

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 shows the Xcode IDE with the AppDelegate.m file open. The code defines an AppDelegate class that implements the AppDelegate protocol. It includes a method didFinishLaunchingWithOptions that sends an HTTP GET request to a URL. The console output shows the response of the request, which is a JSON object containing information about the application and the device.

```

1 //
2 // MyApp
3 //
4 // Created by Idan Adar on 25/03/15.
5 // Copyright (c) 2015 IBM. All rights reserved.
6
7
8
9 #import "AppDelegate.h"
10 #import "MyResponse.h"
11
12 @interface AppDelegate ()
13
14 @end
15
16 @implementation AppDelegate
17
18
19 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
20     NSURL* url = [NSURL URLWithString:@"http://myapp.com/getData"];
21     NSMutableURLRequest * request = [NSMutableURLRequest requestWithURL:url method:GET timeoutInterval:60.0];
22     [request setQueryParameterValues:@{@"technology": @"ios"}];
23
24     [request sendWithCompletionHandler:^(NSURLResponse *response, NSError *error) {
25         if(error != nil){
26             NSLog(@"%s", error.description);
27         }
28         else {
29             NSLog(@"%s", response.description);
30         }
31     }];
32
33     return YES;
34 }
35
36

```

The console output shows the response of the request, which is a JSON object containing information about the application and the device.

```

2015-03-25 11:07:02.078 MyApp[53372:602219] {
  errors = (
  );
  info = (
  );
  isSuccessful = 1;
  responseHeaders = {
    "Accept-Ranges" = none;
    "Alternate-Protocol" = "80:quic,p=0.5,80: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,

```