TUTORIEL IONIC

Ionic est un framework qui va vous permettre de créer des applications mobiles en utilisant des technologies Web. Ionic se base pour cela sur d'autres frameworks / technologies qui ont fait leurs preuves.

- Gulp, pour la partie automatisation des compilations.
- AngularJS, pour la partie front-end avec l'utilisation d'Angular UI Router pour la gestion des stats.
- Apache Cordova, pour la création d'une application fonctionnelle sur mobile.

Globalement ionic offre une série de directive et de service construit par dessus angular qui vous permettront de créer rapidement des éléments d'interface. Il suffit de faire un rapide tour sur la documentation pour voir les différents éléments disponibles.

Ma première application

Avant de pouvoir commencer, il nous faut évidemment commencer par installer l'outil. Pour cela il faut utiliser npm

Cette commande va permettre d'installer Ionic de manière globale et nous donnera accès à la commande ionic. On va d'ailleurs pouvoir utiliser cette dernière pour créer notre premier projet.

Cette commande va permettre de créer un dossier qui va contenir un projet vide. Il est possible de partir d'un template comme tabs ou sidemenu. Si vous souhaitez prévisualiser votre application sur votre navigateur web il vous suffit de taper la commande serve

La Structure du projet

Globalement ionic offre une structure d’un projet web avec le dossier www, où on retrouve la meme structure que angular, aussi on trouve les dossier classic de configuration de l’application comme par example le fichier bower.json pour gerer les dependences ou gulpfile.js pour les tash de compilation.

Le routing

Le routing se base sur [Angular UI Router] du coup il n'y a pas de code propre à ionic.

Par contre le contenu sera automatiquement chargé dans l'élément

`<ion-nav-view></ion-nav-view>`.

```
.config(function($stateProvider, $urlRouterProvider){

    $stateProvider.state('home',{
      url:'/home',
      templateUrl: 'templates/home.html',
      controller: 'HomeCtrl',
      controllerAs: 'movies'
    })

    $stateProvider.state('movie',{
      url:'/movie',
      templateUrl: 'templates/movie.html',
      controller: 'MovieCtrl'
    })

    $stateProvider.state('movies',{
      url:'/movies',
      templateUrl:'templates/movies.html',
      controller: 'MoviesCtrl'
    })

    $stateProvider.state('search',{
      url:'/search',
      templateUrl:'templates/search.html',
      controller: 'SearchCtrl'
    })


    $urlRouterProvider.otherwise('/home')

  });
```

Les controllers

Un example de controller qui utilise un service qui se fait de requettes vers un server nodejs pour avoir les informations des films.

Evidemment pour une premier application on peut stocker les information des films dans un fichier JSON en local, pour tester facilement l'application.

```
angular.module('popcornFlixApp')
  .controller('MoviesCtrl', ['$scope','movieService', function ($scope,movieService) {
    movieService.getMovies().then(
      function (data){
        $scope.movies = data.movies;
      },
      function (msgError) {
        console.log(msgError);
      }
    )
  }]);
```

L'interface de base

Évidemment toutes les applications n'ont pas la même interface, mais la structure principale reste sensiblement la même (une barre en haut avec la flèche retour et le contenu en dessous).

```
<body ng-app="starter">

<ion-nav-bar class="bar-positive">
  <ion-nav-back-button></ion-nav-back-button>
</ion-nav-bar>

<ion-nav-view></ion-nav-view>
```

Si on veut un slide menu le code sera un peu plus complexe, mais on retrouvera toujours les éléments<ion-nav-bar> et <ion-nav-view>

```
<body ng-app="starter">
  <ion-side-menus>
    <ion-side-menu-content>
      <ion-nav-bar class="bar bar-dark">
        <ion-nav-back-button></ion-nav-back-button>
        <ion-nav-buttons side="left">
          <button class="button button-icon button-clear ion-navicon" menu-toggle="left"></button>
        </ion-nav-buttons>
        <ion-nav-buttons side="right">
          <button class="button button-icon button-clear ion-search" menu-toggle="right" ui-sref="search"></button>
        </ion-nav-buttons>
      </ion-nav-bar>
      <ion-nav-view></ion-nav-view>
    </ion-side-menu-content>
    <ion-side-menu side="left">
      <ion-item menu-close ui-sref="home">
        Home
      </ion-item>
      <ion-list>
        <ion-item class="bar bar-dark">
          Tops
        </ion-item>
        <ion-item menu-close ui-sref="movies">
          Movies
        </ion-item>
        <ion-item menu-close ui-sref="movies">
          Movie aleatoire
        </ion-item>
        <ion-item menu-close ui-sref="movies">
          Series
        </ion-item>
      </ion-list>
    </ion-side-menu>
  </ion-side-menus>
```

Le comportement de la navbar est automatisé par le reste de l'application

- Les titres seront insérés à partir de l'attribut view-title qui sera chargé à l'intérieur.
- Le bouton <ion-nav-back-button> sera automatiquement affiché suivant l'état de notre navigation

Un example d’une directive avec une view

Avec les directives qui nous propose ionic, plus les directives ecrit pour nous, le resultat du code dans les views et très simple et facile a comprendre.

Directive <movie>

```
angular.module('starter')
  .directive('movie', [function () {
    return {
      template:
      '<div class="list card">'+
      '<div class="item item-image">' +
      '<a  ui-sref="movie">' +
      '<img lazyload ll-src="http://192.168.1.25:3000/images/{{movie.image}}"/>' +
      '<h2>{{movie.title}}</h2>' +
      '</a>' +
      '</div>'+
      '</div>',
      restrict: 'E',
      replace: true,
      scope : {
        movie: '=movie'
      },
      require: 'lazyload'
    };
  }]);
```

View movies.html

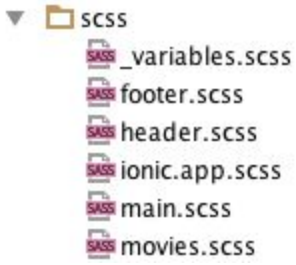
```
<ion-view view-title="Movies">

  <ion-content>
    <ion-nav-back-button></ion-nav-back-button>
      <movie ng-repeat="movie in movies" movie="movie">
      </movie>
  </ion-content>
</ion-view>
```

Le style

Pour personnaliser l'interface un minimum pour que votre application ne ressemble pas à ce qui est proposé par défaut. Pour cela vous pouvez utiliser le CSS qui est fourni ou utiliser SASS qui vous permettra une configuration plus aisée grâce à l'utilisation de variable. Dans ce cas-là il faudra préparer votre projet pour l'utilisation de SASS.

```
garrigos at MacBook-Pro-de-Fernando in ~/Projects/SI5/AdaptationDesInterfaces/PopcornFlix/Mobile (master)
$ ionic setup sass
```

Dossier généré



Example movies.scss

```scss
@import "variables";

section {
  text-align: center;

  > div {
    padding: 10px 10px 10px 10px;
  }

  img {
    width: 30px;
    height: 20px;
    border: 2px solid $background-color;
    border-radius: 10px;

    &:hover {
      border: 2px solid $my-blue;
    }
  }
```

Cette commande aura pour effet de créer un dossier SCSS à la racine qui sera automatiquement observé lorsque vous utiliserez la commande serve.

La compilation

En effet, ionic se repose sur Apache Cordova pour la gestion de la création d'une application mobile. Il sera possible de compiler en utilisant les commandes que l'on connait déjà avec Cordova (ou phonegap) en remplaçant par ionic.

Pour ajouter une plateforme (dans notre cas android)

```
garrigos at MacBook-Pro-de-Fernando in ~/Projects/SI5/AdaptationDesInterfaces/PopcornFlix/Mobile (master)
$ ionic add plataform android
```

Pour ajouter un plugin cordova.

```
garrigos at MacBook-Pro-de-Fernando in ~/Projects/SI5/AdaptationDesInterfaces/PopcornFlix/Mobile (master)
$ ionic plugin add nomplugin
```

Pour build le projet.

```
garrigos at MacBook-Pro-de-Fernando in ~/Projects/SI5/AdaptationDesInterfaces/PopcornFlix/Mobile (master)
$ ionic bluid android
```

Pour l’exporter dans un emulator.

```
garrigos at MacBook-Pro-de-Fernando in ~/Projects/SI5/AdaptationDesInterfaces/PopcornFlix/Mobile (master)
$ ionic emulate android
```

Pour l'exporter dans un telephone android.

```
garrigos at MacBook-Pro-de-Fernando in ~/Projects/SI5/AdaptationDesInterfaces/PopcornFlix/Mobile (master)
$ ionic run android
```