Event Source Push Notifications in Hybrid Applications

- Download sample (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/PushNotificationsHybridProject.zip)

Overview

This tutorial expends on the implementation of event source push notification client-side API, and covers the following topics:

- Sending the token and initializing the `WLPush` class
- Registering the event source
- Subscribing to/unsubscribing from the event source

Notification API - Client-side

Methods

- WL.Client.Push.isPushSupported() – returns `true` if push notifications are supported by the platform, or `false` otherwise.
- WL.Client.Push.isSubscribed(alias) – returns whether the currently logged-in user is subscribed to a specified event source alias.

When a push notification is received by a device, the callback function defined in WL.Client.Push.registerEventSourceCallback is invoked:

[code lang="js"]function pushNotificationReceived(props, payload) {
alert("pushNotificationReceived invoked");
alert("props :: " + JSON.stringify(props));
alert("payload :: " + JSON.stringify(payload));
}[/code]

If the application was in background mode (or inactive) when the push notification arrived, this callback function is invoked when the application returns to the foreground.

Back-end emulator

The sample project for this tutorial is bundled with a back-end emulator which can be used to simulate push notification submissions by a back-end system. The source for the emulator can be found in the sample project.

To run the back-end emulator, open the PushBackendEmulator folder of the sample project in a command prompt, and then run the supplied JAR file by using the following syntax:

[code]java –jar PushBackendEmulator.jar <userId> <message> <contextRoot> <port>[/code]

userId is the user name that you used to log in to the sample application.
contextRoot is the context root of your MobileFirst project.

Example

[code]java –jar PushBackendEmulator.jar JohnDoe “My first push notification” myContextRoot 10080[/code]

The back-end emulator tries to connect to a MobileFirst Server and call asubmitNotification() adapter procedure. It outputs the invocation result to a command prompt console.

Success

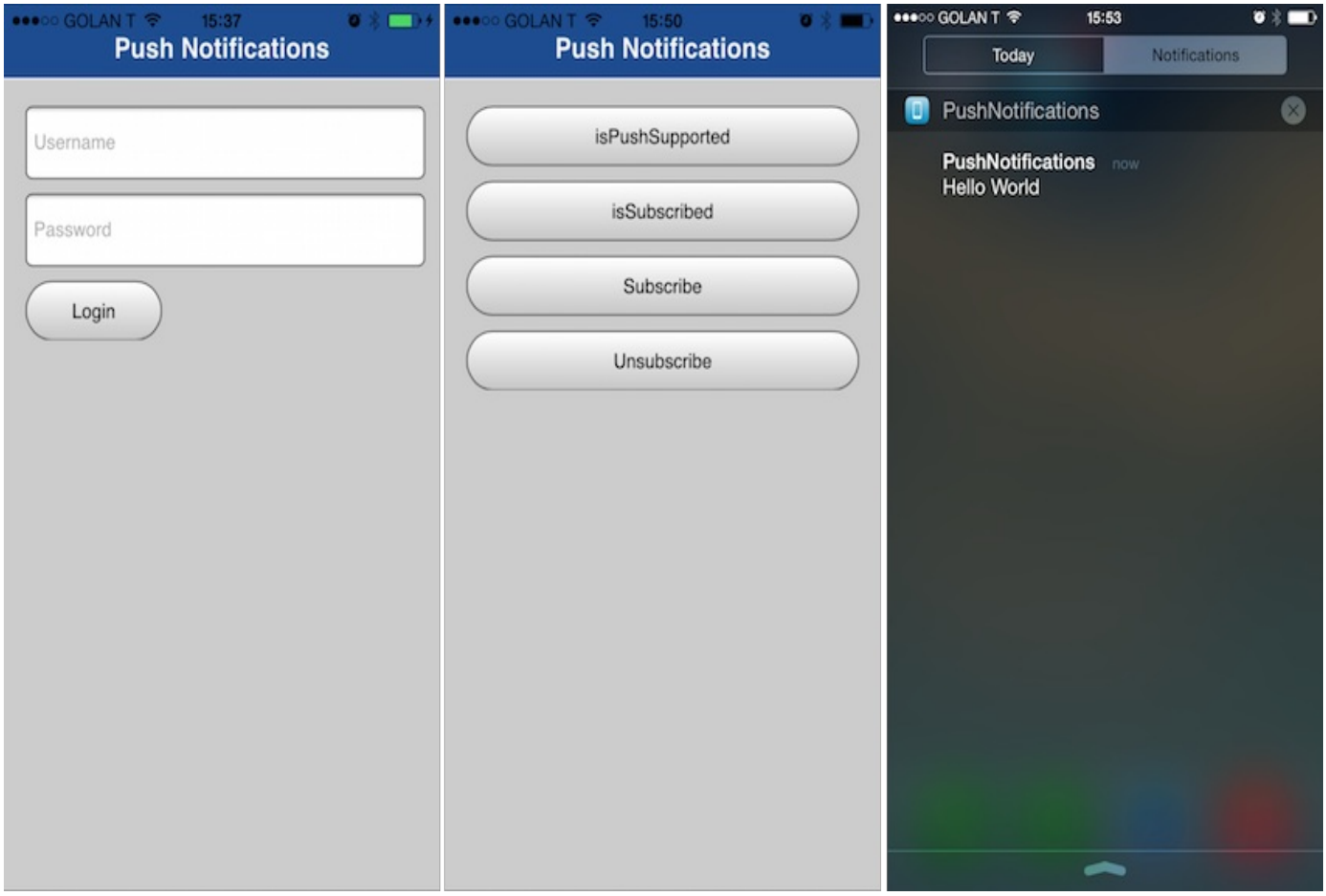
```
c:\>java -jar C:\PushBackendEmulator.jar JohnDoe "hello push" PushNotificationsProject 10080
PushBackendEmulator
User Id: JohnDoe
Notification text: hello push
Server URL: http://localhost:10080/PushNotificationsProject
sending notification
Server response :: {   "isSuccessful": true,   "result": "Notification sent to user :: JohnDoe"}
```

(http://developer.ibm.com/mobilefirstplatform/wp-content/uploads/sites/32/2014/07/08_01_push_success.png)

Failure

```
c:\>java -jar C:\PushBackendEmulator.jar JohnMissing "hello push" PushNotificationsProject 10080
PushBackendEmulator
User Id: JohnMissing
Notification text: hello push
Server URL: http://localhost:10080/PushNotificationsProject
sending notification
Server response :: {   "isSuccessful": true,   "result": "No subscription found for user :: JohnMissing"}
```

(http://developer.ibm.com/mobilefirstplatform/wp-content/uploads/sites/32/2014/07/08_01_push_failure.png)



(http://developer.ibm.com/mobilefirstplatform/wp-content/uploads/sites/32/2014/07/08_01_push_sample.png)

Sample application

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/PushNotificationsHybridProject.zip) the sample application.