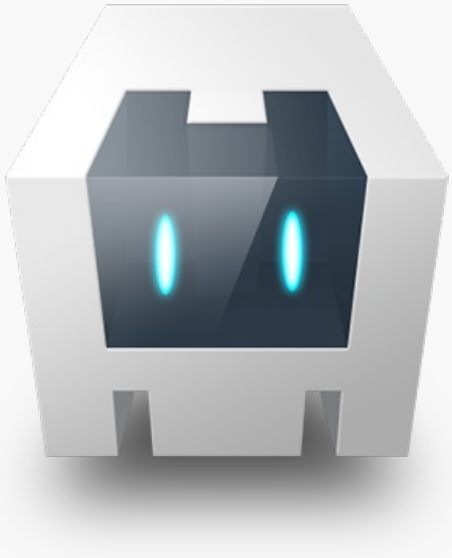
Déploiement Angular multiplateforme avec Cordova



Apache Cordova est un outil développé par Apache permettant de déployer des applications écrites en HTML, CSS et Javascript sur mobile de façon native. Dans ce cours nous verrons comment déployer une application Angular sur Cordova.

Pour suivre ce cours, il faut savoir ouvrir un terminal et entrer quelques commandes :P

I. Installation

1) Cordova

L'installation de Cordova est assez simple. Il suffit d'utiliser npm.

```
npm install -g cordova
```

Vous pouvez vérifier que tout fonctionne bien en vérifiant la version de Cordova installée :

```
cordova -v
```

2) Créer un projet Cordova

Créons un dossier pour notre application : *cordova-heroes*.

Dans ce cours nous diviserons notre projet en deux sous-dossiers : la partie Angular et la partie Cordova. Nous nous baserons sur une application Angular déjà existante, développée dans ce tutoriel et ce tutoriel. Commençons par installer la partie Cordova :

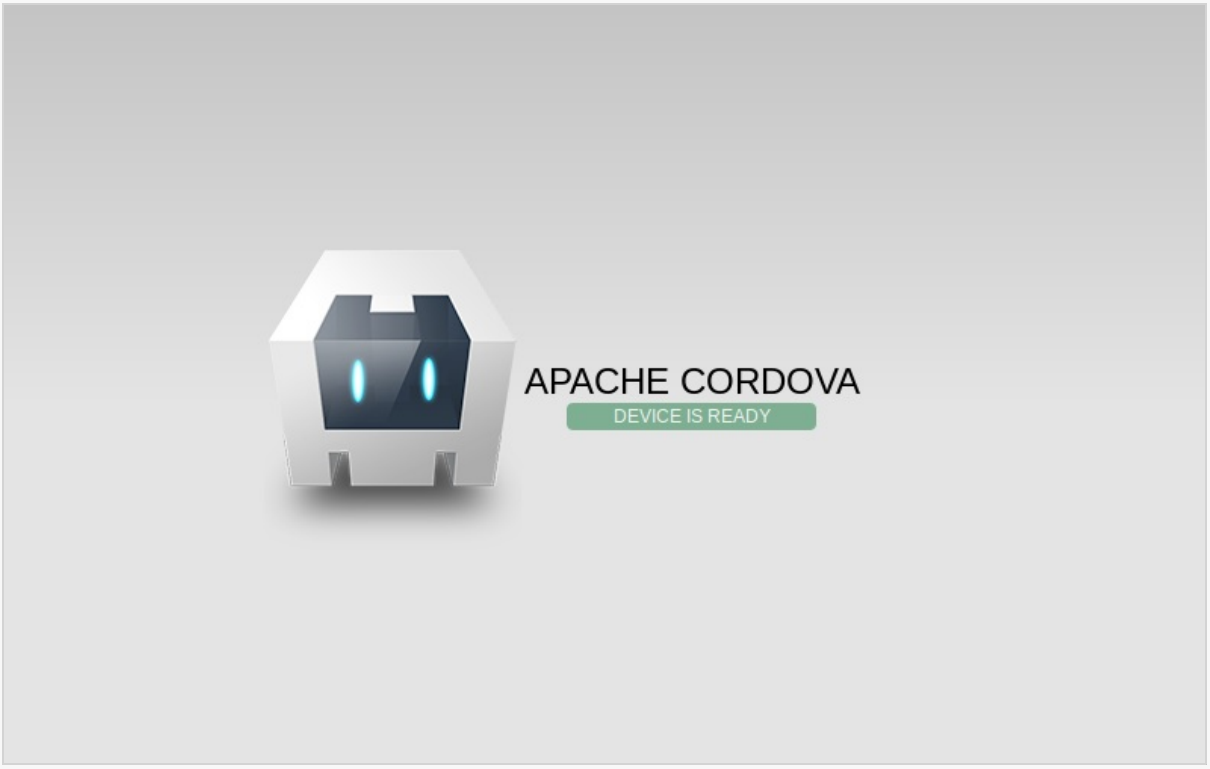
```
cordova create cordova-heroes-app com.acme.app "CordovaHeroesApp"
```

Cette commande crée notre sous-dossier *cordova-heroes-app* contenant notre projet Cordova. Nous ne nous attarderons pas sur sa structure pour l'instant.

Testons que celui-ci fonctionne bien en la lançant dans notre navigateur. Pour cela, nous devons ajouter à Cordova la plateforme browser et la lancer :

```
cordova platform add browser
cordova run browser
```

Nous devrions voir notre navigateur se lancer et afficher la page par défaut de Cordova :



Bien, Cordova fonctionne. Incorporons maintenant notre projet Angular.

3) Incorporer un projet Angular

Pour ce cours, nous partirons d'un projet déjà existant mais il est bien sûr possible d'installer une nouvelle application Angular.

Revenons à la racine du projet et clonons le projet tour-of-heroes.

```
cd ..
git clone https://github.com/Cevantime/tour-of-heroes-typescript.git tour-of-heroes
```

Pour avoir une idée du fonctionnement réel, nous nous baserons sur la branche angular-only qui est un peu avancée :

```
git checkout --track origin/angular-only
```

Pour rendre notre projet Angular compatible avec Cordova, il n'y a presque rien à faire :

- incluez simplement le script de Cordova dans le fichier src/index.html de cette façon :

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>TourOfHeroes</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
   <script type="text/javascript" src="cordova.js"></script>
</head>

<body>
  <app-root></app-root>
</body>

</html>
```

- Nous devons aussi modifier notre fichier main.ts comme suit :

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

const onReady = () => {
  platformBrowserDynamic().bootstrapModule(AppModule)
    .catch(err => console.error(err));
};

document.addEventListener(window.hasOwnProperty('cordova') ? 'deviceready' : 'DOMC
ontentLoaded', onReady, false);
```

Il ne reste plus qu'à déployer notre application Angular dans le dossier www de notre application Cordova !

```
npm install
ng build --prod --base-href . --output-path ../cordova-heroes-app/www
```

Relançons notre projet Cordova dans notre navigateur :

```
cd ../cordova-heroes-app
cordova run browser
```

4) Nettoyer derrière soi !

Il subsiste un léger problème. Nous incluons dans notre index un fichier cordova.js qui n'existe que dans le contexte Cordova. Si nous souhaitons continuer à développer dans l'écosystème Angular (ce qui semble une meilleure idée que réexporter notre projet à chaque fois, notre navigateur loguera une erreur en tentant de charger un script qui n'existe pas. Ce n'est pas critique, mais tout de même...

Pour éviter ce problème, nous commentons notre inclusion de fichier et la décommenterons uniquement dans le contexte de Cordova. Pour cela, nous allons créer un fichier hook qui s'exécutera uniquement avant le déploiement de notre application via Cordova. Ce fichier doit être situé dans le dossier *cordova-heroes-app/hooks/before_prepare.* Nous le nommerons

01_switch_configuration.js avec le contenu suivant :

```
#!/usr/bin/env node

var fs = require('fs');
var path = require('path');

var rootdir = process.argv[2];

if (rootdir) {

    var filestoreplace = ["www/index.html"];

    filestoreplace.forEach(function(val, index, array) {
        var fullfilename = path.join(rootdir, val);
        if (fs.existsSync(fullfilename)) {
            var data = fs.readFileSync(fullfilename, 'utf8');
            data = data.replace(new RegExp(/<!-- web-version-config-on -->/, "g"), '<!-
- web-version-config-off');
            data = data.replace(new RegExp(/<!-- end-web-version-config-on -->/, "g"),
'end-web-version-config-off -->');
            data = data.replace(new RegExp(/<!-- cordova-version-config-off/, "g"), '<!
-- cordova-version-config-on -->');
            data = data.replace(new RegExp(/end-cordova-version-config-off -->/, "g"),
'<!-- end-cordova-version-config-on -->');
            fs.writeFileSync(fullfilename, data, 'utf8');
        } else {
            console.log("missing: " + fullfilename);
        }
    });

}
```

Ce script décommentera notre inclusion, laquelle *doit* être commentée de cette manière dans index.html :

```
<!-- cordova-version-config-off
<script type="text/javascript" src="cordova.js"></script>
end-cordova-version-config-off -->
```

Il existe un problème avec le router qui empêche la touche "back" de fonctionner sur Cordova. Pour contourner le probème, la meilleure solution est d'inclure plutôt ceci :

```
 <!-- cordova-version-config-off
  <script>
  window.addEventListener = function () {
    (window.EventTarget || Window).prototype.addEventListener.apply(this, arguments);
  };
  window.removeEventListener = function () {
    (window.EventTarget || Window).prototype.removeEventListener.apply(this, arguments);
  };
  document.addEventListener = function () {
    (window.EventTarget || Document).prototype.addEventListener.apply(this, arguments);
  };
  document.removeEventListener = function () {
    (window.EventTarget || Document).prototype.removeEventListener.apply(this, arguments
);
  };
</script>
  <script type="text/javascript" src="cordova.js"></script>
  end-cordova-version-config-off -->
```

II. Améliorer le workflow pour travailler confortablement

Pour l'instant, nous sommes obligés de lancer une commande qui prend un peu de temps pour pouvoir tester notre application sur Cordova. Pour une workflow fluide, nous sommes pour l'instant obligés de passer par `ng serve --open` et de tester sur Desktop dans le contexte d'Angular...

Ce n'est pas l'idéal... Essayons de faire mieux.

1) Surveillance des changements

Pour cela, nous allons faire en sorte de lancer automatiquement la commande de déploiement de fichiers depuis Angular vers Cordova chaque fois que notre projet Angular est modifié. Pour cela, il suffit de lancer cette commande :

```
ng build --base-href . --output-path ../cordova-heroes-app/www --watch
```

On ajoute simplement `--base-href` et surtout `--watch` pour surveiller l'activité sur notre projet Angular et mettre à jour directement les fichiers modifiées dans le dossier www de cordova-heores-app.

2) Redéploiement et synchronisation

Pas mal mais insuffisant, car il faut aussi que les palteformes que nous avons ajoutées (browser en l'occurrence) soient mises à jour lorsque des changements se produisent dans le dossier cordova-heroes-app/www. Pour cela, nous allons installer un plugin appelé cordova-plugin-browsersync:

```
cordova plugin add cordova-plugin-browsersync
```

Ce plugin n'est pas à jour, aussi allons-nous le modifier pour qu'il fonctionne correctement dans le contexte de notre application.

- Allez dans *plugins/cordova-plugin-browsersync/pluginHook.js* et modifiez la ligne 47 comme ceci :
  ```
  if (event === 'change' && file !== 'www/index.html') {
  ```
- Puis ligne 53, enlevez le `.raw` pour obtenir cette ligne :
  ```
  context.cordova.prepare().then(function() {
  ```
- Maintenant, dans le fichier *plugins/cordova-plugin-browsersync/utils/Patcher.js*, modifiez la ligne 13 ainsi :
  ```
      android: 'app/src/main/assets/www',
  ```
- Et la ligne 19 ainsi :
  ```
      android: 'app/src/main/res/xml',
  ```

Maintenant, nous pouvons tester notre application en lançant :

```
cordova run browser -- --live-reload
```

Si vous testez dans votre navigateur à l'adresse *http://localhost:3000/platforms/browser/www/*, vous constaterez que toute modification dans Angular déclenche une mise à jour dans Cordova !

Ce qui est intéressant avec ce plugin, c'est qu'il permet aussi de se synchroniser au niveau des appareils android.

le livereload ios demanderait une modification trop profonde du plugin. Peut-être celui-ci sera-t-il mis à jour mais rien n'est moins sûr

III. Déploiement sur d'autres plateformes

Android

IOS

Windows Phone