Creating your first native iOS MobileFirst application

Overview

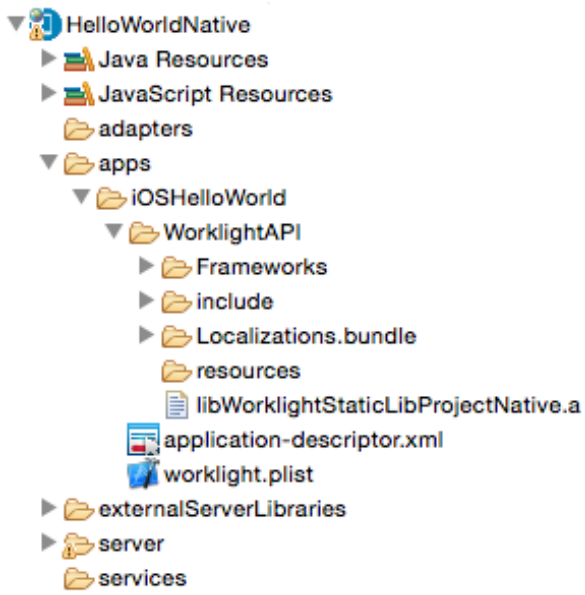
To serve a native iOS application, MobileFirst Server must be aware of it. For this purpose, IBM MobileFirst Platform Foundation provides a Native API library, which contains a set of APIs and configuration files.

This tutorial explains how to generate the iOS Native API and how to integrate it with a native iOS application. You must generate the iOS Native API so that you can use it later on for tasks such as connecting to the MobileFirst Server instance, invoking adapter procedures, implementing authentication methods, and so on.

Prerequisite: Developers must be proficient with the Apple Xcode IDE.

Creating a MobileFirst native API

1. In MobileFirst Studio, create a MobileFirst project and add a MobileFirst Native API.
2. In the **New MobileFirst Native API** dialog, enter your application name and select **iOS** for the **Environment** field.
3. Right-click the generated NativeAPI folder (located in `your-projects/apps/your-nativeapi-app-name`) and select **Run As > Deploy Native API**.
   This action is required in order for MobileFirst Server to recognize the application if it attempts to connect.



The MobileFirst native API contains several components:

- The WorklightAPI folder is an IBM MobileFirst API library that you must copy to your native iOS project.
- You use the **application-descriptor.xml** file to define application metadata and to configure security settings that MobileFirst Server enforces.
- The **worklight.plist** file contains connectivity settings that a native iOS application uses. You must copy this file to your native iOS project.
- As with any MobileFirst project, you create the server configuration by modifying the files that are in the **server\conf** folder.

worklight.plist

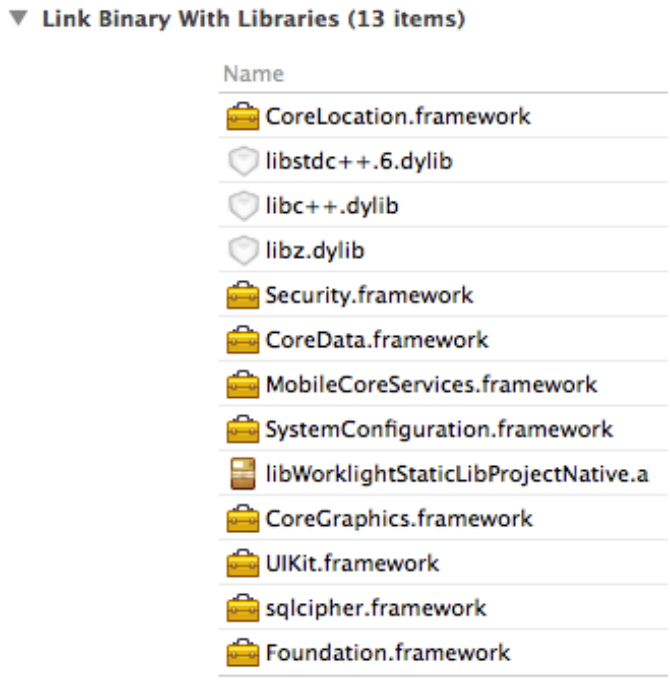
The **worklight.plist** holds server configuration properties and is user-editable:

- `protocol` – The communication protocol to MobileFirst Server, which is either http or https.
- `host` – The hostname of the MobileFirst Server instance.
- `port` – The port of the MobileFirst Server instance.
- `wlServerContext` – The context root path of the application on the MobileFirst Server instance.
- `application id` – The application ID as defined in the **application-descriptor.xml** file.

- `application version` – The application version.
- `environment` – The target environment of the native application (“iOSnative”).
- `wlUid` – This property is used by MTW (Mobile Test Workbench plug-in for Eclipse (../../advanced-topics/testing-mobilefirst-mobile-applications-mobile-test-workbench/)) to identify this as a MobileFirst application.
- `wlPlatformVersion` – The MobileFirst Studio version.

Creating and configuring an iOS native application

<

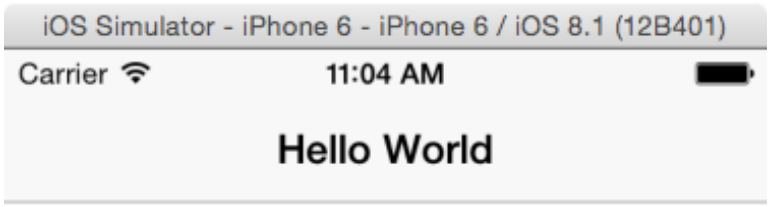


1. Create an Xcode project or use an existing one.
2. Copy the previously-created **WorklightAPI** folder and the **worklight.plist** file from Eclipse to the root of the native project in Xcode.
3. In the Xcode Build Phases section, link the following libraries in your native iOS application:
    - SystemConfiguration.framework
    - sqlcipher.framework (found in **WorklightAPI/Frameworks**)
    - libWorklightStaticLibProjectNative.a (found in **WorklightAPI**)
    - MobileCoreServices.framework
    - CoreData.framework
    - CoreLocation.framework
    - Security.framework
    - libz.dylib
    - libc++.dylib
    - libstdc++.6.dylib
4. In Xcode **Build Settings**:
    - In Header Search Paths, add **$(SRCROOT)/WorklightAPI/include**

- In **Other Linker Flags**, add `-ObjC`
- In the **Deployment** section, for the iOS **Deployment Target** field, select a value that is greater than or equal to 6.0.

> For more information, review the "Developing native applications for iOS" user documentation topic

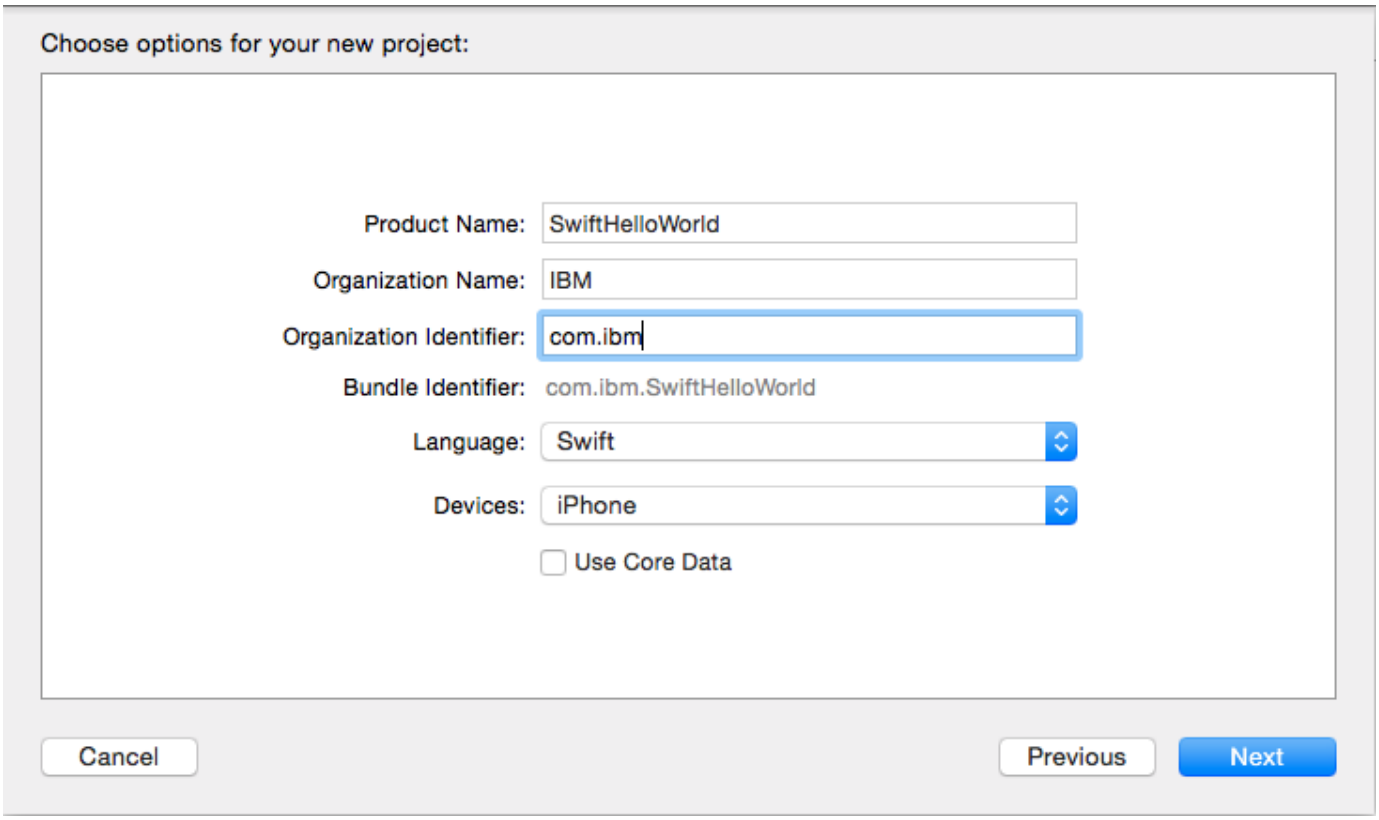
After you run the application in Xcode, the final result is a native application that contains the MobileFirst API library.



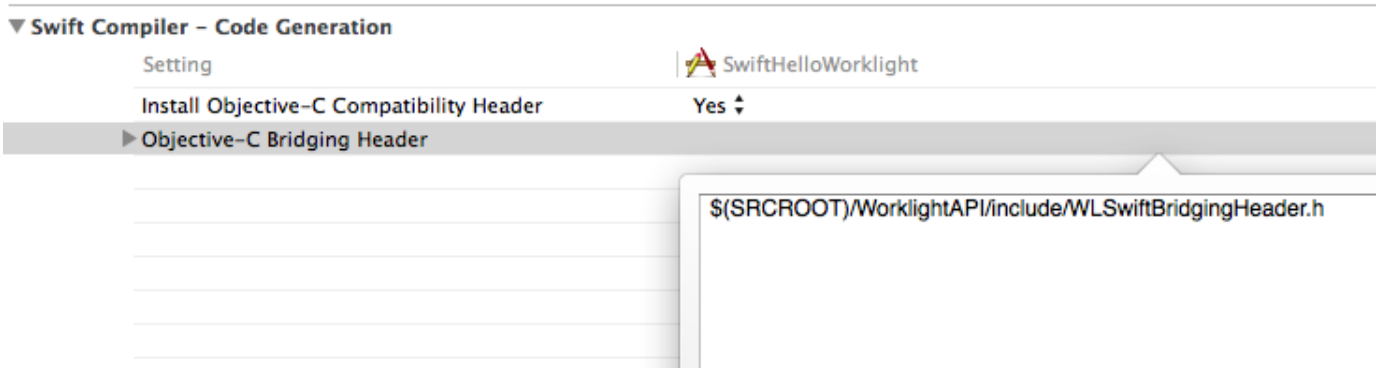
Configuring a Swift application

Because Apple Swift is designed to be compatible with Objective-C, you can use the MobileFirst SDK from within an iOS Swift project, too.
Create a Swift project and follow the same steps, as described at the beginning of the tutorial, to install the

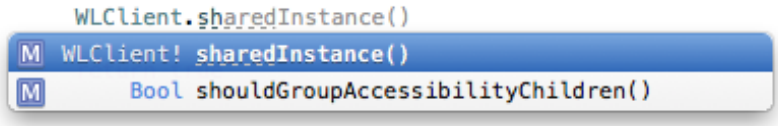
MobileFirst SDK into an iOS native application.



After all the normal steps for an iOS application, go to **Build Settings > Swift Compiler - Code Generation**. In **Objective-C Bridging Header**, add **$(SRCROOT)/WorklightAPI/include/WLSwiftBridgingHeader.h** However, if you already have your own Bridging Header for other purposes, include the Worklight Bridging Header inside your own Bridging Header instead.



All the MobileFirst classes are now available from any of your Swift files.
XCode provides code autocompletion converted to the Swift style.



Sample application

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/HelloWorldNativeProject.zip) the Studio project.
Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/iOSHelloWorldNativeProject.zip) the Native project.