MobileFirst Foundation development in Cordova applications

Overview

From http://cordova.apache.org/ (http://cordova.apache.org/):

> Apache Cordova is an open-source mobile development framework. It allows you to use standard web technologies such as HTML5, CSS3, and JavaScript for cross-platform development, avoiding each mobile platforms' native development language. Applications execute within wrappers targeted to each platform, and rely on standards-compliant API bindings to access each device's sensors, data, and network status.

IBM MobileFirst Foundation provides an SDK in the form of several Cordova plug-ins. Learn how to Add the MobileFirst Foundation SDK to Cordova applications (../../application-development/sdk/cordova).

Jump to:

Cordova application development

Applications developed with Cordova can be further enhanced by using the following Cordova-provided development paths and features:

Hooks

Cordova Hooks are scripts that provide developers with the ability to customize Cordova commands, enabling to create for example custom build flows.
Read more about Cordova Hooks (http://cordova.apache.org/docs/en/dev/guide/appdev/hooks/index.html#Hooks%20Guide).

Merges

The Merges folder provides the ability to have platform-specific web resources (HTML, CSS and JavaScript files). These web resources are then deployed during the `cordova prepare` step to the appropriate native directory. Files placed under the **merges**/ folder will override matching files in the **www**/

folder of the relevant platform. Read more about the Merges folder (https://github.com/apache/cordova-cli#merges).

Cordova plug-ins

Using Cordova plug-ins can provide enhancements such as adding native UI elements (dialogs, tabbars, spinners and the like), as well as more advanced functionalities such as Mapping and Geolocation, loading of external content, custom keyboards, Device integration (camera, contacts, sensors, and so on).

You can find Cordova plug-ins on GitHub.com (https://github.com) and in popular Cordova plug-ins websites, such as Plugreg (http://plugreg.com/) and NPM (http://npmjs.org).

Example plug-ins:

- cordova-plugin-dialogs (https://www.npmjs.com/package/cordova-plugin-dialogs)
- cordova-plug-inprogress-indicator (https://www.npmjs.com/package/cordova-plugin-progress-indicator)
- cordova-plugin-statusbar (https://www.npmjs.com/package/cordova-plugin-statusbar)

3rd-party frameworks

Cordova application development can be further enhanced by using frameworks such as Ionic (http://ionicframework.com/), AngularJS (https://angularjs.org/), jQuery Mobile (http://jquerymobile.com/), Backbone (http://backbonejs.org/) and many others.

3rd-party packages

Applications can be modified using 3rd party packages to achieve requirements such as Minification & Concatenation of the application's web resources and more. Popular packages to do so are:

- uglify-js (https://www.npmjs.com/package/uglify-js)
- clean-css (https://www.npmjs.com/package/clean-css)

MobileFirst APIs

After adding the MobileFirst Cordova SDK (../../application-development/sdk/cordova) to a Cordova application, the MobileFirst set of API methods is now available for use.

> For a complete list of available API methods, refer to the "Client API changes in V8.0.0" topic in the user documentation.

MobileFirst SDK Startup Flow

Android startup flow

iOS startup flow

Windows startup flow

Cordova Application Security

IBM MobileFirst Foundation provides security features that help you protect your Cordova apps.

Much of the content in a cross-platform app can be more easily modified by an unauthorized person than for a native app. Because much of the common content in a cross-platform app is in a readable format, IBM MobileFirst Foundation provides features that can provide a higher level of security for your cross-platform Cordova apps.

> Learn more about the MobileFirst security framework (../../authentication-and-security)

Use the following features to improve security on your Cordova apps:

- Encrypting the web resources of your Cordova packages (securing-apps/#encrypting-the-web-resources-of-your-cordova-packages)
  Encrypts the contents in the www folder of your Cordova app, and decrypts it when the app is installed and run for the first time. This encryption makes it more difficult for someone to view or modify the content in that folder while the app is packaged.
- Enabling the web resources checksum feature (securing-apps/#enabling-the-web-resources-checksum-feature)
  Ensures the integrity of the app when it starts by comparing the contents to the baseline checksum results that were gathered the first time the app was started. This test helps prevent the modification of an app that is already installed.
- Enabling FIPS 140-2 (../../administering-apps/federal/#enabling-fips-140-2)
  Ensures that the encryption algorithms that are used to encrypt data at rest and data in motion are compliant with the Federal Information Processing Standards (FIPS) 140-2 standard.
- Certificate Pinning (../../authentication-and-security/certificate-pinning)
  Helps you prevent man-in-the-middle attacks by associating a host with its expected public key.

Cordova Application Resources

Certain resources are needed as part of the Cordova application. In most cases, they are generated for you when you create your Cordova app with your preferred Cordova development tools. If you use the IBM MobileFirst Foundation template, then splash screens and icons are also provided.

You can use a project template that is provided by IBM for use with Cordova projects that are enabled to use MobileFirst features. If you use this MobileFirst template, the following resources are made available to you as a starting point. If you do not use the MobileFirst template, all of the resources are provided except splash screens and icons. You add the template by specifying the `--template` option and the MobileFirst template when you initially create your Cordova project.

If you change the default file names and paths of any resources, you must also specify such changes in the Cordova configuration file (config.xml). In addition, in some cases, you can change the default names and paths with the mfpdev app config command. If you can change names and paths with the mfpdev app config command, it is noted in the section about the specific resource.

Cordova configuration file (config.xml)

The Cordova configuration file is a required XML file that contains application metadata and is stored in the root directory of the app. The file is automatically generated when you create a Cordova application. You can modify it to add custom properties by using the mfpdev app config command.

Main file (index.html)

This main file is an HTML5 file that contains the application skeleton. This file loads all the web resources (scripts and stylesheets) that are necessary to define the general components of the application and to

hook to required document events. You can find this file in the **your-project-name**/**www** directory. You can change the name of this file with the `mfpdev app config` command.

Thumbnail image

The thumbnail image provides a graphical identification for the application on the MobileFirst Operations Console. It must be a square image, preferably of size 90 by 90 pixels.
A default thumbnail image is provided when you use the template. You can override the default image by using the same file name with a replacement image. You can find thumbnail.png in the **your-project-name**/**www**/**img** folder. You can change the name or path of this file with the `mfpdev app config` command.

Splash image

The splash image is displayed while the application is being initialized. If you use the MobileFirst default template, splash images are provided. These default images are stored in the following directories:

- iOS: /res/screen/ios/
- Android: /res/screen/android/
- Windows: /res/screen/windows/

Various default splash images are included that are appropriate for different displays, and for iOS and Windows, different versions of the operating system. You can replace the default image that is provided by the template with your own splash image, or add an image if you did not use the template. When you use the MobileFirst template to build your app for the Android platform, the **cordova-plugin-splashscreen** plug-in is installed. When this plug-in is integrated, the splash images that Cordova uses are displayed instead of the images that are used by MobileFirst. The images in the folder with the screen.png format are the Cordova standard splash images. You can specify which splash images display by changing the settings in the Cordova **config.xml** file.

If you do not use the MobileFirst template, the default splash images that are displayed are the images that are used by the MobileFirst plug-in. The file names of the default MobileFirst source splash images are in the form **splash-string.9.png**.

> For more information about using your own splash images, seeAdding custom splash screens and icons to Cordova apps (adding-images-and-icons).

Application icons

Default images for application icons are provided with the template. These default images are stored in the following directories:

- iOS: /res/icon/ios/
- Android: /res/icon/android/
- Windows: /res/icon/windows/

You can replace the default image with your own image. Your custom application image must match the size of the default application image that you are replacing, and must use the same file name. Various default images are provided, appropriate to different displays and operating system versions.

> For more information about using your own splash images, seeAdding custom splash screens and icons to Cordova apps (adding-images-and-icons).

Stylesheets

The app code can include CSS files to define the application view.

The stylesheet files are located in the /www/css directory, and are copied to the following platform-specific folders:

- iOS: /platforms/ios/www/css
- Android: /platforms/android/assets/www/css
- Windows: /platforms/windows/www/css

Scripts

Your app code can include JavaScript files that implement various functions of your app such as interactive user interface components, business logic, and back-end query integration.

The JavaScript file index.js is provided by the template, and is located in the **your-project-name/www/js** folder. This file is copied to the following platform-specific folders:

- iOS: /platforms/ios/www/js
- Android: /platforms/android/assets/www/js
- Windows: /platforms/windows/assets/www/js

Previewing an application's web resources

A Cordova application's web resources can be previewed either in the iOS Simulator, Android Emulator, Windows Emulator or physical devices. In MobileFirst Foundation, two additional live-preview options are available: IBM Mobile Browser Simulator and Simple Browser rendering.

> **Security Restriction:** You can preview your web resources, however not all MobileFirst JavaScript APIs are supported by the simulator. In particular, the OAuth protocol is not fully supported. However, you can test calls to adapters with WLResourceRequest. In this case,
>
> - Security checks are not run on the server-side and security challenges are not sent to the client that runs in Mobile Browser Simulator.
> - If you do not use MobileFirst Development Server, register a confidential client that has the adapter's scope in its list of allowed scopes. You can define a confidential client with the MobileFirst Operations Console, by using the Runtime/Settings menu. For more information about confidential clients, see Confidential clients (../../authentication-and-security/confidential-clients).
>
> **Note:** MobileFirst Development Server includes a confidential client "test" that has an unlimited allowed scope ("*"). By default mfpdev app preview uses this confidential client.

Simple Browser

In Simple Browser previewing, the web resources of the application are being rendered in the desktop browser without being treated as an "app", allowing easy debugging of just the web resources.

Mobile Browser Simulator

The Mobile Browser Simulator is a web application that enables testing of the Cordova application by simulating device features without needing to install the app in an Emulator or physical device.

Supported browsers:

- Firefox version 38 and later
- Chrome 49 and later
- Safari 9 and later

Previewing

1. From a **Command-line** window, run the command:

   ```
   mfpdev app preview
   ```

2. Select a preview option:

   ```
   ? Select how to preview your app: (Use arrow keys)
   ❯ browser: Simple browser rendering
   mbs: Mobile Browser Simulator
   ```

3. Select a platform to preview (only added platform will be displayed):

   ```
   ❯◯ android
   ◯ ios
   ```

> **Tip:** Learn more about the various CLI commands in the Using CLI to manage MobileFirst artifacts (../using-mobilefirst-cli-to-manage-mobilefirst-artifacts/) tutorial.

Live preview

Applicative code (HTML, CSS and JS) can now be edited in real-time with live-preview.
After making a change to a resource, save the change and it will be immediately reflected in the browser.

Live reload

To achieve a similar effect while previewing in physical devices or simulators/emulators, add the **cordova-plugin-livereload** plug-in. For usage instructions, see the plug-ins GitHub page (https://github.com/omefire/cordova-plugin-livereload).

Running the application on emulator or on a physical device

To emulate the application execute the Cordova CLI command `cordova emulate ios|android|windows`. For example:

```
cordova emulate ios
```

To run the application on a physical device, attached to the development workstation a run the Cordova CLI command `cordova run ios|android|windows`. For example:

```
cordova run ios
```

Implementing JavaScript Code

Editing the WebView resources is more convenient using an IDE that provides autocompletion for JavaScript.

Xcode, Android Studio, and Visual Studio provide full editing capabilities for editing Objective C, Swift, C#, and Java, however they may be limited in how they assist the editing of JavaScript. To facilitate JavaScript editing, the MobileFirst Cordova project contains a defintion file for providing autocomplete for MobileFirst API elements.

Each MobileFirst Cordova plug-in provides a `d.ts` configuration file for each MobileFirst JavaScript files. The `d.ts` file name matches the corresponding JavaScript file name and is located within the plug-in folder. For example for the main MobileFirst SDK the file is here: **[myapp]\plugins\cordova-plugin-mfp\typings\worklight.d.ts**.

This definition provides autocomplete for all IDEs with TypeScript support: TypeScript Playground (http://www.typescriptlang.org/Playground/), Visual Studio Code (http://www.microsoft.com/visualstudio/eng), WebStorm (http://www.jetbrains.com/webstorm/), WebEssentials (http://visualstudiogallery.msdn.microsoft.com/6ed4c78f-a23e-49ad-b5fd-369af0c2107f), Eclipse (https://github.com/palantir/eclipse-typescript).

The resources (HTML and JavaScript files) for the WebView are located in the **[myapp]\www** folder. When the project is built with the cordova build command, or the cordova prepare command is run, these resources are copied to the corresponding **www** folder in the **[myapp]\platforms\ios\www**, **[myapp]\platforms\android\assets\www**, or **[myapp]\platforms\windows\www** folder.

When you open the main app folder with one of the previous IDEs, the context is preserved. The IDE editor will now be linked to the relevant `d.ts` files and autocomplete the MobileFirst API elements as you type.

CrossWalk support for Android

Cordova applications for the Android platform can have their default WebView replaced with the CrossWalk WebView (https://crosswalk-project.org/).
To add it:

1. From a **Command-line** line, run the command:

   ```
   cordova plugin add cordova-plugin-crosswalk-webview
   ```

   This command will add the CrossWalk WebView to the application.

   Behind the scenes, the MobileFirst Cordova SDK will adjust the Android project activity for using it.
2. Build the project by running the command:

   ```
   cordova build
   ```

WKWebView support for iOS

The default UIWebView used in Cordova iOS applications can be replaced with Apple's WKWebView (https://developer.apple.com/library/ios/documentation/WebKit/Reference/WKWebView_Ref/).
To add, run the following command from a command-line window: `cordova plugin add cordova-plugin-wkwebview-engine`.

> Learn more about the Cordova WKWebView plug-in (https://github.com/apache/cordova-plugin-wkwebview-engine).

Further reading

Learn more about Cordova:

- Cordova Overview (https://cordova.apache.org/docs/en/latest/guide/overview/index.html)
- Cordova best practices, testing, debugging, cosiderations and keeping up (https://cordova.apache.org/docs/en/latest/guide/next/index.html#link-testing-on-a-simulator-vs-on-a-real-device)
- Get started with Cordova applications development (https://cordova.apache.org/#getstarted)

Tutorials to follow next

Get started by adding the MobileFirst SDK to your Cordova application (../../application-development/sdk/cordova), and review MobileFirst-provided features in the All Tutorials (../../all-tutorials/) section.