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
     Console
     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`:

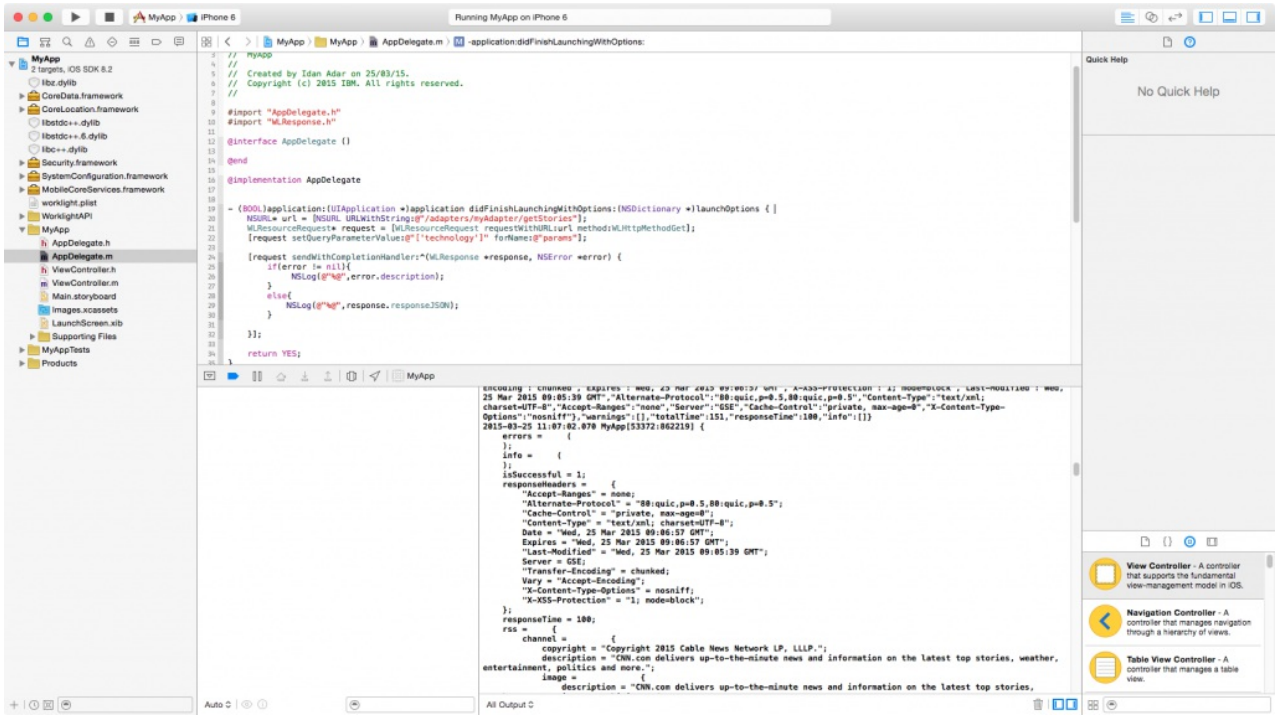
```
- (BOOL)application:(UIApplication *)application didFinishLaunchingW
ithOptions:(NSDictionary *)launchOptions {
    NSURL* url = [NSURL URLWithString:@"/adapters/MyAdapter/getFeed"];
    WLResourceRequest* request = [WLResourceRequest requestWithURL:url
method:WLHttpMethodGet];
    [request setQueryParameterValue:@"['technology']" forName:@"params"
];

    [request sendWithCompletionHandler:^(WLResponse *response, NSError
*error) {
        if(error != nil){
             NSLog(@"%@",error.description);
        }
        else{
            NSLog(@"%@",response.responseJSON);
        }
    }];

    return YES;
}
```

6. Final configurations

- Supply the machine's IP address for the `host` property in `worklight.plist`

7. **Click Run**

Review the Xcode console for the data retrieved by the adapter request.



Last modified on