INGÉNIEURS EN SCIENCES INFORMATIQUES

ADAPTATION DES INTERFACES A L'ENVIRONNEMENT

Tutoriel Ionic framework

Tutoriel et présentation du framework IONIC : technologie permettant la réalisation d'applications mobiles hybrides.

Auteur :

Salah BENNOUR (SI5-IHM)

Enseignants :

Anne-Marie DERY

Christian BREL

SOMMAIRE

1 Introduction

1.1 Qu'est-ce que IONIC framework ?

Ionic est un framework de développement qui permet de créer des applications hybrides en HTML5, CSS, Javascript.

Il est basé sur des frameworks/technologies qui ont fait leurs preuves telles que AngularJS et Apache Cordova.

Comment Ionic utlise AngularJS et Apache Cordova ?

Ionic est développé avec *AngularJS*, il est utilisé pour l'implémentation de la partie applicative web (backend). Apache Cordova permet, quant à lui, la construction des applications natives. C'est un pont de développement qui permet d'encapsuler du code client Web (ici *Ionic*/*AngularJS*) dans une application native telle que Android, IOS ou encore Windows Phone. Il permet par ailleurs d'avoir certains accès aux fonctionnalités natives des appareils tel que la caméra ou l'accéléromètre.

1.2 Avantages

Le *Ionic Framework* dispose de plusieurs avantages :

- Le cross-plateforme. A partir d'un socle de développement pouvoir déployé (en une seule ligne de commande) une application sur plusieurs environnements (IOS, Android, Windows Phone). Cela permet de réduire les couts et délais de développement pour proposer plusieurs applications natives pour les différents systèmes d'exploitation mobile.
- Il est basé sur *Apache Cordova* et *AngularJS* qui disposent d'une grande communauté et par conséquence de nombreux plugins de développement.
- En quelques lignes de commande (trois) il est possible de créer une application multiplateforme.
- Les documentations des technologies *AngularJS* et *Apache Cordova* sont complètes.
- Le Framework s'appuie sur *AngularJS*, gage de stabilité puisque le Framework est maintenant largement utilisé et surtout développé par Google.
- Le déploiement de l'application vers un appareil mobile connecté se fait en une ligne de code (agréable pour les tests).
- La prise en main des technologies est relativement rapide, même pour des développeurs ayant peu d'expérience avec les technologies Web.

1.3 Inconvénient

L'appréhension de l'architecture MVC (Modèle Vue Controller) proposé par *AngularJS* et qui est la structure de l'application, peut-être un inconvénient pour les développeurs peu à l'aise avec celle-ci.

2 Dépendances

Comme cité précédemment *Ionic* intègre des frameworks/technologies. Voici la liste des principales :

Gulp qui est un outil destiné à l'organisation et l'exécution des tâches composant un build. Il est utilisé pour l'automatisation des compilations.

Apache Cordova développé par Fondation Apache est utilisé pour la construction des applications natives.

AngularJS, développé par Google, gère la structure de l'application.

Sass qui est langage de génération dynamique de feuilles de style est utilisé pour la gestion de thème.

3 Installation

3.1 Node JS

Comme cité précédemment, *Ionic* s'appui sur la plateforme *Node JS*. Il est donc indispensable de l'installer si vous ne l'avez pas encore.

Vous trouverez, sur le lien suivant, le site officiel ou vous pouvez installer *Node JS* :

https://nodejs.org/en/download/

Remarque : Sous Windows la commande *npm* ne fonctionnera pas sans les variables d'environnements de système JAVA_HOME, ANDROID_HOME et NPM.

Sous Unix, les variables d'environnement JAVA_HOME et ANDROID_HOME sont nécessaires.

3.2 Apache Cordova

Exécuter la suivante commande pour installer *Apache Cordova*. Si vous l'avez déjà passez cette étape :

```
$ npm install -g cordova
```

3.3 Ionic

Installez maintenant *Ionic* avec cette ligne de commande :

```
$ npm install -g ionic
```

3.4 Android SDK

Pour développer une application pour appareils Android il est indispensable de disposer des outils Android tel que le SDK Android. Si l'on souhaite cibler l'application pour d'autres systèmes d'exploitations, il est là aussi indispensable de disposer du SDK de la plateforme visée. Ceux-ci serviront à la compilation et à la construction de l'application.

3.5 Emulateur

L'utilisation d'un émulateur pour tester l'application lors de la phase de développement est une option viable. En effet, utiliser les différents terminaux virtuels proposés par un émulateur pour tester, par exemple, son adaptation aux différents écrans est recommandé.

L'émulateur Genymotion est une solution viable (https://www.genymotion.com/).

Une alternative est de lancer l'application sur un serveur web local (localhost) et utiliser les outils de visualisation du navigateur pour simuler un appareil précis.

4 Créer un projet

Une fois les outils installés, et qu'on dispose d'un environnement de travail opérationnel, il est maintenant temps de créer notre projet que l'on nommera 'monProjet'.

Pour cela *Ionic Framework* propose trois Templates prédéfinis.

4.1 Blank

Le premier et le plus basique est la création d'une application vide. Pour cela, tapez la commande suivante dans le terminal :

```
$ ionic start monProjet blank
```

Il peut être intéressant de commencer par un projet vide lorsque l'on a une certaine expérience de la technologie. Pour des débutants, il est conseillé d'utiliser un des deux autres Templates.

4.2 SideMenu

Le second et certainement le plus utile, est la création d'une application avec un menu latéral intégré. Pour cela taper la commande suivante dans le terminal :

```
$ ionic start monProjet blank
```

4.3 Tabs

Enfin, le dernier Template proposé qui est celui par défaut lorsqu'on ne spécifie pas le Template est la création d'une application avec des onglets. Pour cela tapez la commande suivante dans le terminal :

```
$ ionic start monProjet tabs
```

5 Plateformes et déploiement

Ionic est un framework permettant la création d'applications hybrides. De ce fait il est possible de cibler les plateformes (systèmes d'exploitations) que l'on souhaite viser.

5.1 Android

Pour configurer votre application pour les appareils Android. Il suffit d'exécuter la suivante commande à la racine de votre projet :

```
$ ionic platform add android
```

Ensuite, pour compiler puis lancer et déployer votre application sur un appareil connecté ou depuis un émulateur, il suffit d'exécuter la commande suivante :

```
$ ionic run android
```

5.2 IOS

Pour configurer votre application pour les appareils IOS. Il suffit d'exécuter la suivante commande à la racine de votre projet :

```
$ ionic platform add ios
```

Ensuite, pour compiler puis lancer et déployer votre application sur un appareil connecté ou depuis un émulateur, il suffit d'exécuter la commande suivante :

```
$ ionic run ios
```

5.3 Web

Il est par ailleurs possible de déployer l'application sur un serveur local de test. Pour cela exécuter à la racine de votre projet la commande suivante :

```
$ ionic serve
```

Dès lors une page du navigateur s'ouvre à la racine du projet.

6 Architecture MVC

Le projet *Ionic* basé sur *AngularJS* a une architecture MVC. Cette architecture 3-tier est intéressante de part la séparation des données (le modèle), de la gestion de l'interface (la vue) et des actions (le contrôleur).

6.1 Modèle

Le modèle dans le cadre du projet réalisé, le model s'est restreint à un fichier Javascript contenant de simples listes de données statiques. Le contrôleur accédait directement aux données peut les variables.

6.2 Contrôleur

Le contrôleur permet de capturer des données du modèle pour les transmettre à la vue.

```
myApp.controller('MainCtrl', ['$scope', function ($scope) {
  // On capture la donnée du modèle
  // on l'initialise avec une chaine de caractères
  $scope.prenom = 'Salah';
}]);
```

Figure 1 : Exemple contrôleur simple

Dans cet exemple, le contrôleur est nommé 'MainCtrl' dispose d'une variable prénom dont la vue pourra consommer. La donnée prénom aurait tout autant pu être capturée depuis le modèle.

6.3 Vue

La vue permet d'afficher des données renvoyées par le controleur.

```
<div ng-controller="MainCtrl">
   <p>Mon prenom: {{ prenom }}</p>
</div>
```

Figure 2 : Exemple d'une vue simple

Dans cet exemple, la vue est associée au contrôleur 'MainCtrl' grâce à la balise 'ng-controller'. Celle-ci peut dont consommer ses services. Ici on ne récupère le prénom.

Il est tout autant possible de remonter l'information dans l'autre sens ; de la vue vers le contrôleur.

6.4 Routes

Enfin un fichier de routes permet la configuration de la navigation entre les différentes pages. Voici un exemple :

```
$routeProvider.
    when('/', {
        controller:'View1Ctrl',
        templateUrl:'view1.html',
        resolve: { wait: 'WaitInit'}
    }).
    when('/view2', {
        controller:'View2Ctrl',
        templateUrl:'view2.html',
        resolve: { wait: 'WaitInit'}
    }).
    otherwise({redirectTo:'/'});
```

Figure 3 : Exemple de configuration de routes

Lorsque l'utilisateur demande la page '/view2', la route va le rediriger vers la page 'view2.html' et appelé le contrôleur 'View2Ctr' qui lui est associé.

Lorsque qu'une page demandée n'est pas reconnue, c'est alors la fonction 'otherwise' qui se charge de rediriger l'utilisateur vers la page 'view1.html'.

7 Accès aux fonctionnalités natif

7.1 ngCordova

La librairie *ngCordova* soutenu par une communauté active, propose de nombreux plugins permettant l'accéder aux fonctionnalités natif d'un appareil tel que l'accès à la caméra, le TouchID, le Push Notifications et bien d'autres.

Il est difficile de ne pas trouver son besoin dans cette librairie qui regroupe plus de 70 plugins.

Pour télécharger la librairie entière il suffit d'exécuter la suivante commande depuis la racine de votre projet :

```
$ bower install ngCordova
```

Où de télécharger la librairie depuis le site officiel (http://ngcordova.com/) puis importer la librairie dans le projet.

Les plugins sont aussi téléchargeable à l'unité.

7.2 Camera et librairies d'images

Pour télécharger le plugin permettant l'accès natif à la camera, exécuter la suivante commande depuis la racine de votre projet :

```
$ cordova plugin add org.apache.cordova.camera
```

Le code (ci-dessous figure 4) permettant d'implémenter l'accès à la camera.

Lorsque la vue associée au contrôleur 'PhotoCtrl' fait appel à la méthode 'takePhoto', celui-ci peut accéder à la caméra de son mobile.

Le contrôleur se doit d'avoir le paramètre '$cordovaCamera' qui est le service de camera. Celui-ci peut appeler la méthode 'getPicture' avec des options en paramètres.

Les options sont modifiables selon les besoins de l'utilisateur. Dans le cas présent les options permettent de redéfinir la taille de la photo en 100px par 100px. La photo n'est pas sauvegardée dans la librairie de photo ou encore la photo est convertie en JPEG.

```
module.controller('PhotoCtrl', function($scope, $cordovaCamera) {

$scope.takePhoto = function() {
    var options = {
      quality: 50,
      destinationType: Camera.DestinationType.DATA_URL,
      sourceType: Camera.PictureSourceType.CAMERA,
      allowEdit: true,
      encodingType: Camera.EncodingType.JPEG,
      targetWidth: 100,
      targetHeight: 100,
      popoverOptions: CameraPopoverOptions,
      saveToPhotoAlbum: false,
      correctOrientation:true
    };

    $cordovaCamera.getPicture(options).then(function(imageData) {
        // success
    }, function(err) {
      // error
    });

  }
});
```

Figure 4 : Accès camera depuis un contrôleur

Pour accéder à la librairie de photos, il suffit de change dans les options la 'sourceType' par 'Camera.DestinationType.PHOTOLIBRARY'.

7.3 Autres plugins

Il existe bien d'autres plugins permettant l'accès aux fonctionnalités natif d'un appareil. Veuillez consulter la page officiel *ngCordova* pour plus de détails :

http://ngcordova.com/docs/plugins/

8 Adapter l'interface aux différents écrans

8.1 Responsive

Ionic propose des tableaux (ou grilles) avec des colonnes responsives. Celles-ci peuvent être utiles pour reformer l'interface d'une page lorsque la largeur de l'appareil est restreinte.

Voici la liste des propriétés proposées :

Propriétés	Cible
.responsive-sm	Les colonnes passe sur une ligne quand la largeur de l'écran est inférieure à la taille d'un Smartphone en mode paysage
.responsive-md	Les colonnes passe sur une ligne quand la largeur de l'écran est inférieure à la taille d'un Tablette en mode portrait
.responsive-lg	Les colonnes passe sur une ligne quand la largeur de l'écran est inférieure à la taille d'un Tablette en mode paysage

8.2 Media queries

Il est par ailleurs possible d'adapter le contenu d'une page avec des média queries.

Voici la liste des média queries et de leur association avec les appareils :

MEDIA QUERIES	CIBLE
`@media (min-width:320px)`	Smartphones portrait
`@media (min-width:481px)`	Smartphones paysages
`@media (min-width:641px)`	Tablettes portrait
`@media (min-width:961px)`	Tablettes paysage
`@media (min-width:1025px)`	Grande tablettes paysage et PC
`@media (min-width:1281px)`	Haute résolution PC

8.3 Visibilité du menu latéral

Le menu latéral *Ionic* peut-être configurer pour être visible de manière forcé à partir d'une certaines largeur de l'écran. Pour cela, il suffit de définir l'attribut à l'intérieur de la balise de menu.

Par exemple pour forcer la visibilité du menu lorsque l'appareil est au moins une tablette en portrait paysage :

```
expose-aside-when="(min-width:961px)"
```

9 Bibliographie

Documentation Ionic :

- http://ionicframework.com/
- http://forum.ionicframework.com/

Documentation Apache Cordova :

- https://cordova.apache.org/
- http://ngcordova.com/

Documentation AngularJS :

- https://angularjs.org/
- https://openclassrooms.com/courses/developpez-vos-applications-web-avec-angularjs

Documentation media queries :

- http://stackoverflow.com/questions/6370690/media-queries-how-to-target-desktop-tablet-and-mobile

- https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3/mise-en-page-adaptative-avec-les-media-queries

Mise à niveau Javascript :

- https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript

Recherches de solutions :

http://stackoverflow.com/