

iOS end-to-end demonstration

fork and edit tutorial (<https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/8.0/quick-start/ios/index.md>) | report issue (<https://github.ibm.com/MFPSamples/DevCenter/issues/new>)

Overview

The purpose of this demonstration is to experience an end-to-end flow where an application and an adapter are registered using the MobileFirst Operations Console, an "skeleton" Xcode project is downloaded and edited to call the adapter, and the result is printed to the log - verifying a successful connection with the MobileFirst Server.

Prerequisites:

- Xcode
- MobileFirst Developer CLI (download (file:///home/travis/build/MFPSamples/DevCenter/_site/downloads))
- *Optional.* Stand-alone MobileFirst Server (download (file:///home/travis/build/MFPSamples/DevCenter/_site/downloads))

1. Starting the MobileFirst Server

If a remote server was already set-up, skip this step.

From a **Command-line** window, navigate to the server's **scripts** folder and run the command: `./start.sh`.

2. Creating an application

In a browser window, open the MobileFirst Operations Console by loading the URL: `http://your-server-host:server-port/mfpconsole`. If running locally, use: `http://localhost:9080/mfpconsole` (`http://localhost:9080/mfpconsole`). The username/password are `admin/admin`.

1. Click on the "New" button next to **Applications** and select the desired *platform*, *identifier* and *version* values.



2. Click on the **Get Starter Code** tile and select to download the iOS Starter Code.





3. Editing application logic

1. Open the Xcode project project by double-clicking the **.xcworkspace** file.
2. Select the **[project-root]/ViewController.m/swift** file and paste the following code snippet, replacing the existing `viewDidLoad()` function:

In Objective-C:

```
- (void)viewDidLoad {
    [super viewDidLoad];

    NSURL* url = [NSURL URLWithString:@"./adapters/javaAdapter/users/world"];
    WLResourceRequest* request = [WLResourceRequest requestWithURL:url method:WLHttpMethodGet];

    [request sendWithCompletionHandler:^(WLResponse *response, NSError *error) {
        if (error != nil){
            NSLog(@"Failure: %@",error.description);
        }
        else if (response != nil){
            // Will print "Hello world" in the Xcode Console.
            NSLog(@"Success: %@",response.responseText);
        }
    }];
}
```

In Swift:

```
override func viewDidLoad() {
    super.viewDidLoad()

    let url = NSURL(string: "/adapters/javaAdapter/users/world")
    let request = WLResourceRequest(URL: url, method: WLHttpMethodGet)

    request.sendWithCompletionHandler { (WLResponse response, NSError error) -> Void in
        if (error != nil){
            NSLog("Failure: " + error.description)
        }
        else if (response != nil){
            NSLog("Success: " + response.responseText)
        }
    }
}
```

4. Creating an adapter

1. Click on the "New" button next to **Adapters** and download the **Java** adapter sample.

If Maven and MobileFirst CLI are not installed, follow the on-screen **Setting up your environment** instructions to install. Alternatively, download this prepared .adapter artifact and deploy it from the MobileFirst Operations Console using the **Actions → Deploy adapter** action.

MobileFirst Operations Console

Analytics Console

Hello, demo

Dashboard

Runtimes

Runtime

Settings

Applications

MyBankApp

Adapters

No adapter deployed

Devices

Client Logs

Error Log

Create new

Create new

Home > runtime > Get Starter Code

Get Starter Code

ApplicationAdapter

Java Adapter Samples

Select the sample that you would like to download

JavaJava-security check

Javascript Adapter Samples

Select the sample that you would like to download

Javascript-SQLJavascript-SAPJavascript-JMSJavascript-HTTP

Next Step

Go to Adapter Quickstart

- mfpdev adapter build

- ## 5. Testing the application

- The adapter response is then printed in the Xcode Console.

```
Date = "Tue, 19 Jan 2016 06:14:40 GMT";  
"Transfer-Encoding" = Identity;  
X-Powered-By = "Servlet/3.1";  
}  
  
Response Data:  
{"access_token": "eyJhbGciOiJIUzI1NiIsImNpdjkiZmVlJjoiaWQVFBIiIsImRhdGVudDQiOiJBTTEBZDRwQ2RNkgteHM3dCIxLUNMZEUiOmkya2o8NXpnbnREZF9sczhmdmZZcmRvcPrtYjRfMnQ7ODHOENWNUNLNDFQTkXBzdjE2MDMEwdWIJmS24HvhwWFLGY01TYU9LSXFVZSl5ySkEdwpIdz  
JySh5WjNNXVMtSL2V6ULzj089C3zIWJSrZ8tNmIXKzLMWFZWdzrhUB9Q3JKSMVuocU3VKR3T2LztJKTdsUdeWMVCyIBaZmtOkwkstFUB0b1PaSUckMSXa01THZ2MdLOTDVB63MVTKExNXZL08twJDxCGLTGJTJTNjFuGHIn2MRW95BUURVeVqFKlQW9omMsUS02NZJHW  
Z2VLTVKzJoLVIS9wB7LYt6e-gkdouDUcZHXNHu3gc3RLduWDJOJVTVRCE=TfOhNMVgldGoe4FRKHJUMU4IKjs91jOUHLIawjaLz01kyYZ2jnZDMUKct1OTLIghBMZdT7KSzOHMTKEtGRKLNGMB0T3zm19_eJpc3mljljzb2uHm4tlmcINcsINmYLGI6gm  
Z2NVTLVTKsyjYNDAZY2R5OWMLK16TR6jRjPzL2ZCGic6mw5SympubUwZ3KjgwYrYALCIZ2ybSwCSilI9_rF64dYAhuysdtF_6CTyWX3P7WP_pswdWLkmzlujY_BedixcdITJPntmsBmEsH3KgLBSTFHMFu93bv_wqb9XodyMpaIX2  
ITrcxyOVnoU6clIP3cpQvl5j1eqseLUKJRirFnrfPalIpb720H_z8HB4XP264ikbyKDsgslQ6t1d52cgLpC2psnm_o-w3pGCVkuPK53Q0zy1AbbkFTNS_6VEAMER4Y8rJDBm1FCVS2gb3dbbzEB4Z2ILdWkXuTyYnGDJKagMKkJjyl3el12NDzc_w  
_dy4kpjgr7mJLrs6zxHG84gy8BMWG2Ba7jFNpLGdgg3","token_type": "Bearer","expires_in":3599,"scope":""})  
  
Status code=200  
2016-01-19 08:14:40.418 MyAppApplication[93738:36590517] +[OCLogger printMessageWithMetadata:andLevelTag:andPackage:] [Line 1005] [DEBUG] [WL_AFHTTPSessionManagerWrapper_PACKAGE] -[WLAFHTTPSessionManagerWrapper start]  
in WLAFHTTPSessionManagerWrapper.m:372 : Starting the request with URL http://1980/mfp/api/adapters/javaAdapter/users/world  
2016-01-19 08:14:40.444 MyAppApplication[93738:36590517] +[OCLogger printMessageWithMetadata:andLevelTag:andPackage:] [Line 1005] [DEBUG] [WL_AFHTTPSessionManagerWrapper_PACKAGE] -[WLAFHTTPSessionManagerWrapper  
requestFinished:responseObject:] in WLAFHTTPSessionManagerWrapper.m:388 : Request Success  
2016-01-19 08:14:40.444 MyAppApplication[93738:36590517] +[OCLogger printMessageWithMetadata:andLevelTag:andPackage:] [Line 1005] [DEBUG] [WL_AFHTTPSessionManagerWrapper_PACKAGE] -[WLAFHTTPSessionManagerWrapper  
requestFinished:responseObject:] in WLAFHTTPSessionManagerWrapper.m:381 : Response Status Code : 200  
2016-01-19 08:14:40.444 MyAppApplication[93738:36590517] -[WLAFHTTPSessionManagerWrapper requestFinished:responseObject:] [Line 393] Response Content : Hello world  
2016-01-19 08:14:40.441 MyAppApplication[93738:36590517] Adapter invocation response: Hello world
```

All Output ↕

Note: Xcode 7 enables Application Transport Security (ATS)

(https://developer.apple.com/library/ios/releasenotes/General/WhatsNewIniOS/Articles/iOS9.html#//apple_ref/doc/uid/TP40016198-SW14) by default.

To complete the tutorial, disable ATS (<http://iosdevtips.co/post/121756573323/ios-9-xcode-7-http-connect-server-error>).

1. In Xcode, right-click the **[project]/info.plist** file → **Open As** → **Source Code**
2. Paste the following:

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSAllowsArbitraryLoads</key>
  <true/>
</dict>
```

3. Press the **Play** button.

Next steps

Learn more on using adapters in applications, and how to integrate additional services such as Push Notifications, using the MobileFirst security framework and more:

- Review the Server-side development tutorials ([../adapters/](#))
- Review the Authentication and security tutorials ([../authentication-and-security/](#))
- Review the Notifications tutorials ([../notifications/](#))
- Review All Tutorials ([../all-tutorials](#))