

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 folder and run the command: `./run.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.

The screenshot shows the MobileFirst Operations Console interface. The left sidebar contains a navigation menu with 'Dashboard', 'Runtimes', 'mfp', 'Applications' (with a 'New' button), 'Adapters' (with a 'New' button), 'Settings', 'Devices', and 'Error Log'. The main content area is titled 'Register an Application' and includes the following fields: 'Application Name' (text input), 'Optional display name of the Application' (text input), 'Choose Platform' (radio buttons for Android, iOS (selected), and Windows), 'Bundle ID' (text input, labeled 'Application Identifier; case sensitive'), and 'Version' (text input, labeled 'Application Version'). A 'Register application' button is at the bottom. The top right of the console shows 'Analytics Console', 'Hello, admin', and an information icon.

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**
 - Select the **Actions** → **Download sample** option. Download the **Java** adapter sample.

- From a **Command-line** window, navigate to the adapter's Maven project root folder and run the command:

- When the build finishes, deploy it from the MobileFirst Operations Console using the **Actions** → **Deploy adapter** action. The adapter can be found in the **[adapter]/target** folder.
- Alternatively, download this prepared .adapter artifact and deploy it from the MobileFirst Operations Console using the **Actions** → **Deploy adapter** action.

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.JU";
```

} Response Data:

```
"access.token": "eyJhbGciOiJSUzI1NiIsImtpZSI6eyIxYm9jaWV8QVFBOjQiInM4OjEJBTBTEZDkxOTRlZWdWMN3IC4UNNZEUtM0kyazS0NXpnNmREZF9xczhmdnSZ2NMpcRVrtvJRfMnQT0dHOENWNUNLNDFOqTjdXB3J2dlMDNEwdWIJwms5ZHhhvwVLGY0ITyU9ISXFvZSl5SkEdwpIdz  
JysGHwVINXKVsIZ2VEGLuzJO9ZCziFOLWLRSztBiZnoIXznLVNWFMFzdIrHUB93QkJJsMuovCU3VKR3TZ7LLZJKUtDsMEVCYCbaZmtOkktSFuoBBopasLUckSMelJAoElTHZMDLOTDVBg3NVTKCEHXZZLQ8twadJCxCglTBTTJTnFuRGhtINZdmRW9YSBUrUEVFKIQ9wmLuSSoSZNJHW  
YZVMVVQCZ03JVUVLS9nj2TVybEKsekdOpUCczzCHUHdzKSLRHWDVOZmjVRNKEDrbThQuMNduwiiaIwaRs3ojIUHEItswiazkljiyoJZY2JMzouMKwtIOtlLnghBMBDZLTksZQUWTEnGRNLGNubGUOC3in19_eyP3CcMI0lj1bzBuwhZbmImncInslnNIyl16tmhm  
STeMaKtYRJdXMN5lySiFnfZIGistmhvSpymHubzcZiwIezXiowioQNDUzmZShjqepYLALCIzY29czSIEti39_-P56Ad4UhuyQ5FY_6CY7KHXP7NP_psgMLVmzkuiUY_BxeidccdtIKphrmBNhdILqKLBSBFHMufdyob_vqb9xoicQypmaPKXZ  
ITrcxyVYNNO3QWVnljW1fy1eqsekLMpDKurFrPaLIpb7z0H_r4BXMSZPE4BKqdYKOdkSGMLD6T1QGS2CijgmLCpZOsnQzg_P3gvPCUKM5FSX3OGZoFyzabbtkFN5_6VEAMEAYR8-JDBmlVcfTSZgb3bbdBZE4ZIULW6KiuxYTynGDHGJKQKKXkj1vm3Bl1DNdc_z_  
_4ykUpjr7mXLrsExZhG84gyB8MWGBa7TFnpLKddgz","token.type":"Bearer","expires_in":3599,"scope":""}
```

Status code=200
2016-01-19 08:14:40 .418 MyApplication[93738:36590517] [OCLogger printMessage:withMetadata:andLevelTag;andPackage:] [Line 1005] [DEBUG] [WL_AFHTTPSessionManagerWrapper_PACKAGE] ~-[WLAFHTTPSessionManagerWrapper start]
in WLAFHTTPSessionManagerWrapper.m:372 : Starting the request with URL http://:9808/mfp/api/adapters/javaAdapter/users/world
2016-01-19 08:14:40 .440 MyApplication[93738:36590517] [OCLogger printMessage:withMetadata:andLevelTag;andPackage:] [Line 1005] [DEBUG] [WL_AFHTTPSessionManagerWrapper_PACKAGE] ~-[WLAFHTTPSessionManagerWrapper
requestFinished:responseObject:] in WLAFHTTPSessionManagerWrapper.m:388 : Request Success
2016-01-19 08:14:40 .448 MyApplication[93738:36590517] [OCLogger printMessage:withMetadata:andLevelTag;andPackage:] [Line 1005] [DEBUG] [WL_AFHTTPSessionManagerWrapper_PACKAGE] ~-[WLAFHTTPSessionManagerWrapper
startResponseReceivedData:MultipartStreamedRequestWrapper:m:responseStatus Code: 200
2016-01-19 08:14:40 .448 MyApplication[93738:36590517] ~-[WLAFHTTPSessionManagerWrapper requestFinished:responseObject:] [Line 393] Response Content : Hello world
2016-01-19 08:14:40 .441 MyApplication[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>
```

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](#))