

iOS Quick Start demonstration

Overview

The purpose of this demonstration is to experience an end-to-end flow where the MobileFirst Platform Foundation SDK for iOS is integrated into a Xcode project and used to retrieve data using a MobileFirst adapter.

To learn more about creating projects and applications, using adapters and lots more, visit the Native iOS Development ([../ios-tutorials/](#)) landing page.

Required installed:

- MobileFirst Platform commandline tool (download
(file:///home/travis/build/MFPSamples/DevCenter/_site/downloads/))
 - Xcode 6.x
-

1. Create a MobileFirst project and adapter

- Create a new project and iOS framework/server-side application entity

```
mfp create MyProject
cd MyProject
mfp add api MyiOSFramework -e ios
```

- Add a HTTP adapter to the project

```
mfp add adapter MyAdapter -t http
```

2. Deploy artifacts to the MobileFirst Server

- Start the MobileFirst Server and deploy the server-side application entity and adapter

```
mfp start
# Wait until a browser window is opened, displaying the MobileFirst C
onsole
mfp build
mfp deploy
```

3. Create a Xcode project

4. Add the MobileFirst iOS SDK to the Xcode project

- In **Project explorer** right-click and select **Add Files to your-iOS-app-name...**
 - Navigate to **project-folder-location > MyProject > apps > MyiOSFramework** and select **worklight.plist** file and the **WorklightAPI** folder

- In **Build Phases** open **Link Binary With Libraries** and add:
 - libWorklightStaticLibProjectNative.a (found in **WorklightAPI**)
 - sqlcipher.framework (found in **WorklightAPI/Frameworks**)
 - SystemConfiguration.framework
 - MobileCoreServices.framework
 - CoreLocation.framework
 - Security.framework
 - libstdc++.6.dylib
 - libc++.dylib
 - libz.dylib
- In **Build Settings** search for:
 - Header Search Path: add \$(SRCROOT)/WorklightAPI/include
 - Other Linker Flags: add -ObjC

5. Implement MobileFirst adapter invocation

- **AppDelegate.h** Add the header:

```
#import "WLResourceRequest.h"
```

- **AppDelegate.m** Add the header:

```
#import "WLResponse.h"
```

Add the following to didFinishLaunchingWithOptions:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    NSURL* url = [NSURL URLWithString:@"adapters/MyAdapter/getFeed"];
    WLResourceRequest* request = [WLResourceRequest requestWithURL:url method:WLHttpMethodGet];
    [request setQueryParameterValue:@"[technology]" forName:@"params"];

    [request sendWithCompletionHandler:^(WLResponse *response, NSError *error) {
        if(error != nil){
            NSLog(@"%@",error.description);
        }
        else{
            NSLog(@"%@",response.responseJSON);
        }
    }];

    return YES;
}
```

6. Final configurations

- ## 7. Click Run

The screenshot displays the Xcode IDE with the AppDelegate.m file open. The code defines the AppDelegate class and implements the didFinishLaunchingWithOptions method. In this method, a GET request is sent to the URL "http://www.cnn.com/". The response is handled by a completion handler that logs the response and the response body. The console output shows the successful response from the API, including headers and JSON data.

```

1 //
2 // Targeted to the iPhone 6.
3 //
4 // Created by Idan Adar on 25/03/15.
5 // Copyright (c) 2015 IBM. All rights reserved.
6
7
8 #import "AppDelegate.h"
9 #import "MyResponse.h"
10
11 @interface AppDelegate ()
12
13 @end
14
15 @implementation AppDelegate
16
17
18 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
19     NSURL* url = [NSURL URLWithString:@"http://www.cnn.com/"];
20     NSMutableURLRequest* request = [NSMutableURLRequest requestWithURL:url method:GET timeoutInterval:60.0];
21     [request setQueryParameterValues:@{@"technology": @"flame",@"params": ""}];
22
23     [request sendWithCompletionHandler:^(NSData* response, NSError* error) {
24         if (error != nil) {
25             NSLog(@"Error: %@", error.description);
26         } else {
27             NSLog(@"Success: %@", response);
28             NSLog(@"Response Body: %@", [NSString stringWithData:response encoding:NSUTF8StringEncoding]);
29         }
30     }];
31
32     return YES;
33 }
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

The console output shows the successful response from the API, including headers and JSON data:

```

2015-03-25 11:07:02.078 MyApp[53372:602219] {
  errors = (
  );
  info = (
  );
  isSuccessful = 1;
  isSuccessful = 1;
  responseHeaders = {
    "Accept-Ranges" = none;
    "Alternate-Protocol" = "88;quic,p=0.5,88;quic,p=0.5";
    "Cache-Control" = "private, max-age=0";
    "Content-Type" = "text/html; charset=UTF-8";
    Date = "Wed, 25 Mar 2015 09:06:57 GMT";
    Expires = "Wed, 25 Mar 2015 09:06:57 GMT";
    "Last-Modified" = "Wed, 25 Mar 2015 09:06:57 GMT";
    Server = GSE;
    "Transfer-Encoding" = chunked;
    Vary = "Accept-Encoding";
    "X-Content-Type-Options" = nosniff;
    "X-XSS-Protection" = "1; mode=block"
  };
  responseTime = 100;
  rss = {
    channel = {
      copyright = "Copyright 2015 Cable News Network LP, LLLP.";
      description = "CNN.com delivers up-to-the-minute news and information on the latest top stories, weather, entertainment, politics and more.";
      image = {
        description = "CNN.com delivers up-to-the-minute news and information on the latest top stories,

```