMB
```

Xcode7未満または上記方法が利用できない場合

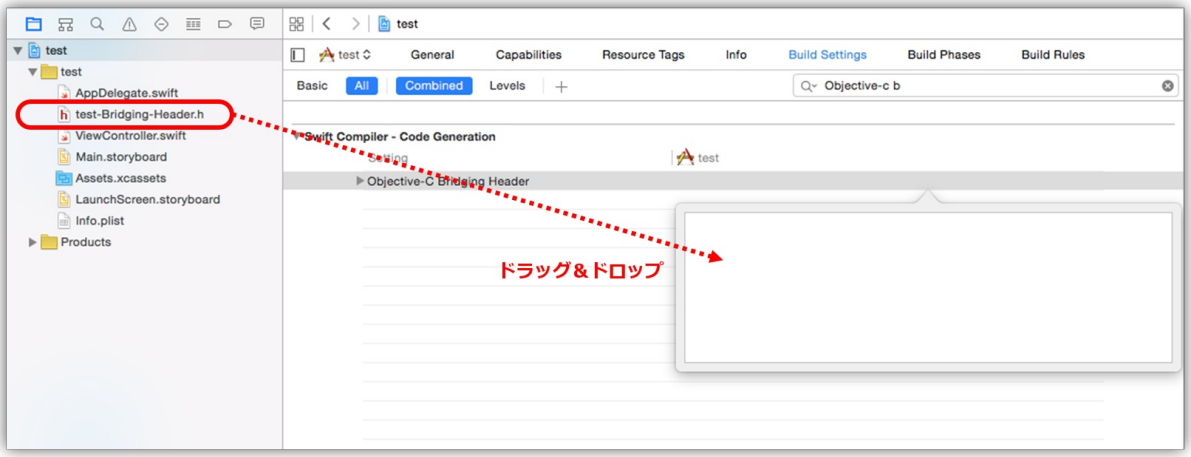
- `AppDelegate.swift`と同じディレクトリに、次の手順でヘッダーファイルを作成します
    i. `AppDelegate.swift`上で右クリック＞「New File...」＞「Header File」＞「Next」をクリックするします
    ii. 「`Save As:`」の欄に「`XXXXXXX-Bridging-Header`」と記入（「`XXXXXXX`」のところは任意ですが、プロジェクト名にするのが一般的です）し、「Create」をクリックします
- 作成したファイルの中に下記の内容を追記します

```
#import <NCMB/NCMB.h>
```

- 作成した`XXXXXXX-Bridging-Header.h`ファイルを次の手順で読み込みます

    i. プロジェクト＞「Build Settings」をクリックします
    ii. 「Objective-C Bridging Header」（右上の検索を使うとすぐ見つけられます）をダブルクリックすると入力用のふきだしが出てきます
    iii. そこに先ほど作成した「`XXXXXXX-Bridging-Header.h`」を下図のようにドラッグ＆ドロップします

    iv. ふきだしが閉じ、「`XXXXXXX-Bridging-Header.h`」のディレクトリが入力されたことが確認できれば読み込み完了です



SDKをダウンロードして利用する方法

- 上記「Xcode7未満または上記方法が利用できない場合」と同様の方法でヘッダーファイルを作成し、SDKの読み込みを行ってください

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.swift` の `didFinishLaunchingWithOptions` メソッド内に次のコードを書きます
    - Swift3.0 の場合

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    // APIキーの設定とSDKの初期化
    NCMB.setApplicationKey("YOUR_NCMB_APPLICATION_KEY", clientKey: "YOUR_NCMB_CLIENT_KEY")
    // ↓ここにサンプルコードを実装↓


    return true
}
```

    - Swift2.0 の場合

```
// 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.swift` の `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
- Swift3.0 の場合

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    // APIキーの設定とSDKの初期化
    NCMB.setApplicationKey("YOUR_NCMB_APPLICATION_KEY", clientKey: "YOUR_NCMB_CLIENT_KEY")
    // ↓ここにサンプルコードを実装↓


    return true
}
```

- Swift2.0 の場合

```
func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {
    // 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 true
}
```

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

    - Swift3.0 の場合

```
// クラスのNCMBObjectを作成
let obj = NCMBObject(className: "TestClass")
// オブジェクトに値を設定
obj?.setObject("Hello, NCMB!", forKey: "message")
// データストアへの登録
obj?.saveInBackground({ (error) in
    if error != nil {
        // 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
    }
})
```

    - Swift2.0 の場合

```
// クラスのNCMBObjectを作成
let obj = NCMBObject(className: "TestClass")
// オブジェクトに値を設定
obj.setObject("Hello, NCMB!", forKey: "message")
// データストアへの登録
obj.saveInBackgroundWithBlock { (error: NSError!) -> Void in
    if error != nil {
        // 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