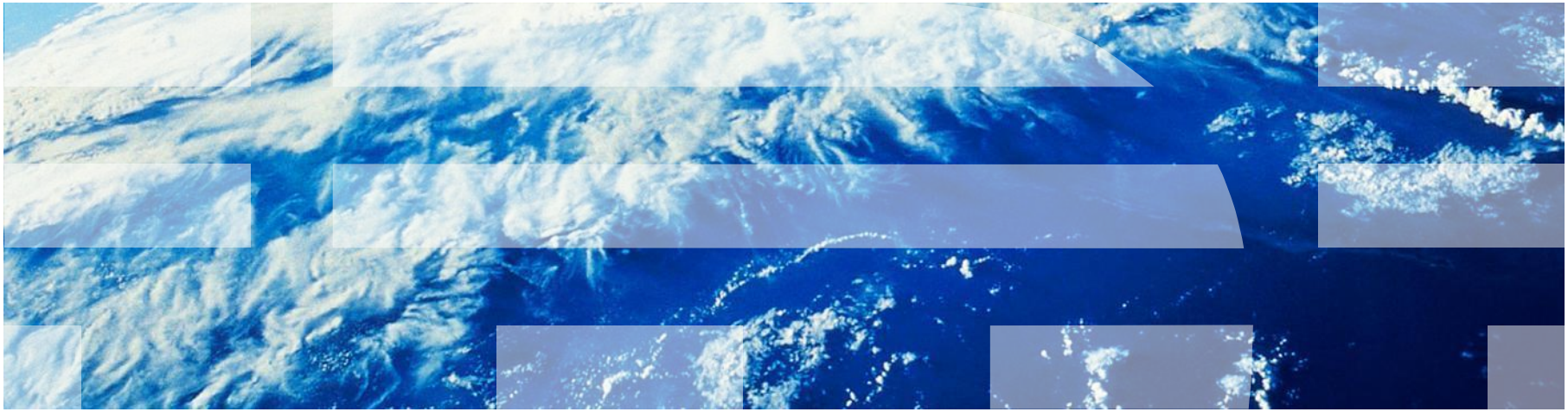
IBM Worklight Foundation V6.2.0 入門

ハイブリッド・アプリケーションでの ネイティブ iOS UI エレメントの追加



2014 年 9 月 2 日

商標

- IBM、IBM ロゴ、ibm.com および Worklight は、世界の多くの国で登録された International Business Machines Corporation の商標です。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。IBM、IBM ロゴ、ibm.com および Worklight は、世界の多くの国で登録された International Business Machines Corporation の商標です。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml をご覧ください。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- この資料は、事前に IBM の書面による許可を得ずにその一部または全部を複製することは禁じられています。

IBM® について

- http://www.ibm.com/ibm/us/en/ を参照してください。

アジェンダ

- 概要
- 開始フロー
- ネイティブ SplashScreen のサンプル
- JavaScript からネイティブへの送信アクション
- ネイティブから JavaScript への送信アクション
- SendAction のサンプル
- 共有セッション

概要

- Web テクノロジーのみを使用してハイブリッド・アプリケーションを作成できますが、IBMWorklight® Foundation® を使用すると、必要に応じてネイティブ・コードと Web コードをミックス・アンド・マッチさせることができます。
- 例えば、一部のネイティブ UI コントロールを使用すること、アニメーション化したネイティブ概要画面を提供すること、iOS または Android が提供するネイティブ・エレメントを使用することなどを選択できます。
- これを行うには、ハイブリッド・アプリケーションの開始フローの一部を制御する必要があります。
- このチュートリアルは、ネイティブ iOS 開発の作業知識があることを前提とします。

アジェンダ

- 概要
- 開始フロー
- ネイティブ SplashScreen のサンプル
- JavaScript からネイティブへの送信アクション
- ネイティブから JavaScript への送信アクション
- SendAction のサンプル
- 共有セッション

開始フロー (1/2)

- 新しいハイブリッド・アプリケーションを作成する場合、アプリケーションの開始フローの各段階を処理するアプリケーション・デリゲートが Worklight により生成されます。このファイル (YourAppName.m) を開き、デフォルトのフローについて学習してください。
- Worklight は、空の `UIViewController` インスタンス (このサンプルでは `Compatibility50ViewController` と呼ばれる) を始動し、これをアプリケーションのルート・ビューとして設定します。
- `showSplashScreen` メソッドが呼び出され、リソースのロード中に簡単なスプラッシュ画面が表示されます。この行は、任意のネイティブ概要画面に置き換えることができます。
- `initializeWebFrameworkWithDelegate` メソッドにより、Web ビューを正しく動作させるために必要なリソースがロードされます。
- Web フレームワークの初期化が完了した直後に `wlInitWebFrameworkDidCompleteWithResult` メソッドが呼び出されます。

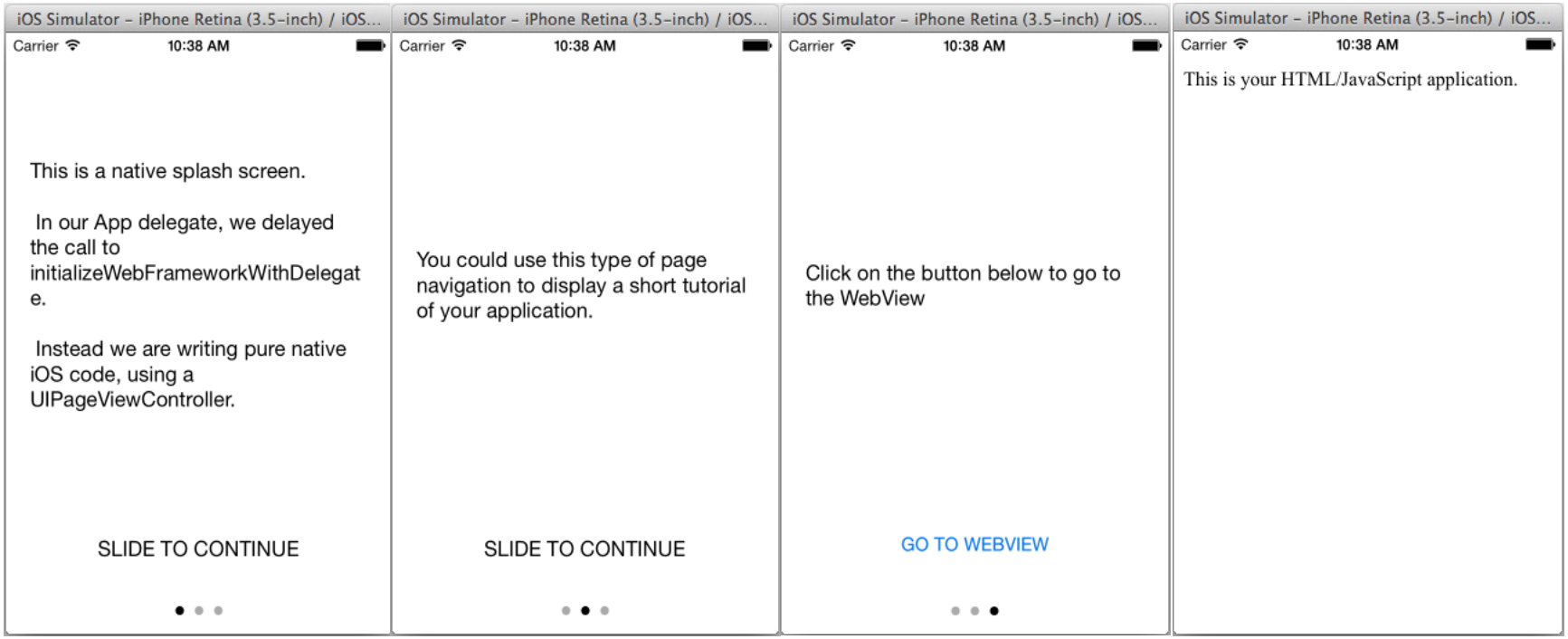
開始フロー (2/2)

- この時点で、デフォルトでは、アプリケーションにスプラッシュ画面がまだ表示されており、Web ビューはまだ表示されません。
- この実装を変更して、フレームワークにより返される多くの状況コードを処理できます。
- デフォルトでは、ロードが成功すると、`wlInitDidCompleteSuccessfully` メソッドが呼び出されます。
- Cordova Web ビュー・コントローラー (`CDVViewController` クラス) が初期化され、アプリケーションの開始ページが設定されます。
- 次に、Cordova コントローラーがルート・ビューの子として追加され、画面に表示されます。
- 前述したように独自のカスタム概要画面を作成する場合、ネイティブ画面での作業が終了するまで、Cordova ビューの表示を遅らせることが必要な場合があります。

アジェンダ

- 概要
- 開始フロー
- ネイティブ SplashScreen のサンプル
- JavaScript からネイティブへの送信アクション
- ネイティブから JavaScript への送信アクション
- SendAction のサンプル
- 共有セッション

ネイティブSplashScreen サンプル (1/7)



ネイティブ SplashScreen サンプル (2/7)

- NativeUIInHybrid プロジェクトをダウンロードします。このプロジェクトには、NativeSplashScreen という名前のハイブリッド・アプリケーションが含まれます。
- このサンプルでは、Page View Controller を使用して、アプリケーションにスライド式の概要を表示します。
- ユーザー・インターフェースは、XCode の Storyboard を使用して作成されています。
- スライドを処理するために、2 つのクラス `PageViewController` および `PageContentViewController` が作成されています。
- `UIPageViewController` でチュートリアルをオンライン検索します。

ネイティブSplashScreen サンプル (3/7)

- MyAppDelegate.didFinishLaunchingWithOptions メソッドで、ウィンドウが初期化され、PageViewController インスタンスがウィンドウのルートとして設定されます。

```
@implementation MyAppDelegate
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    //...
        self.window = [[UIWindow alloc] initWithFrame:[[UIScreen
mainScreen] bounds]];

        PageViewController* pageViewController = [[UIStoryboard
storyboardWithName:@"Storyboard" bundle: nil]
instantiateViewControllerWithIdentifier:@"PageViewController"];
        [self.window setRootViewController:pageViewController];
        [self.window makeKeyAndVisible];
    //..
}
```

ネイティブSplashScreen サンプル (4/7)

- initializeWebFrameworkWithDelegate メソッドが didFinishLaunchingWithOptions メソッド内から呼び出されます。
- このメソッドにより、バックグラウンドで Worklight フレームワークが初期化されます。フレームワークが初期化されると wlInitWebFrameworkDidCompleteWithResult メソッドが呼び出されます。

```
@implementation MyAppDelegate
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
        //...
    [self.window setRootViewController:pageViewController];
    [self.window makeKeyAndVisible];
    [[WL sharedInstance] initializeWebFrameworkWithDelegate:self];
    return result;
}
```

ネイティブSplashScreen サンプル (5/7)

- wlInitWebFrameworkDidCompleteWithResult メソッド内で、WLWebFrameworkInitResult オブジェクトの statusCode 値に基づき、異なるシナリオを処理できます。
- このサンプルでは、一般的なケースの WLWebFrameworkInitResultSuccess 値のみ変更されています。

```
-(void)wlInitWebFrameworkDidCompleteWithResult:(WLWebFrameworkInitResult
*)result
{
    if ([result statusCode] == WLWebFrameworkInitResultSuccess) {
        [self wlInitDidCompleteSuccessfully];
    } else {
        [self wlInitDidFailWithResult:result];
    }
}
```

ネイティブSplashScreen サンプル (6/7)

- `wlInitDidCompleteSuccessfully` で、Cordova コントローラーが準備されますが、まだ表示されません。
- オプションで、JavaScript コードの初期化をバックグラウンドで開始する場合は、フレームをそれ自体に設定して、バックグラウンドで Web ビューが初期化されるようにします。

```
-(void)wlInitDidCompleteSuccessfully
{
        // Create a Cordova View Controller
        self.cordovaViewController = [[CDVViewController alloc] init] ;
        self.cordovaViewController.startPage = [[WL sharedInstance]
mainHtmlFilePath];

    //This will trigger initialization in the background, optional
    self.cordovaViewController.view.frame =
self.cordovaViewController.view.frame;
}
```

ネイティブ SplashScreen サンプル (7/7)

- このサンプルでは、PageViewController インスタンスは、onSplashScreenDone (AppDelegate) という名前のカスタム・メソッドをトリガーするボタンで終了します。
- onSplashScreen カスタム・メソッドは、フローが中断された場所から再開され、事前に初期化されている Cordova ビューを表示します。

```
-(void)onSplashScreenDone {
    UIViewController* rootViewController = [[Compatibility50ViewController
alloc] init];

    [self.window setRootViewController:rootViewController];
    [self.window makeKeyAndVisible];

    self.cordovaViewController.view.frame = rootViewController.view.bounds;

    [rootViewController addChildViewController:self.cordovaViewController];
    [rootViewController.view addSubview:self.cordovaViewController.view];
}
```

アジェンダ

- 概要
- 開始フロー
- ネイティブ SplashScreen のサンプル
- JavaScript からネイティブへの送信アクション
- ネイティブから JavaScript への送信アクション
- SendAction のサンプル
- 共有セッション

JavaScript からネイティブへの送信アクション (1/2)

- Worklight アプリケーションで、(JavaScript を使用した) Web ビューから (Objective-C で作成された) ネイティブ・クラスにパラメーターを含むコマンドを送信します。
- この機能を使用すると、ネイティブ・コードをトリガーしてバックグラウンドで実行すること、ネイティブ UI を更新すること、ネイティブのみの機能を使用することなどができます。
- JavaScript で、`WL.App.sendActionToNative("doSomething", { customData: 12345} );` を記述します。
  ここで、`doSomething` は、ネイティブ側で使用する任意のアクション名で (次のスライドを参照)、2 番目のパラメーターはデータを含む JSON オブジェクトです。

JavaScript からネイティブへの送信アクション (2/2)

- アクションを受け取るネイティブ・クラスで WLActionReceiver プロトコルを実装する必要があります。

```
@interface MyReceiver: NSObject <WLActionReceiver>{}
```

- WLActionReceiver プロトコルには onActionReceived メソッドが必要です。このメソッドで、アクション名をチェックし、アクションに必要なネイティブ・コードを実行します。

```
-(void) onActionReceived:(NSString *)action withData:(NSDictionary *) data
{
      if ([action isEqualToString:@"doSomething"]){
         // perform required actions, e.g., update native user interface
      }
}
```

- アクションの受信側で Worklight Web ビューからアクションを受け取るために、このアクションを登録する必要があります。アプリケーションの開始フロー中にアクションを登録することで、十分早くアクションをキャッチできます。

```
[[WL sharedInstance] addActionReceiver:myReceiver];
```

アジェンダ

- 概要
- 開始フロー
- ネイティブ SplashScreen のサンプル
- JavaScript からネイティブへの送信アクション
- ネイティブから JavaScript への送信アクション
- SendAction のサンプル
- 共有セッション

ネイティブから JavaScript への送信アクション (1/2)

- Worklight アプリケーションで、ネイティブ Objective-C コードから Web ビューの JavaScript コードにパラメーターを含むコマンドを送信できます。
- この機能を使用すると、ネイティブ・メソッドから応答を受け取ること、バックグラウンド・コードの実行終了時に Web ビューに通知すること、ネイティブ UI で Web ビューのコンテンツを制御することなどができます。
- Objective-C から以下を使用します: `sendActionToJS`

```
NSDictionary *data = @{@"someProperty": @"12345"};

[[WL sharedInstance] sendActionToJS:@"doSomething" withData:data];
```

ここで、「doSomething」は、JavaScript 側で使用する任意のアクション名で (次のスライドを参照)、2 番目のパラメーターはデータを含む `NSDictionary` オブジェクトです。

ネイティブから JavaScript への送信アクション (2/2)

- アクション名を検証し、JavaScript コードを実装する任意の名前の JavaScript 関数を作成します。

```
function actionReceiver(received){
    if (received.action == "doSomething" && received.data.someProperty ==
    "12345"){
      //perform required actions, e.g., update web user interface
    }
}
```

- アクションを受け取るためにこの JavaScript 関数を登録します。この関数で、できるだけ早くこれらのアクションを処理するために、JavaScript コードで十分に早い段階に関数を登録します。

```
WL.App.addActionReceiver ("MyActionReceiverId", actionReceiver);
```

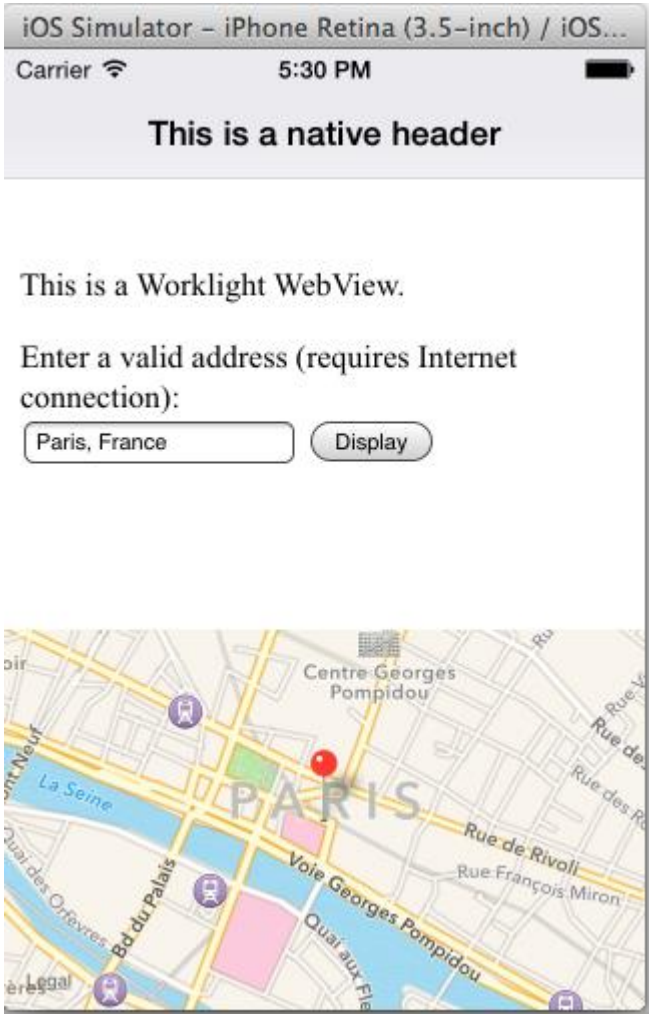
- 1 番目のパラメーターは任意の名前です。これを後で使用して、アクションの受信側を削除できます。

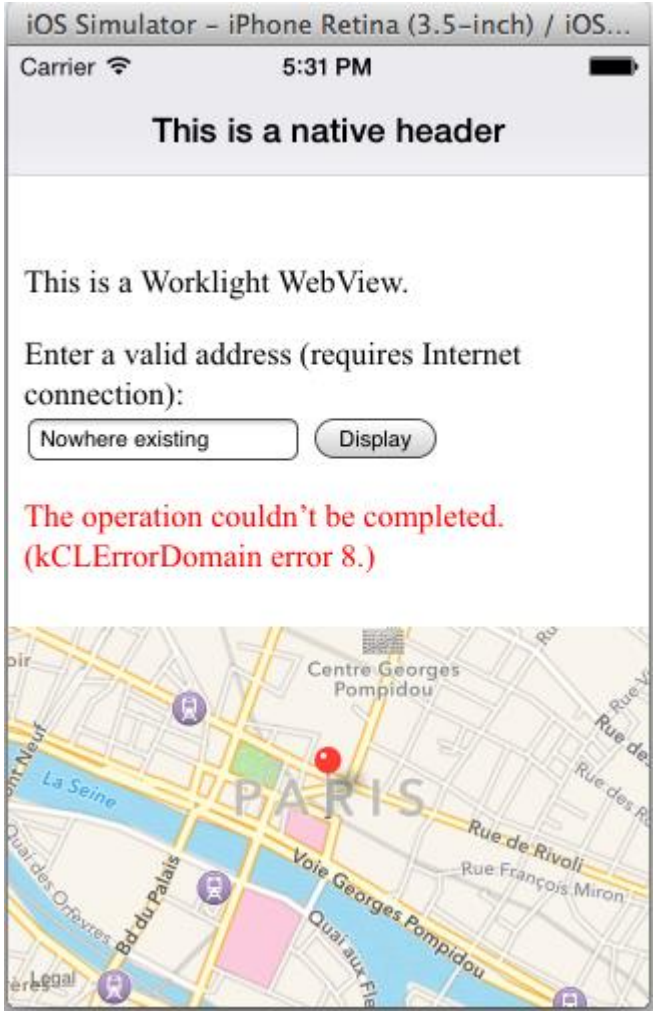
```
WL.App.removeActionReceiver("MyActionReceiverId");
```

アジェンダ

- 概要
- 開始フロー
- ネイティブ SplashScreen のサンプル
- JavaScript からネイティブへの送信アクション
- ネイティブから JavaScript への送信アクション
- SendAction のサンプル
- 共有セッション

SendAction サンプル - プレビュー





SendAction サンプル - 概要

- NativeUIInHybrid プロジェクトをダウンロードします。このプロジェクトには、SendAction という名前のハイブリッド・アプリケーションが含まれます。
- このサンプルでは、画面が 2 つの部分に分割されます。
- 上半分は Cordova Web ビューで、住所を入力するためのフォームが含まれます。
- 下半分はネイティブ・マップ・ビューで、入力した場所が有効な場合には、その場所が表示されます。
- 住所が無効の場合、ネイティブ・マップはエラーを Web ビューに転送し、その Web ビューでエラーが表示されます。
- このサンプルには MapKit フレームワークが必要です。

SendAction サンプル - HTML

- HTML ページには、次のオブジェクトが表示されます。
  - 住所を入力するために簡単な入力フィールド。
  - 検証をトリガーするためのボタン。
  - エラー・メッセージがある場合にそれを表示するための空の <p> 行。

```
<p>This is a Worklight WebView.</p>
<p>Enter a valid address (requires Internet connection):<br/>
        <input type="text" name="address" id="address"/>
    <input type="button" value="Display" id="displayBtn"/>
 </p>
 <p id="errorMsg" style="color:red;"></p>
```

SendAction サンプル - JavaScript

- ボタンをクリックすると、sendActionToNative メソッドが呼び出され、住所がネイティブ・コードに送信されます。

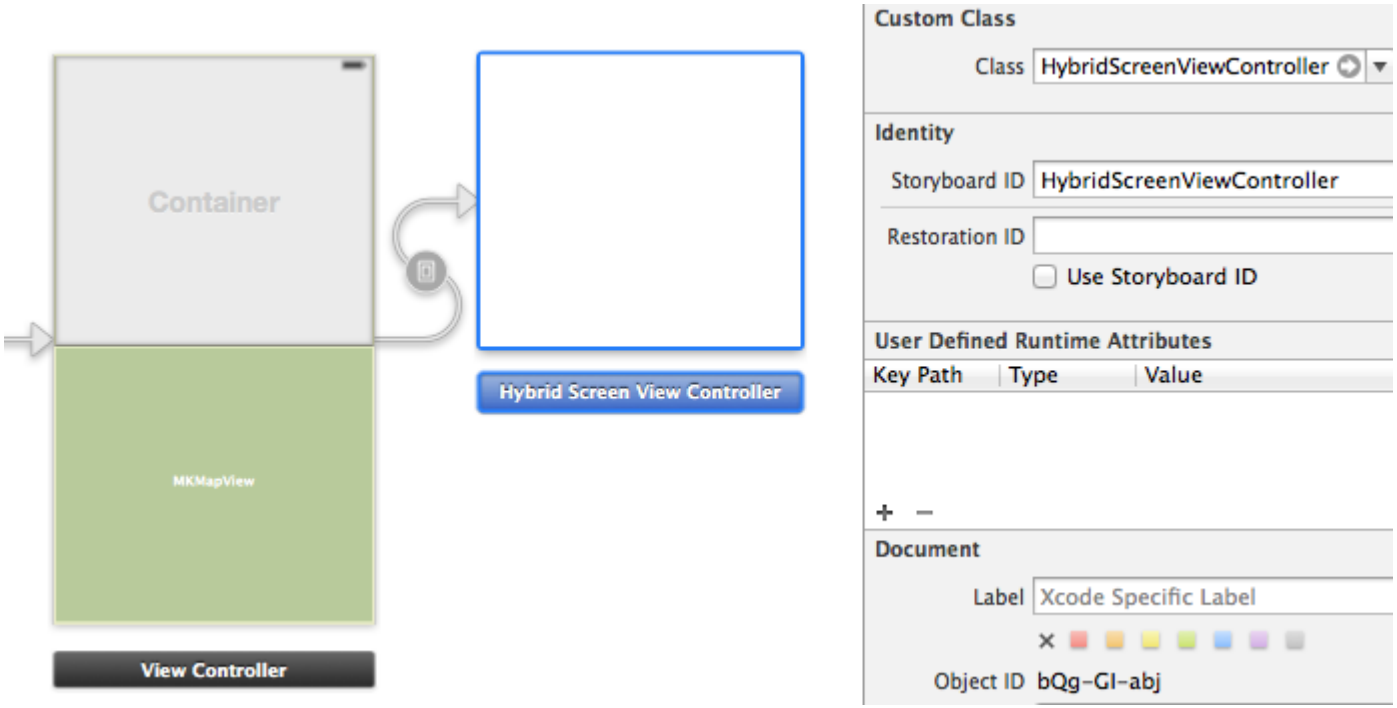
```
$('#displayBtn').on('click', function(){
$('#errorMsg').empty();
WL.App.sendActionToNative("displayAddress",
                        { address: $('#address').val()}
    );
});
```

- ネイティブ・コードから送られてくる可能性のあるエラー・メッセージを表示するために、このコードでアクションの受信側も登録します。

```
WL.App.addActionReceiver ("MyActionReceiverId", function
actionReceiver(received){
    if(received.action == 'displayError'){
            $('#errorMsg').html(received.data.errorReason);
    }
});
```

SendAction サンプル - Storyboard

- インターフェースは Storyboard ファイルで設計されています。
- これは、`ViewController` カスタム・クラスを使用した汎用的なビュー・コントローラーを特徴とします (後述します)。
- このビュー・コントローラーには `MKMapView` オブジェクトとコンテナー・ビューが含まれます。
- コンテナー・ビューには、`HybridScreenViewController` クラスを使用するように設定されたビュー・コントローラーが含まれます (後述します)。



SendAction サンプル - HybridScreenViewController

- HybridScreenViewController は、CDVViewController (Worklight 提供の Cordova Web ビュー) を拡張したものです。@interface HybridScreenViewController : CDVViewController
- このクラスの実装は、Cordova Web ビューの startPage の設定を除いてほとんど空です。

```
@implementation HybridScreenViewController
- (id)initWithCoder:(NSCoder*)aDecoder {
    self = [super initWithCoder:aDecoder];
    self.startPage = [[WL sharedInstance] mainHtmlFilePath];
    return self;
}
//...
```

SendAction サンプル – ViewController (1/5)

- ViewController クラスは UIViewController を拡張したクラスです。
- このクラスには、プロパティーとして MKMapView オブジェクトへの参照が含まれます。
- このクラスは MKMapViewDelegate プロトコルに準拠しています。これは、マップに関する更新を受け取るためです。
- このクラスは WLActionReceiver プロトコルに準拠しています。これは、Worklight Web ビューからアクションを受け取るためです。
- このクラスには、ジオコーディングの住所を使用可能にするために CLGeocoder オブジェクトへの参照が含まれます。

```
@interface ViewController ()<MKMapViewDelegate, WLActionReceiver>
@property (weak, nonatomic) IBOutlet MKMapView *map;
@property CLGeocoder* geocoder;
@end
```

SendAction サンプル - ViewController (2/5)

- コントローラーのタイトルが設定され、`UINavigationController` オブジェクトの一部として表示されます。
- geocoder が初期化されます。
- マップ・デリゲートが設定されます。
- `ViewController` が Worklight のアクション受信側として登録されます。

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    self.title = @"This is a native header";
    self.geocoder = [[CLGeocoder alloc] init];
    [self.map setDelegate:self];
    [[WL sharedInstance] addActionReceiver:self];
}
```

SendAction サンプル - ViewController (3/5)

- ユーザーがフォームを送信すると、onActionReceived メソッドが呼び出されます。
- アクション名がチェックされ、入力された住所が取得されます。
- geocoder に住所が渡されます。

```
-(void) onActionReceived:(NSString *)action withData:(NSDictionary *) data {
    if ([action isEqualToString:@"displayAddress"]
        && [data objectForKey:@"address"]){
        NSString* address = (NSString*) [data objectForKey:@"address"];
        [self.geocoder geocodeAddressString:address
                    completionHandler:^(NSArray* placemarks, NSError* error){
                              //DO STUFF - next slide...

                    }];
    }
}
```

SendAction サンプル - ViewController (4/5)

- 場所が見つかると、その地域が中心となり、新しい MKPlacemark がマップに追加されます。

```
completionHandler:^(NSArray* placemarks, NSError* error){
    if([placemarks count]){
        CLPlacemark *topResult = [placemarks objectAtIndex:0];
        float spanX = 0.00725;
        float spanY = 0.00725;
        MKCoordinateRegion region;
        region.center.latitude = topResult.location.coordinate.latitude;
        region.center.longitude = topResult.location.coordinate.longitude;
        region.span = MKCoordinateSpanMake(spanX, spanY);
        [self.map setRegion:region animated:YES];

        MKPlacemark *placemark = [[MKPlacemark alloc]initWithPlacemark:topResult];
        [self.map addAnnotation:placemark];
    }
}
```

SendAction サンプル - ViewController (5/5)

- 検索が失敗した場合、または場所が見つからない場合、sendActionToJS メソッドが呼び出され、エラーが Web ビューに送信されます。

```
completionHandler:^(NSArray* placemarks, NSError* error){
    if([placemarks count]){
        //...
    }
    else{
        [[WL sharedInstance] sendActionToJS:@"displayError"
                                withData:@{@"errorReason": [error localizedDescription]}
         ];
    }
}
```

SendAction サンプル - MyAppDelegate

- デフォルトの Compatibility50ViewController クラスは、UIViewController クラスではなく UINavigationController クラスを拡張するために変更されています。この変更により、ネイティブ・ヘッダーが追加されます。

  ```
  @interface Compatibility50ViewController : UINavigationController
  ```

- アプリケーション・デリゲートの didFinishLaunchingWithOptions メソッドは、Worklight により生成され、この例では未変更のままです。
- wlInitDidCompleteSuccessfully 状況コードは変更され、CDVViewController オブジェクトを直接ロードする代わりに、Storyboard から ViewController オブジェクトがロードされます。
- JavaScript はまだ始動していないため、スプラッシュ画面はネイティブ・コードでは非表示です。

```
-(void)wlInitDidCompleteSuccessfully
{
    UINavigationController* rootViewController = self.window.rootViewController;
    ViewController* viewController = [[UIStoryboard storyboardWithName:@"Storyboard"
bundle:nil] instantiateViewControllerWithIdentifier:@"ViewController"];
    [rootViewController pushViewController:viewController animated:YES];
    [[WL sharedInstance] hideSplashScreen];
}
```

アジェンダ

- 概要
- 開始フロー
- ネイティブ SplashScreen のサンプル
- JavaScript からネイティブへの送信アクション
- ネイティブから JavaScript への送信アクション
- SendAction のサンプル
- 共有セッション

共有セッション

- 同じアプリケーションで JavaScript とネイティブ・コードの両方を使用するとき、場合により Worklight サーバーに対して HTTP 要求を行う必要があります (接続、プロシージャーの呼び出しなど)。
- HTTP 要求については他のチュートリアルで説明します (ハイブリッドとネイティブの両方)。
- Worklight 6.2 以降、JavaScript クライアントとネイティブ・クライアントの間でセッション (Cookie および HTTP ヘッダー) 自動的に同期を保ちます。

特記事項

- これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。
- 本書は米国 IBM が提供する製品およびサービスについて作成したものです。
- 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作または サービスの 評価および検証は、お客様の責任で行っていただきます。
- IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。
  - 〒103-8510
    東京都中央区日本橋箱崎町19番21号
    日本アイ・ビー・エム株式会社
    法務・知的財産
    知的財産権ライセンス渉外
- **以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含む すべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。**
- この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。
- 本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。
- IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、 自ら適切と信ずる方法で、使用もしくは配布することができるものとします。
- 本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- 本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。
- 本書で説明されているライセンス・プログラムまたはその他のライセンス資 料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。
- IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、 他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

著作権使用許諾:

- 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。
- それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次の ように、著作権表示を入れていただく必要があります。
  - © (お客様の会社名) (西暦年).このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. _年を入れる_. All rights reserved.

プライバシー・ポリシーの考慮事項

- サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。
- このソフトウェア・オファリングは、展開される構成に応じて、(アプリケーション・サーバーが生成する) セッション情報を収集するセッションごとの Cookie を使用する場合があります。これらの Cookie は個人情報を含まず、セッション管理のために要求されるものです。加えて、匿名ユーザーの認識および管理のために持続的な Cookie が無作為に生成される場合があります。これらの Cookie も個人情報を含まず、要求されるものです。
- この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

サポートおよびコメント

- IBM Worklight の一連の文書、トレーニング資料、および質問をポストできるオンライン・フォーラムはすべて、次の IBM Web サイトからご覧になれます。
  - http://www.ibm.com/mobile-docs
- **サポート**
  - ソフトウェア・サブスクリプション & サポート (ソフトウェア・メンテナンスと呼ばれる場合もあります) は、パスポート・アドバンテージおよびパスポート・アドバンテージ・エクスプレスから購入されたライセンスに含まれています。International Passport Advantage Agreement および IBM International Passport Advantage Express Agreement の追加情報については、次のパスポート・アドバンテージ Web サイトを参照してください。
    - http://www.ibm.com/software/passportadvantage
  - ソフトウェア・サブスクリプション & サポートが有効になっている場合、IBM は、インストールおよび使用法 (ハウツー) に関する短期間の FAQ に対するサポートや、コード関連の質問に対するサポートを提供します。詳しくは、次の IBM ソフトウェア・サポート・ハンドブックを参照してください。
    - http://www.ibm.com/support/handbook
- **ご意見**
  - 本資料に関するご意見をお寄せください。本資料の具体的な誤りや欠落、正確性、編成、題材、または完成度に関するご意見をお寄せください。お寄せいただくご意見は、本マニュアルまたは製品の情報、およびその情報の提示方法に関するもののみとしてください。
  - 製品の技術的な質問および情報、および価格については、担当の IBM 営業所、IBM ビジネス・パートナー、または認定リマーケターにお問い合わせください。
  - IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。IBM またはいかなる組織も、お客様から提示された問題についてご連絡を差し上げる場合にのみ、お客様が提供する個人情報を使用するものとします。
  - どうぞよろしくお願いいたします。
  - 次の IBM Worklight Developer Edition サポート・コミュニティーにご意見をお寄せください。
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - IBM からの回答を希望される場合は、以下の情報をご連絡ください。
    - 氏名
    - 住所
    - 企業または組織
    - 電話番号
    - E メール・アドレス

ありがとうございました