

# 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
```

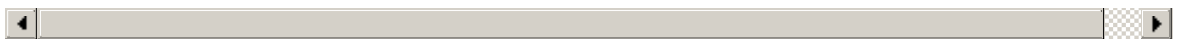
- Add a HTTP adapter to the project

```
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 Conso
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:

```
1 | #import "WLResourceRequest.h"
```

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

```
1 | #import "WLResponse.h"
```

Add the following to `didFinishLaunchingWithOptions`:

```
1 | - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary*)launchOptions {
2 |     NSURL* url = [NSURL URLWithString:@"./adapters/MyAdapter/getFeed"];
3 |     WLResourceRequest* request = [WLResourceRequest requestWithURL:url method:WLRequestMethodGET];
4 |     [request setQueryParamValue:@"[\"technology\"]" forName:@"params"];
5 |
6 |     [request sendWithCompletionHandler:^(WLResponse *response, NSError *error) {
7 |         if(error != nil){
8 |             NSLog(@"%@",error.description);
9 |         }
10 |         else{
11 |             NSLog(@"%@",response.responseJSON);
12 |         }
13 |     }];
14 |
15 |     return YES;
16 | }
```

## 6. Final configurations

- ## 7. Click Run

The screenshot displays the Xcode IDE interface. On the left, the Project Navigator shows the project structure with files like AppDelegate.m, ViewController.h, and Main.storyboard. The central area shows the AppDelegate.m file with the following code:

```

1 //
2 // Created by Idan Adar on 25/03/15.
3 // Copyright (c) 2015 IBM. All rights reserved.
4
5 #import "AppDelegate.h"
6 #import "MLResponse.h"
7
8 @interface AppDelegate ()
9
10 @end
11
12 @implementation AppDelegate
13
14 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
15     NSURL* url = [NSURL URLWithString:@"adapters/myAdapter/getStories"];
16     NSURLRequest* request = [NSURLRequest requestWithURL:url method:MLRequestMethodGET];
17     [request setQueryParameterValues:@{@"technology":@"forName":@"params"}];
18
19     [request sendWithCompletionHandler:^(MLResponse *response, NSError *error) {
20         if(error != nil){
21             NSLog(@"%@" ,error.description);
22         }
23         else {
24             NSLog(@"%@" ,response.responseJSON);
25         }
26     }];
27
28     return YES;
29 }

```

On the right, the Console window shows the network log for the GET request to the news service. The log includes the request URL, headers, status, and response body.

```

2015-03-25 09:05:39 GMT "Alternate-Protocol": "80,quic,p=0.5", "Content-Type": "text/html; charset=UTF-8", "Accept-Ranges": "none", "Server": "GSE", "Cache-Control": "private, max-age=0", "X-Content-Type-Options": "nosniff", "warnings": [{"totalSize": 151, "responseTime": 100, "info": []}]
2015-03-25 11:07:02: 078 MyApp[53372:862219] {
  errors = (
  );
  info = (
  );
  isSuccessfull = 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 09:05:39 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,

```

The right sidebar shows the Quick Help panel with a search bar and a list of controllers: View Controller, Navigation Controller, and Table View Controller, each with a brief description of their function.