iOS - Using native pages in hybrid applications

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.1/adding-native-functionality/ios-using-native-pages-hybrid-applications.html) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new)

Overview

This tutorial explains how to integrate native and web "pages" in an iOS application by using the `WL.NativePage.show()` API to open a native page from JavaScript.

With this method, you can have data sent from JavaScript to the open native page, and specify a callback to be called after the native page closes.

This tutorial covers the following topics:

- Connecting to the plugin from the JavaScript code
- Creating a native page
- Returning control to the web view
- Sample application

Connecting to the plugin from the JavaScript code

1. Implement the `WL.NativePage.show()` method to open the native page:

```
function openNativePage(){
    var params = {
        nameParam : $('#nameInput').val()
    };
    WL.NativePage.show(nativePageClassName, backFromNativePage, params)
;
}
```

    - `nativePageClassName`: The name of a native iOS `UIViewController` instance to start.
    - `backFromNativePage`: A callback function to call when the native page closes.
    - `params`: Optional custom parameters object to pass to the native code.

2. To handle the callback function, write the following code:

```
function backFromNativePage(data){
    alert("Received phone number is: " + data.phoneNumber)
;
}
```

    The `backFromNativePage(data)` parameter can pass data back to the web part of an application after the native closes.

Creating a native page

To manage a native page, proceed as follows:

1. Open the generated iOS projects in Xcode.
2. Add a new Cocoa Touch Class file, make sure that it is a subclass of `UIViewController` and save the file in the `Classes` folder of the Xcode project.
   Important:
   If you work with existing `.m` and `.h` files, reference the files while in Xcode.
   Placing the `.m` and `.h` files only in the `iphone\native\Classes` folder in Eclipse is not sufficient, because these files are not referenced in the Xcode project unless they have been added in Xcode.
3. To retrieve custom data parameters that are passed from the web view, use the `setDataFromWebView:(NSDictionary*)data` method:

```
-(void)setDataFromWebView:(NSDictionary*)data{
    self.nameParam = (NSString*)[data valueForKey:@"nameParam"]
;
}
```

Returning control to the web view

When the native page switches back to the web view, a call to the `[NativePage showWebView:]` method is necessary.
To pass data back to the web view, use the `NSDictionary` object. For example:

Objective-C

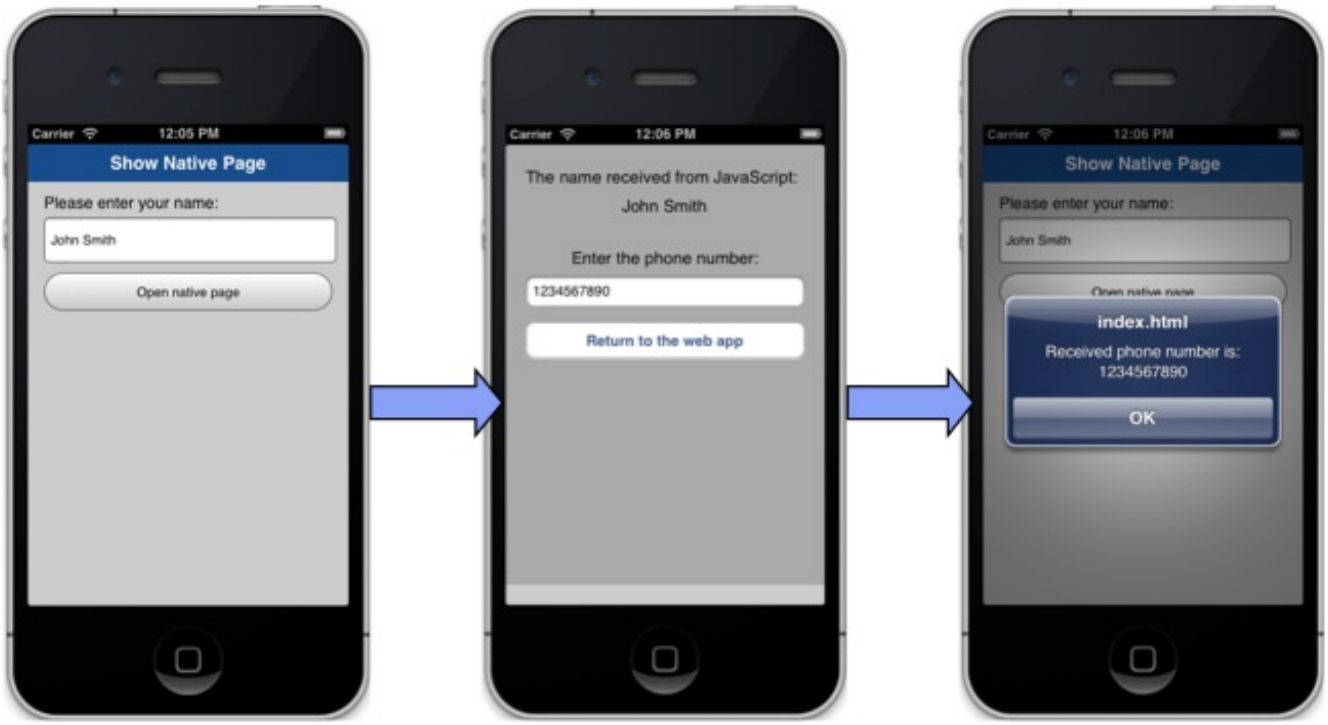
```
-(IBAction)returnClicked:(id)sender{
    NSString *phone = [phoneNumber text];
    NSDictionary *returnedData = [NSDictionary dictionaryWithObject:phone forKey:@"phoneNumber"]
;
    [NativePage showWebView:returnedData];
}
```

JavaScript

```
function backFromNativePage(data){
    alert("Received phone number is: " + data.phoneNumber)
;
}
```

Sample application

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/NativePagesInHybridApp/tree/release71) the MobileFirst project.


Carrier
12:05 PM
Show Native Page
Please enter your name:
John Smith
Open native page
Carrier
12:06 PM
The name received from JavaScript:
John Smith
Enter the phone number:
1234567890
Return to the web app
Carrier
12:06 PM
Show Native Page
Please enter your name:
John Smith
index.html
Received phone number is:
1234567890
OK