

Préparation de l'environnement de développement

Introduction

Avant d'entamer le développement d'applications mobiles hybrides il est impératif de préparer l'environnement de développement. Le cœur de cet environnement est le framework Apache Cordova. Celui-ci va offrir les commandes nécessaires pour pouvoir créer un projet pour une application mobile hybride, la gérer, la compiler et la tester.



Présentation

Apache Cordova est un framework de développement mobile hybride gratuit et open source. A l'origine ce framework a porté le nom de PhoneGap. Il a été développé en 2009 par la société Nitobi Software. Suite au rachat de cette société par Adobe Systems en 2011 le projet a été repris par Adobe. Celle-ci a décidé la même année de le passer au domaine du libre en l'offrant à la fondation Apache comme un projet gratuit en open source. Tout d'abord il a été renommé par Apache en « Apache Callback ». Puis, le projet a été renommé en « Apache Cordova ».

Adobe Systems continue à travailler sur le projet PhoneGap qui propose désormais des fonctionnalités qui se basent sur Apache Cordova.

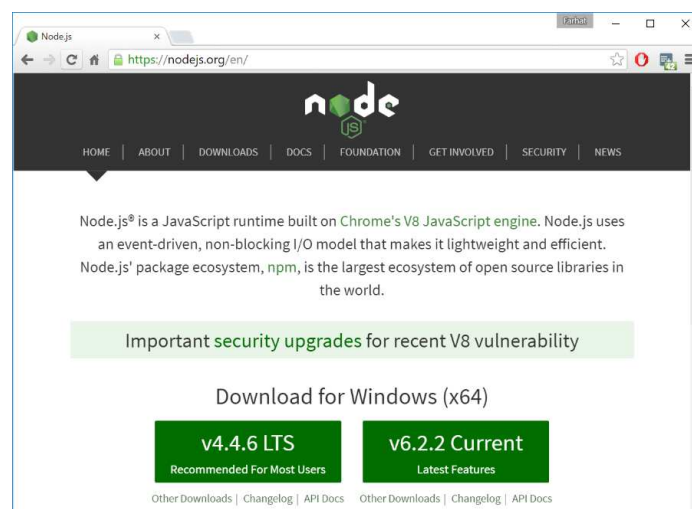
Principe

Apache Cordova offre les moyens nécessaires pour créer une application mobile hybride. Rappelons qu'une application mobile hybride est une application qui est développée en utilisant les langages du Web : HTML 5, CSS 3 et JavaScript. Ce code Web est exécuté dans un Webview. Un Webview est l'équivalent d'un navigateur Web classique mais dépourvu des composants graphiques d'un navigateur, tels qu'à titre d'exemples : la barre d'adresse, les boutons de navigation, la barre d'outils et la barre d'état. En plus, une application mobile hybride est capable d'utiliser les fonctionnalités natives d'un dispositif mobile via des appels en JavaScript grâce à une API (Application Program Interface). Ainsi, un même code source peut être compilé pour plusieurs plateformes mobiles.

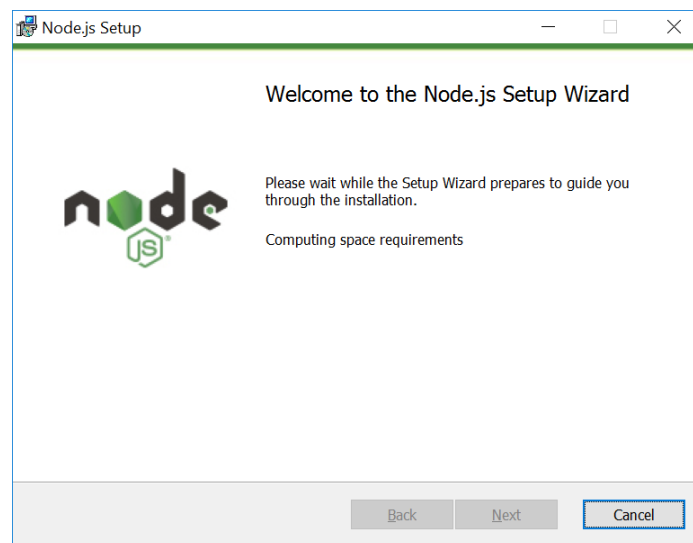
Justement, le framework Apache Cordova permet de générer des applications mobiles hybrides. Il permet donc d'encapsuler le code Web dans un Webview natif et d'offrir une API permettant d'exploiter les fonctionnalités natives du dispositif mobile. Il est possible d'utiliser des plugins pour étendre les fonctionnalités de l'application mobile hybride. En effet, Apache Cordova permet, quel que soit la plateforme cible, d'utiliser par exemple l'accéléromètre ou la caméra.

Installation

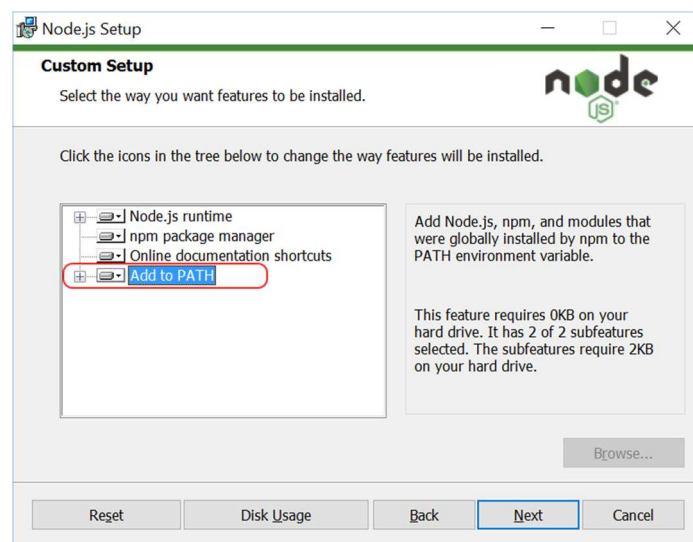
Pour installer Apache Cordova il est impératif de commencer par installer « Node.js ». Il s'agit d'une plateforme logicielle libre en JavaScript qui permet entre autres d'installer des paquets via son gestionnaire de paquets « npm » (Node.js Package Manager). En fait, Apache Cordova est justement offert sous forme d'un paquetage « Node.js ». L'adresse de téléchargement de « Node.js » est : <https://nodejs.org>



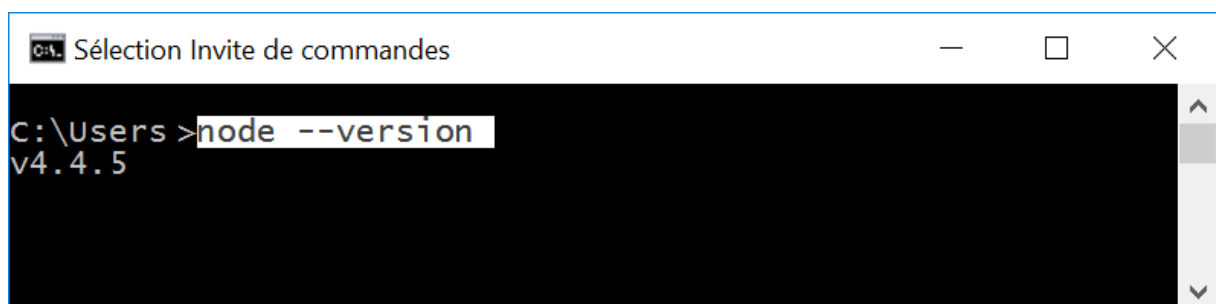
Après le téléchargement, il suffit de lancer l'installation :



Lors de l'installation vérifiez que « Add to PATH » est incluse dans l'installation (c'est le cas par défaut) afin d'éviter d'éditer à la main la variable système PATH par la suite :



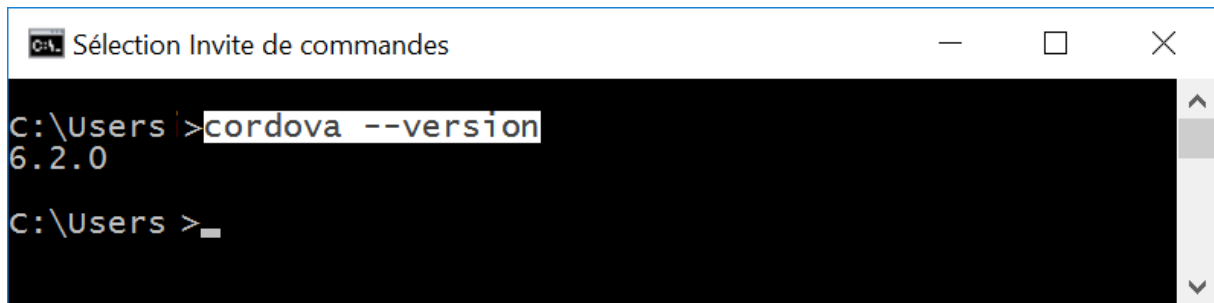
Une fois l'installation est terminée vous pouvez lancer votre interpréteur de commande et vérifier la disponibilité de « Node.js ». Par exemple, vous pouvez demander l'affichage de la version installée de « Node.js ».



Maintenant, il faut passer à l'installation de Apache Cordova via le gestionnaire de paquets « npm ». Il faut donc lancer la version suivante :

```
npm install -g cordova
```

Une fois l'installation est terminée il suffit de vérifier que Apache Cordova est fonctionnel en invoquant par exemple la commande qui affiche la version installée :

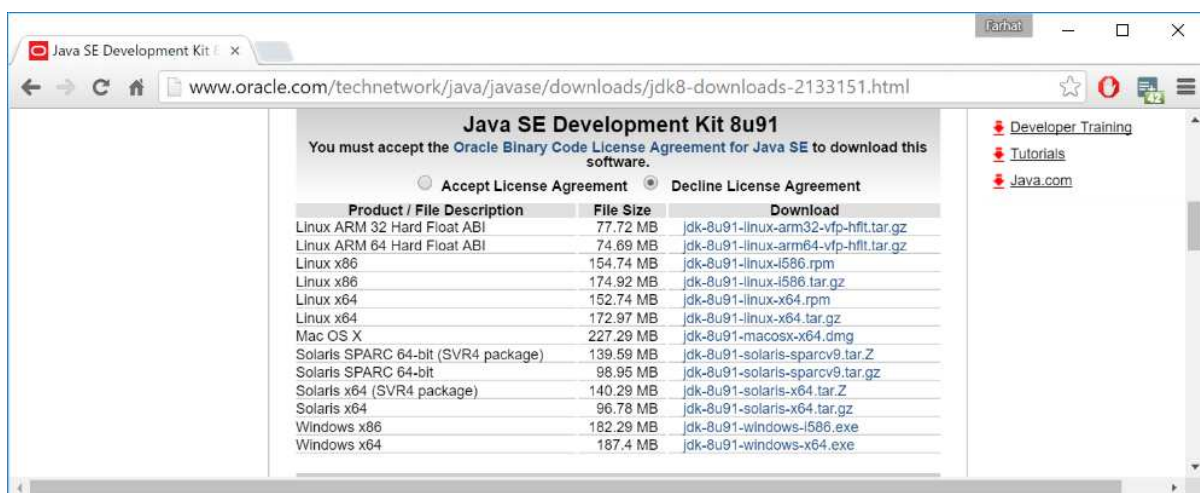


```
C:\Users > cordova --version
6.2.0
C:\Users >
```

A partir de ce moment vous pouvez utiliser les commandes Cordova pour la création d'une application mobile hybride. Toutefois, il n'est pas encore possible de compiler pour un environnement mobile donné que si vous installez le SDK (Software Development Kit) associé.

Ajout du SDK Android

Le SDK Android nécessite pour son fonctionnement la présence du JDK (Java Development Kit). Si vous ne l'avez pas déjà installé sur votre machine vous pouvez le télécharger à l'adresse suivante : <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>



Vous pouvez donc télécharger et installer la version qui correspond à votre système d'exploitation. Après il suffit de lancer le programme d'installation qui va se charger par défaut

d'ajouter le JDK à la variable PATH du système. Finalement, pour vérifier que vous disposez du JDK sur votre système via la commande :

```
java -version
```

Maintenant il faut passer au téléchargement du SDK Android qui est disponible à l'adresse suivante (à la fin de la page) : <https://developer.android.com/studio/index.html>

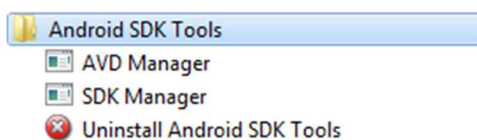
Get just the command line tools

If you do not need Android Studio, you can download the basic Android command line tools below. You can use the included [sdkmanager](#) to download other SDK packages.

These tools are included in Android Studio.

Platform	SDK tools package	Size	SHA-256 checksum
Windows	sdk-tools-windows-3859397.zip	132 MB (138,449,982 bytes)	7f6037d3a7d6789b4fdc06ee7af041e071e9860c51f66f7a4eb5913df9871fd2
Mac	sdk-tools-darwin-3859397.zip	82 MB (86,182,133 bytes)	4a81754a760fce88cba74d69c364b05b31c53d57b26f9f82355c61d5fe4b9df9
Linux	sdk-tools-linux-3859397.zip	130 MB (136,964,098 bytes)	444e22ce8ca0f67353bda4b85175ed3731cae3ffa695ca18119cbacef1c1bea0

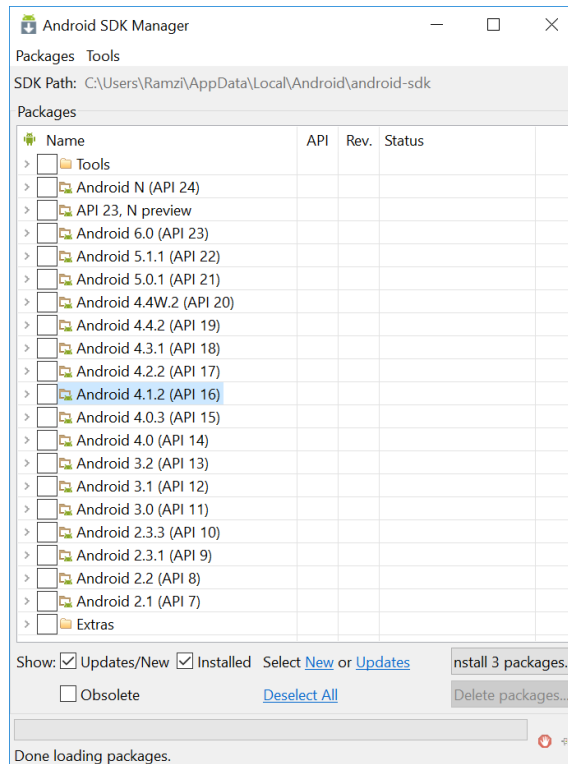
Une fois l'installateur est lancé il suffit de suivre les étapes. Ce qui va vous permettre d'avoir deux programmes : le SDK Manager et le AVD Manager. Le SDK manager permet de gérer les paquetages et les outils nécessaires pour le développement et la compilation d'applications mobiles pour l'environnement Android.



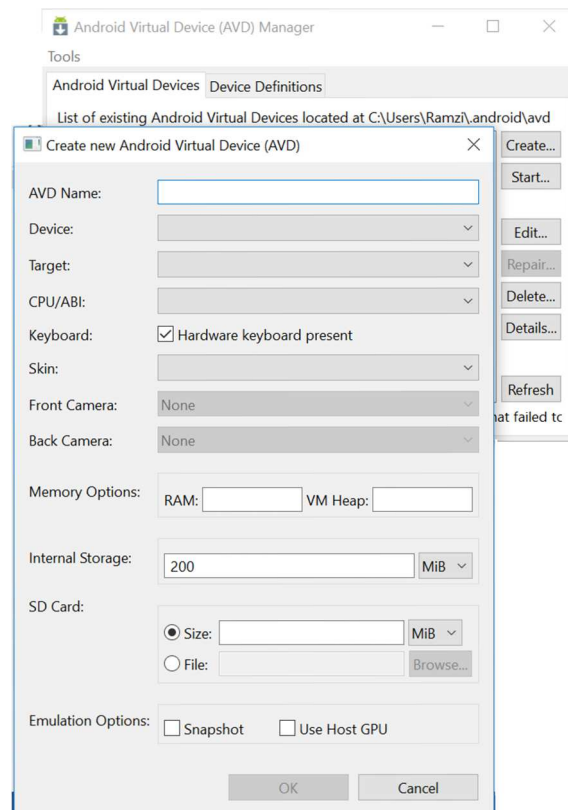
Vous devez lancer le SDK Manager pour installer les paquetages dont vous avez besoin. Dans la documentation¹ d'Apache Cordova ils spécifient que les versions d'Android dont la part du marché est inférieure à 5% ne sont pas prises en compte. Ils préconisent l'installation de la version 6.X.X d'Android (API 25) et une version du compilateur (Andoird SDK Build-tools)

¹ <https://cordova.apache.org/docs/fr/latest/guide/platforms/android/>

supérieure à 19.1.0. En plus il faut vérifier que dans la partie Extras le « Google USBDriver » est coché.



Il faut ensuite créer un émulateur via le « AVD Manager ». Il suffit de lancer ce dernier et de créer un ou plusieurs dispositifs mobiles virtuels pour tester vos applications.



Tester l'environnement

Pour tester votre environnement après installation le plus simple est de créer un premier projet de test. Pour mieux organiser votre travail vous pouvez créer un répertoire pour tous vos projets Cordova nommé « ProjetsCordova ». Dedans vous pouvez créer vos divers projets.

Pour créer votre premier projet de test, il suffit de lancer l'invite de commandes et de se situer dans le répertoire « ProjetsCordova ».

```
D:\Developpement>cd ProjetsCordova
```

Ensuite, il faut créer un nouveau projet. Apache Cordova offre la commande « create » qui accepte jusqu'à trois arguments : le nom du projet, le nom de domaine et le nom de l'application.

Le premier paramètre, nom du projet, est utilisé par Cordova pour créer un répertoire qui porte ce nom (le répertoire ne doit pas exister auparavant). Le deuxième paramètre est un nom de domaine style inversé. Il est optionnel mais devient nécessaire si nous souhaitons spécifier le troisième paramètre qui est le nom de l'application. Le nom de domaine est similaire au niveau forme aux noms des domaines sur internet sauf qu'il est inversé. Voici un exemple : « org.alecso.test ». Finalement, le troisième paramètre est le nom de l'application. Ce nom va s'afficher au-dessous de l'application une fois installée sur un dispositif mobile. Et c'est le nom sous lequel l'application va apparaître dans les magasins d'applications mobiles. Ce paramètre est optionnel, mais pour l'utiliser il faut auparavant préciser les deux premiers paramètres. Si vous avez omis l'un des deux paramètres optionnels vous pouvez les spécifier après dans le fichier « config.xml ».

Voici un exemple de commande pour créer une application de test :

```
D:\Developpement\ProjetsCordova>cordova create test org.test ApplicationDeTest
```

Pour pouvoir compiler cette application de test pour l'environnement Android, il faut ajouter cet environnement au projet. Pour faire ceci dans un premier temps il faut se positionner dans le répertoire du projet fraîchement créé.

```
D:\Developpement\ProjetsCordova>cd test
```

A partir de ce moment il suffit d'ajouter le ou les environnements cibles souhaités via la commande « cordova platform add ».

```
D:\Developpement\ProjetsCordova\test>cordova platform add android
```

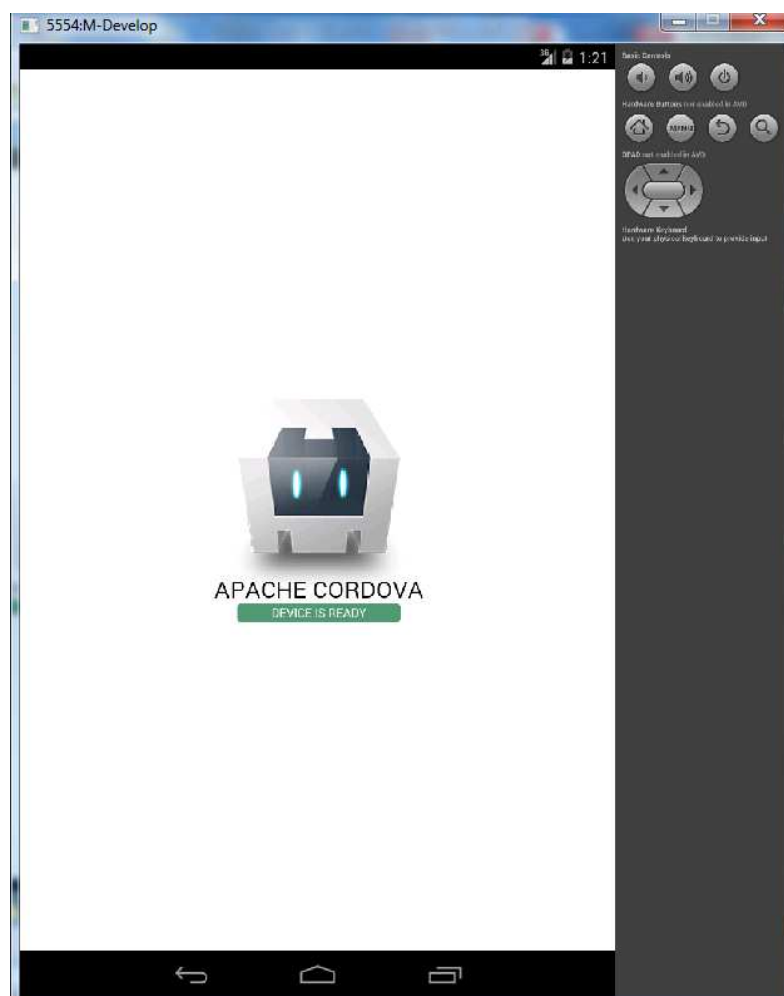
Maintenant il est possible de compiler l'application créée par défaut par « Cordova » pour l'environnement « Android ». Il suffit d'utiliser la commande « cordova build » puisque c'est le seul environnement que nous avons ajouté. S'il y avait plusieurs environnements et que nous souhaitons juste compiler pour « Android » nous pouvons utiliser la commande « cordova build android ».

```
D:\Developpement\ProjetsCordova\test>cordova build
```

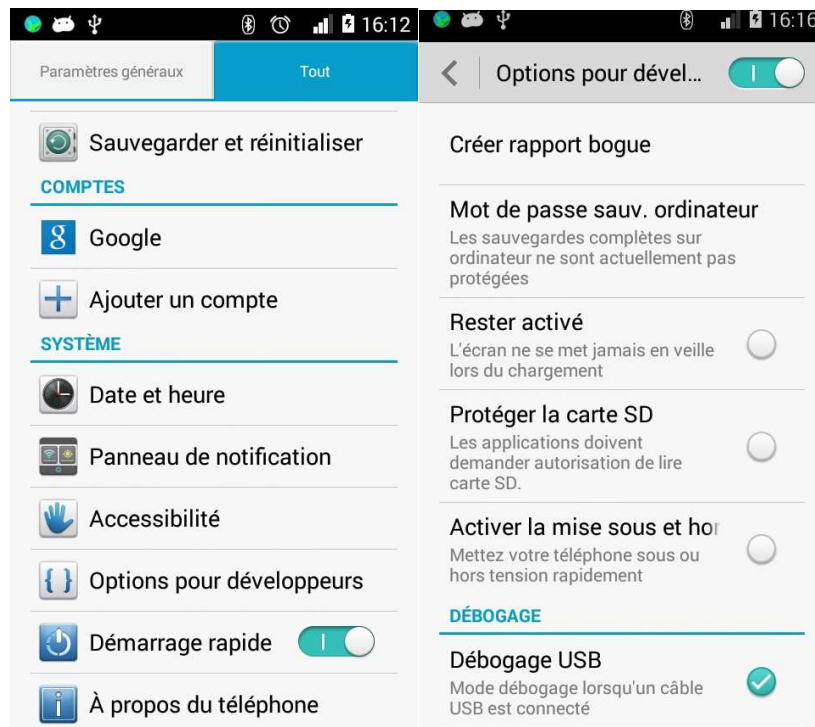
Il ne reste qu'à tester votre application sur l'émulateur via la commande « cordova emulate android ».

```
D:\Developpement\ProjetsCordova\test>cordova emulate android
```

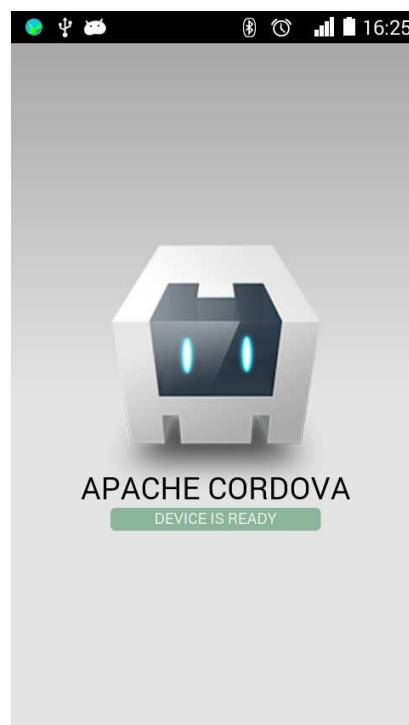
Cette commande va démarrer un dispositif virtuel, déployer l'application et l'exécuter. Si vous voulez tester l'application sur un dispositif virtuel particulier parmi votre liste de dispositifs mobiles il suffit d'utiliser l'option « --target » suivie par le nom du dispositif virtuel (exemple : cordova emulate android --target MonEmulateur)



Il est également possible de tester directement l'application sur votre dispositif mobile. Il faut en fait activer le mode débogage au niveau du dispositif dans un premier temps. Cette option se trouve dans le menu de configuration intitulé « options pour développeurs ».



Ensuite il suffit de connecter votre dispositif mobile à votre ordinateur via un câble USB. Après il faut lancer la commande « cordova run » pour installer et lancer l'application sur votre dispositif mobile connecté.



Structure d'un projet Apache Cordova

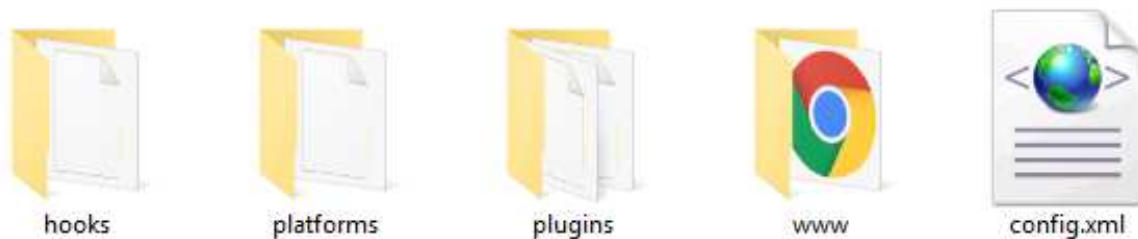
Introduction

Apache Cordova permet de créer des applications mobiles hybrides. Ces applications sont développées en utilisant les langages du Web et compilées pour un ou plusieurs environnements mobiles.

La création d'un projet Apache Cordova via la commande « cordova create » entraîne la création d'un répertoire qui porte le nom du projet. Celui-ci contient toute une arborescence et un ensemble de fichiers qui sont utilisés par Apache Cordova pour créer l'application mobile hybride.

Structure d'un projet Cordova

La création d'un nouveau projet Apache Cordova va donner naissance à une arborescence dont la racine est le répertoire qui porte le nom du projet. Celui-ci à son tour contient initialement quatre sous-répertoires : `www`, `plugins`, `platforms` et `hooks`. Egalement, dans ce répertoire racine on va trouver un fichier intitulé « `config.xml` » qui permet de configurer l'application mobile.



Répertoire `www`

Le développement de l'application mobile se fait essentiellement au sein du répertoire « `www` ». En effet, celui-ci est destiné à accueillir le code source de l'application. Il contient par défaut trois sous-répertoires : `css`, `img` et `js`. En plus, il contient le fichier « `index.html` » qui est par défaut le point d'entrée de l'application. En effet, c'est lui qui sera exécuté par le Webview lorsque l'application est lancée.



Le répertoire « js » est destiné à accueillir tous les fichiers JavaScript relatifs à l'application mobile. Par défaut, il contient le fichier « index.js » qui est invoqué dans le fichier « index.html ».

Le répertoire « img » est destiné à accueillir toutes les images utilisées dans l'application mobile. Par défaut, il contient le fichier « logo.png » qui contient le logo d'apache cordové utilisé dans l'application créée par défaut lors de la création d'un nouveau projet.

Le répertoire « css » est destiné à accueillir tous les fichiers de style css. Il contient par défaut le fichier « index.css » utilisé dans le fichier « index.html » de l'application générée par défaut. Toutes les feuilles de styles de l'application mobile hybrides doivent résider dans ce répertoire.

Respecter cette arborescence permet d'avoir un code source organisé et facile à maintenir. L'emplacement des fichiers est ainsi standardisé ainsi que les noms des fichiers principaux (index.html, index.js et index.css).

D'autres répertoires et fichiers peuvent être créés et ajoutés par la suite par le développeur. C'est le cas du répertoire « res » par exemple qui va contenir les icônes et les splash screens. C'est également le cas des fichiers de certains Frameworks qui peuvent être utilisés dans le projet à l'instar des fichiers du framework « JQuery Mobile ».

Répertoire plugins

Pour pouvoir profiter pleinement des fonctionnalités natives d'un dispositif mobile il faut utiliser des plugins. Selon l'application mobile hybride il se peut que le développeur ait besoin d'un plugin pour utiliser une fonctionnalité supportée par celui-ci. A ce moment il doit utiliser la commande « cordova plugin add » suivie par le nom du plugin. L'exécution de cette commande va permettre l'ajout du plugin au projet en cours dans le répertoire réservé aux plugins intitulé « plugins ».



Le développeur ne doit pas toucher au contenu du répertoire plugin. En effet, la gestion des plugins (tels que l'ajout ou la suppression) doit se faire à travers les commandes Cordova.

Répertoire platforms

Une application mobile hybride a la particularité de pouvoir être compilée pour plusieurs plateformes mobiles à partir d'un même code source. Toutefois, pour pouvoir compiler l'application pour un environnement mobile particulier il faut tout d'abord installer son SDK. Puis, il faut ajouter le ou les plateformes ciblées via la commande « `cordova platform add` ». Celle-ci va permettre d'ajouter les fichiers nécessaires pour pouvoir compiler l'application pour l'environnement souhaité. Ces fichiers résident dans un répertoire qui porte le nom de l'environnement sous le répertoire « `platforms` ».



Ce répertoire peut être utile au développeur dans plusieurs cas. Les deux cas les plus fréquents sont la récupération de l'exécutable et la récupération du code source. Pour le premier cas, si le développeur souhaite récupérer l'exécutable (fichier d'extension APK pour le cas de la plateforme Android) pour le tester, pour le partager ou pour le publier. Le deuxième cas c'est lorsqu'il n'est pas possible de compiler le code source sur la même machine. C'est le cas si le développement s'effectue sur un PC et que la plateforme cible est « iOS ». Dans ce cas, il faut récupérer le code source et le compiler sur un ordinateur de la marque Apple avec le système d'exploitation « Mac OS X » avec l'environnement de développement « Xcode ».

Répertoire hooks

Ce répertoire est destiné à accueillir des scripts appelés « hooks ». Il s'agit de script qui sont développés par les développeurs des applications mobiles ou des développeurs de plugins pour personnaliser les commandes Cordova. Par exemple, vous pouvez définir un « hook » qui permet de vérifier systématiquement le formatage du code dans vos fichiers JavaScript avant chaque appel à la commande « `cordova build` » pour générer l'application. Pour automatiser l'exécution de ce script il suffit d'indiquer son type (Exemples : `before_build`, `after_build`, `before_run` et `after_run`).

Cependant, l'usage du répertoire « hooks » pour créer ce type de script est devenu obsolète du moment qu'il est possible de faire la même chose dans les fichiers « conig.xml » et « plugin.xml ». Il existe encore pour des raisons de compatibilité avec des anciens plugins qui utilisent ce répertoire.

Modèle de programmation

Introduction

Une application mobile hybride est une application dont le code source est écrit en utilisant les langages du Web : HTML 5, CSS 3 et JavaScript. Ce code source réside dans le fichier « www » sous le dossier racine du projet. Celui-ci contient à son tour trois répertoires permettant d'organiser les fichiers contenant le code source de l'application : css, js et img. Pour le code HTML 5 il est généralement écrit dans un seul fichier « index.html » en suivant le principe du document unique contenant des pages multiples.

Lors de la création d'un nouveau projet Apache Cordova le code source d'une application minimale est créé. Ce code peut servir de modèle à suivre dans le développement d'autres applications mobiles hybrides.

index.html

Lorsque une application mobile hybride est compilée et exécutée, par défaut le premier fichier chargé par le Webview est « index.html ». Celui-ci donne un certain nombre d'informations sur la façon avec laquelle la page doit être affichée, les feuilles de styles à appliquer, les composants de l'interface graphique à présenter à l'utilisateur et le code JavaScript à exécuter.

Le fichier « index.html » créé par défaut contient au niveau de la balise « head » le code suivant :

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self' data: gap:
https://ssl.gstatic.com 'unsafe-eval'; style-src 'self' 'unsafe-inline'; media-src *">
<meta name="format-detection" content="telephone=no">
<meta name="msapplication-tap-highlight" content="no">
<meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-
scale=1, width=device-width">
<link rel="stylesheet" type="text/css" href="css/index.css">
<title>Hello World</title>
```

La première balise « meta » est relative à des configurations de la politique de sécurité. Les détails de configuration sont donnés à l'adresse suivante : <https://github.com/apache/cordova-plugin-whitelist/blob/master/README.md#content-security-policy>

La deuxième balise « meta » nommée « format-detection » permet de spécifier si les numéros de téléphones dans la page HTML doivent être détectés et présentés comme un lien permettant de lancer l'appel ou non.

La troisième balise « meta » nommée « viewport » permet de paramétrer l'affichage dans le Viewport. En particulier il est possible de paramétrer la largeur de la page pour correspondre à la largeur du dispositif mobile.

La quatrième balise est « link » qui permet de préciser l'emplacement du fichier de style CSS qui sera utilisé par le document html. Sans surprises l'emplacement précisé est le répertoire « css » et le nom du fichier est « index.css ».

La dernière balise est « title » qui permet de spécifier le titre du document HTML en cas d'affichage par un navigateur.

Au niveau de la partie « body » le fichier « index.html » contient par défaut le code suivant :

```
<div class="app">
  <h1>Apache Cordova</h1>
  <div id="deviceready" class="blink">
    <p class="event listening">Connecting to Device</p>
    <p class="event received">Device is Ready</p>
  </div>
</div>
<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="js/index.js"></script>
```

Le contenu présenté est structuré dans des balises « div ». A la fin de la partie « body » contient deux balises « script ». La première balise fait référence au fichier « cordova.js » qui est ajouté automatiquement par Apache Cordova et qui permet d'assurer la communication avec la partie native du dispositif mobile. La deuxième balise fait référence au fichier « index.js » qui réside dans le répertoire « js » et qui contient le code JavaScript spécifique à l'application.

index.js

L'objectif du fichier « index.js » est d'héberger le code source JavaScript qui décrit le comportement applicatif de l'application mobile hybride. Le code source de l'application par défaut explique la démarche à suivre.

Le fichier comporte la déclaration d'un objet « app » qui contient un certain nombre de fonctions. Après cette déclaration il y a une instruction d'invocation de la fonction **initialize()** associée à cet objet « app ».

```
var app = {  
    // Application Constructor  
    initialize: function() {  
        this.bindEvents();  
    },  
    ...  
}  
app.initialize();
```

La fonction `initialize()` ne fait qu'appeler une autre fonction du même objet appelée `bindEvents()`. La raison d'être de la fonction `initialize()` n'est pas évident dans le cas de l'application par défaut. Toutefois, c'est là où il faut faire les initialisations nécessaires. Vu que le code par défaut est simple alors il n'y a pas de variables à initialiser. Ainsi, il n'y a qu'un appel à une fonction qui va associer des traitements à des événements : c'est la fonction `bindEvents()`.

```
bindEvents: function() {  
    document.addEventListener('deviceready', this.onDeviceReady, false);  
},
```

La fonction `bindEvents()` ajoute un écouteur d'événement à l'objet `document` qui représente le document HTML. L'événement surveillé est un événement Cordova appelé « `deviceready` ». Celui-là se déclenche lorsque le dispositif est prêt pour exécuter l'application. Ceci permet d'attendre que toutes les ressources, telles que les plugins, sont chargées et prêtes pour l'utilisation.

Lorsque l'événement « `deviceready` » se produit une fonction Callback intitulée `onDeviceReady()` se déclenche. C'est au programmeur de décider ce qu'il souhaite faire à ce moment.

Le modèle de programmation suivi est un modèle événementiel. Il s'agit d'ajouter des écouteurs d'événements à des éléments du document HTML et d'associer à ces événements des fonctions Callback pour déclencher des traitements souhaités.

Il existe deux types d'événements. Tout d'abord les événements JavaScript classiques et les événements Cordova.

Parmi les événements JavaScript Classique nous citons (liste non exhaustive) :

- click,
- dblclick,
- mouseover,
- mouseout,
- mousemove,
- focus,
- blur,
- change,
- input,
- select,
- submit
- reset.

En ce qui concerne les événements Cordova, nous avons dix événements :

- deviceready (fin de chargement de Cordova),
- pause (application qui passe en arrière-plan),
- resume (application qui retourne au premier plan),
- backbutton (l'appui sur le bouton de retour),
- menubutton (l'appui sur le bouton menu),
- searchbutton (l'appui sur le bouton recherche),
- startcallbutton (l'appui sur le bouton d'appel),
- endcallbutton (l'appui sur le bouton de fin d'appel),
- volumedownbutton (l'appui sur le bouton de diminution du volume),
- volumeupbutton (l'appui sur le bouton d'augmentation du volume).

Il faut noter que certains boutons son spécifiques à certain types de dispositifs mobiles.

Pour associer un événement à un élément dans le DOM il suffit d'utiliser la fonction **addEventListener()**. Celle-ci prend trois arguments. Le premier c'est le nom de l'événement. Le second, c'est le nom de la fonction callback à exécuter. Le troisième, qui est **optionnel**, est un booléen qui indique si l'on souhaite utiliser la phase de capture ou bien celle de

bouillonnement. Par défaut, la valeur de ce booléen est **false** ce qui correspond à la phase de capture.

Pour expliquer le principe des phases capture et bouillonnement prenons pour exemple le code suivant :

```
<div id="div1">
  <div id="div2">Mon texte</div>
</div>
<script>
  var d1=document.getElementById("div1");
  var d2=document.getElementById("div2");
  d1.addEventListener("click",f1,true);
  d2.addEventListener("click",f2,true);
  function f1(){
    alert("div1");
  };
  function f2(){
    alert("div2");
  };
</script>
```

Ici nous avons un texte dans un div dont l'id est « div2 » qui est à son tour dans un div dont l'id est « div1 ». Puis dans le code JavaScript nous avons indiqué que suite au click dans « div2 » il faut afficher une alerte qui contient « div2 ». Egalement, nous avons indiqué que suite au click dans le « div1 » il faut afficher une alerte qui contient « div1 ».

L'utilisateur ne va voir que le texte englobé dans les deux « div ». Lorsqu'il click dessus la question est quelle alerte va s'afficher en premier ?

Puisque nous avons spécifié que le mode de propagation est le bouillonnement (la valeur du troisième paramètre est **true**) alors l'élément de plus haut niveau va prendre le dessus. C'est-à-dire que la première alerte va afficher « div1 » puis l'alerte qui va afficher « div2 ». Maintenant si on change le paramètre à **false**, alors la première alerte va afficher « div2 » tandis que la deuxième va afficher « div1 ». C'est ce dernier comportement qui est le plus souvent utilisé et qui est le comportement par défaut si on ne spécifie pas ce troisième paramètre.