MobileFirst Platform {dev}

Invoking adapter procedures from native iOS Objective-C applications

Relevant to:
Native iOS

To create and configure an iOS native project, first follow the "Configuring a native iOS application with the MobileFirst Platform SDK" tutorial.

MobileFirst applications can adapt procedures to communicate with any data source. This tutorial explains how to use the REST API for returning data from an HTTP adapter. The same can be applied using other data sources (such as SQL adapters, etc).

Initializing WLClient

1. Access the `WLClient` functionality by using `[WLClient sharedInstance]` anywhere in your application.
2. Initiate the connection to the server by using the `wlConnectWithDelegate` method.
   For most actions, you must specify a delegate object, such as a `MyConnectListener` instance in the following example:

   ```
   MyConnectListener *connectListener = [[MyConnectListener alloc]
   initWithController:self];
   [[WLClient sharedInstance] wlConnectWithDelegate:connectListener];
   ```

   Note: Remember to `#import <IBMMobileFirstPlatformFoundation/IBMMobileFirstPlatformFoundation.h>` when using the MobileFirst SDK.

3. Create a delegate to be used in the `wlConnectWithDelegate` method and receive the response from the MobileFirst Server instance. Name the class `MyConnectListener`.
   The header file must specify that it implements the `WLDelegate` protocol.

   ```
   @interface MyConnectListener : NSObject <WLDelegate> {
       @private
       ViewController *vc;
   }
   ```

   The `WLDelegate` protocol specifies that the class implements the following methods:
   – The `onSuccess` method: `(WLResponse *)response`
   – The `onFailure` method: `(WLFailResponse *)response`

   After `wlConnectWithDelegate` finishes, the `onSuccess` method or the `onFailure` method of the supplied `MyConnectListener` instance is invoked.
   In both cases, the response object is sent as an argument.

4. Use this object to operate data that is retrieved from the server.

   ```
   -(void)onSuccess:(WLResponse *)response{
   ```

```
    NSLog(@"\nConnection Success: %@", response);
    NSString *resultText = @"Connection success. ";

    if ([response responseText] != nil){
        resultText = [resultText stringByAppendingString:[response
responseText]];
    }
    [vc updateView:resultText];
}

-(void)onFailure:(WLFailResponse *)response{
    NSString *resultText = @"Connection failure. ";

    if ([response responseText] != nil){
        resultText = [resultText stringByAppendingString:[response
responseText]];
    }
    [vc updateView:resultText];
}
```

Invoking an adapter procedure

After the connection is established with a MobileFirst Server instance, you can use the `WLResourceRequest` class to invoke adapter procedures or call any REST resources.

1. Define the URI of the resource. For a JavaScript HTTP adapter: `/adapters/{AdapterName}/{ProcedureName}`

   ```
   NSURL* url = [NSURL URLWithString:@"/adapters/RSSReader/getStories"];
   ```

2. Create a `WLResourceRequest` object and choose the HTTP method (GET, POST, etc).

   ```
   WLResourceRequest* request = [WLResourceRequest requestWithURL:url
   method:WLHttpMethodGet];
   ```

3. Add the required parameters.
   - For JavaScript-based adapters, use the `params` parameter name to set an array of parameters.

     ```
     [request setQueryParameterValue:@"['technology']" forName:@"params"];
     ```

   - For Java adapters or other resources, you can use `setQueryParameterValue` for each parameter.

     ```
     [request setQueryParameterValue:@"value1" forName:@"param1"];
     [request setQueryParameterValue:@"value2" forName:@"param2"];
     ```

4. Trigger the request with a call to the `sendWithCompletionHandler` method. Specify a `completionHandler` instance.

   ```
   [request sendWithCompletionHandler:^(WLResponse *response, NSError
    *error) {
       NSString* resultText;
       if(error != nil){
           resultText = @"Invocation failure.";
           resultText = [resultText stringByAppendingString:
   error.description];
       }
       else{
           resultText = @"Invocation success.";
   ```

```
            resultText = [resultText
    stringByAppendingString:response.responseText];
        }
        [self updateView:resultText];
    }];
```

Other signatures, which are not covered in this tutorial, exist for the `send` method. Those signatures enable you to set parameters in the body instead of the query, or to handle the response with a delegate instead of a completion handler. See the user documentation to learn more.

Sample and result

Click to download the MobileFirst project.

Click to download the Native project.

- The `InvokingAdapterProcedures` project contains a **MobileFirst native API** which you can deploy to your MobileFirst Server instance and required to deploy to the server.
- The `InvokingAdapterProceduresObjC` project contains a **native iOS application** that uses a MobileFirst native API library to communicate with the MobileFirst Server instance.
- Make sure to update the `worklight.plist` file in **NativeiOSInvoking** with the relevant server settings.