eBook Gratuit

APPRENEZ ionic2

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#ionic2

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ionic2

It is an unofficial and free ionic2 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ionic2.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec ionic2

Remarques

Ionic 2 est une technologie de développement mobile multiplateforme. Ce framework est conçu pour construire des applications mobiles hybrides et peut également être utilisé pour des applications de bureau. C'est une technologie d'écriture unique, exécutée partout. Il utilise les technologies Web telles que JavaScript / Typescript, Angular 2, HTML et CSS (SCSS / LESS). Les applications Ionic2 fonctionnent bien sur `>=android 4.4` , mais vous devez exécuter sur `android 4.1` à `android 4.3` vous devez utiliser Cross-Walk .

Examples

Installation ou configuration

Comme Ionic 2 s'améliore de jour en jour, consultez toujours la **documentation officielle** pour vous tenir au courant des dernières modifications et améliorations.

Prérequis: Vous aurez besoin de NodeJS pour construire des projets Ionic 2. Vous pouvez télécharger et installer le nœud ici et en savoir plus sur npm et les packages utilisés par Ionic 2 ici .

---

1. Installer Ionic 2

Comme Ionic 1, vous pouvez utiliser l'interface Ionic CLI ou l'interface graphique pour créer et tester rapidement des applications directement dans le navigateur. Il a même toutes les fonctionnalités pour travailler avec vos applications Ionic 1, vous n'aurez donc pas besoin de changer quoi que ce soit!

Pour utiliser Ionic 2, installez simplement ionic from npm:

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

Si vous obtenez une erreur EACCES, suivez les instructions ci- dessous pour donner à node les autorisations nécessaires.

2. Créer votre première application

Une fois la CLI installée, exécutez la commande suivante pour démarrer votre première application:

```
$ ionic start MyIonic2Project
```

Le modèle d'onglet est utilisé par défaut, mais vous pouvez choisir un autre modèle en transmettant un indicateur. Par exemple:

```
$ ionic start MyIonic2Project tutorial
$ cd MyIonic2Project
$ npm install
```

Cela utilisera le modèle de tutoriel .

Pour exécuter votre application, `ionic serve -lc` répertoire de vos projets et exécutez `ionic serve -lc` :

```
$ ionic serve -lc
```

Le -l active le rechargement en direct de la page, le -c affiche les journaux de la console. Si vous rencontrez des problèmes lors de la création de votre application, assurez-vous que votre package.json correspond à celui de la base ionic2-app.

Vous pouvez jouer avec votre nouvelle application directement dans le navigateur!

3. Construire vers un périphérique

Vous pouvez également créer votre nouvelle application sur un périphérique physique ou un émulateur de périphérique. Vous aurez besoin de Cordova pour continuer.

Pour installer Cordova, exécutez:

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

Consultez les documents du simulateur iOS pour créer des applications iOS (REMARQUE: vous ne pouvez pas construire sur des appareils iOS ou des émulateurs sur un autre système d'exploitation que OSX), ni sur les documents Genymotion pour créer une application Android.

En cours d'exécution sur un appareil iOS:

Pour construire une application iOS, il est nécessaire de travailler sur un ordinateur OSX, car vous aurez besoin du framework cacao pour pouvoir créer pour ios. Si c'est le cas, vous devez d'abord ajouter la plateforme à Cordova en exécutant le programme. commande suivante:

```
$ ionic cordova platform add ios
```

Vous aurez besoin de Xcode pour compiler sur un appareil iOS.

Enfin, exécutez votre application avec la commande suivante:

```
$ ionic cordova run ios
```

En cours d'exécution sur un appareil Android:

Les étapes pour Android sont presque identiques. Tout d'abord, ajoutez la plate-forme:

```
$ ionic cordova platform add android
```

Installez ensuite le SDK Android qui vous permet de compiler sur un appareil Android. Bien que le SDK Android soit livré avec un émulateur, il est vraiment lent. Genymotion est beaucoup plus rapide. Une fois installé, exécutez simplement la commande suivante:

```
$ ionic cordova run android
```

Et c'est tout! Félicitations pour la construction de votre première application Ionic 2!

Ionic a aussi rechargé en direct. Donc, si vous souhaitez développer votre application et voir les modifications se produire en direct sur l'émulateur / périphérique, vous pouvez le faire en lançant les commandes suivantes:

Pour iOS:

```
$ ionic cordova emulate ios -lcs
```

Attention, sur iOS 9.2.2, le livereload ne fonctionne pas. Si vous souhaitez travailler avec livereload, modifiez le fichier config.xml en ajoutant les éléments suivants:

```
<allow-navigation href="*"/>
```

Puis dans le `<platform name="ios">` :

```
<config-file parent="NSAppTransportSecurity" platform="ios" target="*-Info.plist">
  <dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
  </dict>
</config-file>
```

Pour Android:

```
$ ionic cordova run android -lcs
```

Le `l` signifie live-reload, `c` pour les journaux de la console et `s` pour les journaux du serveur. Cela vous permettra de voir s'il y a des erreurs / avertissements pendant l'exécution.

Construire pour Windows

Si vous souhaitez créer votre projet pour Windows, vous devez travailler sur un ordinateur Windows. Pour commencer, installez la plate-forme Windows sur votre projet ionic2 en exécutant

la commande suivante:

```
$ionic cordova platform add windows
```

Ensuite, lancez simplement la commande suivante:

```
$ionic cordova run windows
```

Pour exécuter dans le navigateur

```
$ionic serve
```

pour le navigateur chrome inspecter le périphérique. (tapez la barre d'adresse du navigateur chrome)

```
chrome://inspect/#devices
```

Lire Démarrer avec ionic2 en ligne: https://riptutorial.com/fr/ionic2/topic/3632/demarrer-avec-ionic2

Chapitre 2: Ajouter une application ionique à la vue ionique

Introduction

La vue ionique est une application mobile que vous devez installer sur votre mobile pour que vous puissiez voir votre application sans créer de fichiers .apk. En partageant l'identifiant de votre application, les autres utilisateurs peuvent également visualiser votre application sur leur mobile en utilisant la vue ionique.

Site: https://view.ionic.io/

Examples

Étapes pour ajouter votre application à la vue ionique

Les étapes suivantes doivent être effectuées dans le fichier app.ionic.io

1. Créez un compte ou connectez-vous à votre compte ionique
2. Cliquez sur "Nouvelle application" dans le tableau de bord et donnez un nom à votre application.

```
I named my app as 'MyIonicApp'
```

3. Dans la section aperçu de cette application nouvellement créée, il y aura un identifiant sous le nom de l'application.

```
MyIonicApp ID is 4c5051c1
```

Les étapes ci-dessous sont effectuées dans l'invite de commande **Node.js**

1. Connectez-vous à votre compte ionique en exécutant

```
$ ionic login
```

2. Racinez votre dossier d'applications.
3. Pour télécharger votre application sur la vue ionique, vous devez d'abord associer votre application à l'ID que vous avez créé sur le site ionique. Exécutez la commande suivante pour lier,

```
$ ionic link [your-app-id]
```

Pour MyIoincApp, la commande sera,

```
$ ionic link 4c5051c1
```

La commande ci-dessus mettra à jour l'identifiant de l'application dans le fichier de configuration de MyIonicApp.

4. Une fois la liaison effectuée, téléchargez l'application en exécutant

```
$ ionic upload
```

Remarque

Une fois le téléchargement réussi, ouvrez la vue ionique de votre mobile pour afficher l'application.

D'autres peuvent voir votre application en soumettant l'identifiant de l'application dans la section "Aperçu d'une application" en mode ionique.

Lire Ajouter une application ionique à la vue ionique en ligne:
https://riptutorial.com/fr/ionic2/topic/10542/ajouter-une-application-ionique-a-la-vue-ionique

Chapitre 3: Angularfire2 avec Ionic2

Introduction

Ici, nous vous montrons comment intégrer AngularFire2 et utiliser cette base de données en temps réel dans notre application ionique.

Examples

Initialisation AngularFire

Tout d'abord, vous devez initialiser les modules angularfire dans votre module d'application comme ceci:

```
  const firebaseConfig = {
  apiKey: 'XXXXXXXXXX',
  authDomain: 'XXXXXXXXXX',
  databaseURL: 'XXXXXXXXXX',
  storageBucket: 'XXXXXXXXXX',
  messagingSenderId: 'XXXXXXXXXX'
};
```

Vous pouvez obtenir cette clé en vous connectant sur firebase et en créant un nouveau projet.

```
imports: [
    AngularFireModule.initializeApp(firebaseConfig),
    AngularFireDatabaseModule,
    AngularFireAuthModule
  ],
```

Utiliser AngularFire2

Une fois que vous l'avez sur votre application, importez-le simplement:

```
import { AngularFireDatabase } from 'angularfire2/database';
constructor (private _af: AngularFireDatabase) {}
```

Avec cette liste observable, vous pouvez accéder à une liste d'éléments sous un chemin, par exemple si vous avez racine / éléments / nourriture, vous pouvez obtenir des articles alimentaires comme ceci:

```
this._af.list('root/items/food');
```

Et vous pouvez simplement mettre un nouvel élément ici et apparaître sur votre base de données Firebase, ou vous pouvez mettre à jour un élément et vous le verrez mettre à jour sur votre base de données. Vous pouvez pousser et mettre à jour comme ceci:

```
this._af.list('root/items/food').push(myItemData);
this._af.list('root/items/food').update(myItem.$key, myNewItemData);
```

Ou vous pouvez même éloigner des objets de votre liste de nourriture:

```
this._af.list('root/items/food').remove(myItem.$key);
```

Lire Angularfire2 avec Ionic2 en ligne: https://riptutorial.com/fr/ionic2/topic/10918/angularfire2-avec-ionic2

Chapitre 4: Composants Ionic2 CSS

Examples

la grille

Le système de grille d'Ionic est basé sur la flexbox, une fonctionnalité CSS prise en charge par tous les périphériques supportés par Ionic. La grille est composée de trois unités-grille, lignes et colonnes. Les colonnes seront développées pour remplir leur ligne et seront redimensionnées pour s'adapter à des colonnes supplémentaires.

Classe	Largeur
largeur-10	dix%
largeur-20	20%
largeur-25	25%
largeur-33	33,3333%
largeur-50	50%
largeur-67	66.6666%
largeur-75	75%
largeur-80	80%
largeur-90	90%

Exemple.

```
    <ion-grid>
      <ion-row>
        <ion-col width-10>This column will take 10% of space</ion-col>
      </ion-row>
    </ion-grid>
```

Cartes

Les cartes sont un excellent moyen d’afficher des éléments de contenu importants et se révèlent rapidement comme un modèle de conception de base pour les applications. Ils sont un excellent moyen de contenir et d'organiser les informations, tout en définissant des attentes prévisibles pour l'utilisateur. Avec un tel contenu à afficher à la fois, et souvent si peu d’écran, les cartes sont rapidement devenues le modèle de choix pour de nombreuses entreprises.

Exemple.

```
<ion-card>
  <ion-card-header>
        Header
  </ion-card-header>
  <ion-card-content>
        The British use the term "header", but the American term "head-shot" the English
simply refuse to adopt.
  </ion-card-content>
</ion-card>
```

Lire Composants Ionic2 CSS en ligne: https://riptutorial.com/fr/ionic2/topic/8011/composants-ionic2-css

Chapitre 5: Configuration et débogage Ionic 2 dans le code Visual Studio

Introduction

Visual Studio est un IDE open source qui fournit des fonctionnalités d'édition et de programmation pour le code. Cet IDE prend en charge de nombreux langages tels que (Ionic, C, C #, AngularJs, TypeScript, Android, etc.). Ces langages peuvent exécuter leur code en ajoutant ses extensions dans VSCode. En utilisant VSCode, nous pouvons exécuter et déboguer le code de différentes langues.

Examples

Installation de VSCode

Tout d'abord, vous devez télécharger et installer le VSCode. Cette dernière version de VSCode est disponible pour téléchargement sur son site Web officiel . Après avoir téléchargé le VSCode, vous devez l'installer et l'ouvrir.

```
Introduction of Extensions in VSCode
```

VSCode est un éditeur ouvert, il fournit donc un éditeur pour toutes les langues, mais pour exécuter un code, vous devez ajouter l'extension pour cette langue particulière. Pour exécuter et éditer votre code ionique, vous devez ajouter l'extension **ionic2-vscode** dans yourVSCode. Dans la partie gauche de l'éditeur VSCode, il y a 5 icônes dans lesquelles l'icône la plus basse est utilisée pour l'extension. Les extensions que vous pouvez obtenir en utilisant la **touche de raccourci (Ctrl + Maj + X)** .

```
Add Extension for Ionic2 in VsCode
```

En appuyant sur **Ctrl + Maj + X,** vous avez montré la partie de l'extension où les **trois** premiers **points** sont affichés **.** Ces points sont connus sous le nom de plus d'icône. Choisissez l'option selon vos besoins, mais pour obtenir toutes les extensions, sélectionnez **Extension recommandée** affichée .iN la liste de toutes les extensions que vous pouvez installer `(ionic2-vscode),npm`

Créez et ajoutez votre projet ionique dans VSCode

VsCode est incapable de créer le projet ionique car il s'agit d'un éditeur de code. Vous pouvez donc créer votre projet ionique par **CLI** ou **cmd** . créer votre projet par la commande ci-dessous

```
$ ionic start appName blank
```

Commande ci-dessus pour créer une application ionique modèle vierge. Ionic2 fournit trois types de modèles **vierges, des onglets et des menus latéraux** . Alors. vous pouvez remplacer le modèle vierge par deux autres modèles selon vos besoins.

Maintenant, votre projet ionique a été créé. Ainsi, vous pouvez ajouter votre projet dans VSCode pour le modifier. Pour ajouter votre projet, suivez les points ci-dessous.

1. Allez dans le menu **Fichier** dans VScode.
2. Cliquez sur le menu **Ouvrir un dossier** dans un fichier.
3. Recherchez et ouvrez votre dossier de projet.

   Vous pouvez directement ouvrir le dossier en utilisant la touche de raccourci **ctrl + O** ou **ctrl + k**

`

Exécuter et déboguer votre projet ionique

> Exécuter et déboguer dans Chrome

Pour exécuter le projet ionique, utilisez la commande ci-dessous dans **terminal ou cmd ou CLI**

```
$ ionic serve
```

Pour déboguer le projet ionique, vous devez d'abord ajouter une extension **(Debugger for chrome)** , puis configurer le fichier launch.json comme ceci.

```
{
        "version": "0.2.0",
        "configurations": [

            {
          "name": "Launch in Chrome",
          "type": "chrome",
          "request": "launch",
          "url": "http://localhost:8100",
          "sourceMaps": true,
          "webRoot": "${workspaceRoot}/src"
        }
]
}
```

> ***Exécuter et déboguer dans Android***

Pour Run ionic project dans Android, vous devez ajouter la plate-forme Android par la commande ci-dessous dans le terminal ou cmd ou CLI:

```
$ ionic cordova platform add android
```

Construire Android avec cette commande

```
$ ionic cordova build android
```

Exécuter la commande pour la plateforme Android

```
$ ionic cordova run android
```

Maintenant, votre application s'exécute sur le vrai appareil Android.

Pour déboguer dans un appareil Android, vous devez ajouter l'extension **Cordova ou Android** dans VSCode. et configurez le fichier launch.json comme ceci.

```
{
    "version": "0.2.0",
    "configurations": [
    {
                "name": "Run Android on device",
                "type": "cordova",
                "request": "launch",
                "platform": "android",
                "target": "device",
                "port": 9222,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}",
                "ionicLiveReload": false
            },
            {
                "name": "Run iOS on device",
                "type": "cordova",
                "request": "launch",
                "platform": "ios",
                "target": "device",
                "port": 9220,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}",
                "ionicLiveReload": false
            },
            {
                "name": "Attach to running android on device",
                "type": "cordova",
                "request": "attach",
                "platform": "android",
                "target": "device",
                "port": 9222,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}"
            },
            {
                "name": "Attach to running iOS on device",
                "type": "cordova",
                "request": "attach",
                "platform": "ios",
                "target": "device",
                "port": 9220,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}"
            },
            {
                "name": "Run Android on emulator",
```

```
            "type": "cordova",
            "request": "launch",
            "platform": "android",
            "target": "emulator",
            "port": 9222,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}",
            "ionicLiveReload": false
        },
        {
            "name": "Run iOS on simulator",
            "type": "cordova",
            "request": "launch",
            "platform": "ios",
            "target": "emulator",
            "port": 9220,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}",
            "ionicLiveReload": false
        },
        {
            "name": "Attach to running android on emulator",
            "type": "cordova",
            "request": "attach",
            "platform": "android",
            "target": "emulator",
            "port": 9222,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}"
        },
        {
            "name": "Attach to running iOS on simulator",
            "type": "cordova",
            "request": "attach",
            "platform": "ios",
            "target": "emulator",
            "port": 9220,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}"
        },
        {
            "name": "Serve to the browser (ionic serve)",
            "type": "cordova",
            "request": "launch",
            "platform": "serve",
            "cwd": "${workspaceRoot}",
            "devServerAddress": "localhost",
            "sourceMaps": true,
            "ionicLiveReload": true
        },
        {
            "name": "Simulate Android in browser",
            "type": "cordova",
            "request": "launch",
            "platform": "android",
            "target": "chrome",
            "simulatePort": 8000,
            "livereload": true,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}"
        },
```

```
            {
                "name": "Simulate iOS in browser",
                "type": "cordova",
                "request": "launch",
                "platform": "ios",
                "target": "chrome",
                "simulatePort": 8000,
                "livereload": true,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}"
            }
]
}
```

Après la configuration, suivez les étapes suivantes ou les raccourcis clavier pour le débogage:

1. Aller au menu de **débogage** .
2. Cliquez sur **Démarrer le débogage** .

   ou

```
Short keys
```

- Débogage - F5
- StepOver - F10
- Entrez et sortez - F11
- Arrêtez le débogage - Maj + F5
- Redémarrer le débogage -ctrl + shift_F5

Lire Configuration et débogage Ionic 2 dans le code Visual Studio en ligne: https://riptutorial.com/fr/ionic2/topic/10559/configuration-et-debogage-ionic-2-dans-le-code-visual-studio

Chapitre 6: Connexion sociale avec Angularfire2 / Firebase

Examples

Login Facebook natif avec Angularfire2 / Firebase

app.ts

```
import {Component} from '@angular/core';
import {Platform, ionicBootstrap} from 'ionic-angular';
import {StatusBar} from 'ionic-native';
import {LoginPage} from './pages/login/login';
import {FIREBASE_PROVIDERS, defaultFirebase, AuthMethods, AuthProviders, firebaseAuthConfig}
from 'angularfire2';

@Component({
  template: '<ion-nav [root]="rootPage"></ion-nav>'
})

export class MyApp {

  private rootPage: any;

  constructor(private platform: Platform) {
    this.rootPage = LoginPage;

    platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
      StatusBar.styleDefault();
    });
  }
}

ionicBootstrap(MyApp, [
  FIREBASE_PROVIDERS,
  defaultFirebase({
    apiKey: myAppKey,
    authDomain: 'myapp.firebaseapp.com',
    databaseURL: 'https://myapp.firebaseio.com',
    storageBucket: 'myapp.appspot.com',
  }),
  firebaseAuthConfig({})
]);
```

login.html

```
<ion-header>
  <ion-navbar>
    <ion-title>Home</ion-title>
  </ion-navbar>
</ion-header>
```

```
<ion-content padding class="login">
  <button (click)="facebookLogin()">Login With Facebook</button>
</ion-content>
```

login.ts

```
import {Component} from '@angular/core';
import {Platform} from 'ionic-angular';
import {AngularFire, AuthMethods, AuthProviders} from 'angularfire2';
import {Facebook} from 'ionic-native';

declare let firebase: any; // There is currently an error with the Firebase files, this will
fix it.

@Component({
  templateUrl: 'build/pages/login/login.html'
})
export class LoginPage {

  constructor(private platform: Platform, public af: AngularFire) {

  }

  facebookLogin() {
    Facebook.login(['public_profile', 'email', 'user_friends'])
      .then(success => {
        console.log('Facebook success: ' + JSON.stringify(success));
        let creds =
firebase.auth.FacebookAuthProvider.credential(success.authResponse.accessToken);
        this.af.auth.login(creds, {
          provider: AuthProviders.Facebook,
          method: AuthMethods.OAuthToken,
          remember: 'default',
          scope: ['email']
        }).then(success => {
          console.log('Firebase success: ' + JSON.stringify(success));
        }).catch(error => {
          console.log('Firebase failure: ' + JSON.stringify(error));
        });
      }).catch(error => {
        console.log('Facebook failure: ' + JSON.stringify(error));
      });
  }
}
```

Lire Connexion sociale avec Angularfire2 / Firebase en ligne:
https://riptutorial.com/fr/ionic2/topic/5518/connexion-sociale-avec-angularfire2---firebase

Chapitre 7: Constructeur et OnInit

Introduction

En ce qui concerne ionic2, le `constructor` : en termes simples, nous l'utilisons pour créer des instances de nos plugins, services, etc. Par exemple: Vous avez une page (vue) où vous voulez afficher la liste de tous les étudiants qui contient tous les étudiants (ce fichier est votre fichier de données) ce que vous devez faire est de créer un service dans ce service, vous allez créer une méthode et cliquer sur une requête http.get pour obtenir les données json, alors vous avez besoin de quoi? http faites simplement comme suit:

Examples

Exemple de méthode de service de l'étudiant pour utiliser Http dans un constructeur

```
import {Http} from '@angular/http';
@Injectable()
export class StudentService{
    constructor(public http: Http){}
    getAllStudents(): Observable<Students[]>{
        return this.http.get('assets/students.json')
        .map(res => res.json().data)
        }
    }
```

remarquez le constructeur maintenant si nous voulons utiliser cette méthode de service, nous irons à notre vue / page et:

```
import {StudentService} from './student.service';
import { SocialSharing } from '@ionic-native/social-sharing';
export class HomePage implements OnInit {

  constructor(public _studentService: StudentService, public socialSharing: SocialSharing) {
   }
```

Remarquez à nouveau le constructeur ici, nous créons une instance de StudentService dans constructeur et une chose de plus, nous utilisons le plugin socialSharing pour utiliser ce que nous créons également dans le constructeur.

Méthode ngOnInit pour obtenir la liste des étudiants en visualisation

`OnInit` : c'est vraiment génial dans ionic2 ou on peut dire dans AngularJs2. Avec le même exemple ci-dessus, nous pouvons voir ce qu'est ngOnInit. Donc, vous êtes prêt avec la méthode de service, maintenant dans votre vue / page que vous voulez que les données de la liste des étudiants soient disponibles dès que votre vue va apparaître, cela devrait être la première opération se déroulant automatiquement sur la charge, car la liste devrait être visible. La classe

implémente donc OnInit et vous définissez ngOnInit. Exemple:

ngOnInit exemple pour obtenir la liste des étudiants sur la page / vue

```
export class HomePage implements OnInit {
...
....
constructor(....){}

ngOnInit(){
    this._studentService.getAllStudents().subscribe(
     (students: Students[]) => this.students = students,
    )
```

Lire Constructeur et OnInit en ligne: https://riptutorial.com/fr/ionic2/topic/9907/constructeur-et-oninit

Chapitre 8: Du code à l'App Store - Android

Introduction

Vous trouverez des instructions détaillées sur la préparation et le téléchargement d'une application de production ionique sur Google Play.

Examples

Production prête

Création d'un projet d'application

Lors de la création d'une application Android prête pour l'App Store, il est important, lors de l'utilisation d' `ionic start` , d'ajouter les `--appname|-a` et `--id|-i` utilisés pour Google Play pour identifier votre application à partir d'autres applications.

Si vous démarrez un nouveau projet d'application mobile, vous pouvez utiliser l'exemple ci-dessous.

```
$ ionic start --v2 -a "App Example" -i "com.example.app" -t "tabs"
```

1. Fichier de configuration de l'application

Si vous souhaitez définir ces informations dans une application existante, vous pouvez modifier `config.xml` . Je recommande également ceux qui ont utilisé la commande ci-dessus pour modifier `config.xml` .

Confirmez / modifiez l' `widget id` , le `name` , la `description` et les attributs de l' `author` `widget id` .

Exemple:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<widget id="com.example.app" version="1.0.0" xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
  <name>Example App</name>
  <description>Example app for stackoverflow users</description>
  <author email="admin@example.com" href="http://example.com/">Your name or team</author>
  ...
</widget>
```

2. icône et écran de démarrage

Les types de fichiers pris en charge par les icônes et les images de démarrage sont png, psd ou ai et doivent avoir un nom de fichier correspondant à ce qu'il est `icon` ou `splash` et placé sous le répertoire de ressources à la racine de votre projet. Les dimensions minimales de l'image doivent être 192x192 px et ne doivent pas avoir de coins arrondis. et l'écran de démarrage est beaucoup

plus compliqué, alors cliquez ici pour en savoir plus. Néanmoins, les dimensions minimales doivent être 2208x2208 px.

si vous avez un fichier icône pour générer, utilisez cette commande `ionic resources --icon` si vous avez un fichier splash pour générer l'utilisation de cette commande `ionic resources --splash`

3. Application de production de bâtiments

Avant de créer votre application de production, supprimez toutes les données de journal sensibles.

Pour construire une version avec toutes les optimisations par défaut, utilisez la balise --release & --prod

```
ionic build android --release --prod
```

Pour une liste complète des optimisations disponibles, vous pouvez visiter le référentiel @ ionic / app-scripts

4. Créer une clé privée

Maintenant, nous devons signer l'APP non signé (`android-release-unsigned.apk`) et lancer un utilitaire d'alignement pour l'optimiser et le préparer pour l'App Store. Si vous avez déjà une clé de signature, ignorez ces étapes et utilisez-la à la place.

Ensuite, localisez le fichier APK non signé `android-release-unsigned.apk` dans le répertoire du projet `/platforms/android/build/outputs/apk/` et utilisez la commande `keytools` qui sera utilisée pour signer notre fichier apk. Vous pouvez utiliser l'exemple ci-dessous:

```
$ keytool -genkey -v -keystore my-release-key.keystore -alias androidKey -keyalg RSA -keysize
2048 -validity 10000
```

Vous pouvez trouver `my-release-key.keystore` dans votre répertoire actuel.

Générons notre clé privée en utilisant la commande keytool fournie avec le JDK. Si cet outil est introuvable, reportez-vous au guide d'installation:

Vous serez invité à créer un mot de passe pour le fichier de clés. Ensuite, répondez aux questions des autres outils sympas et lorsque tout sera terminé, vous devriez avoir un fichier nommé my-release-key.keystore créé dans le répertoire en cours.

Note: Assurez-vous de sauvegarder ce fichier dans un endroit sûr, si vous le perdez, vous ne pourrez pas soumettre de mises à jour à votre application!

5. Sign APK

Pour signer l'APP non signé, exécutez l'outil jarsigner qui est également inclus dans le JDK:

```
$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore
HelloWorld-release-unsigned.apk alias_name
```

Cela signe l'apk en place. Enfin, nous devons exécuter l'outil d'alignement zip pour optimiser l'APK. L'outil zipalign se trouve dans / path / to / Android / sdk / build-tools / VERSION / zipalign.

```
$ zipalign -v 4 HelloWorld-release-unsigned.apk HelloWorld.apk
```

Nous avons maintenant notre version finale binaire appelée HelloWorld.apk et nous pouvons la publier sur le Google Play Store pour que tout le monde puisse en profiter!

Publiez votre application sur Google Play Store. Maintenant que notre version APK est prête pour le Google Play Store, nous pouvons créer une liste Play Store et télécharger notre APK. Pour commencer, vous devez visiter la console du développeur de Google Play Store et créer un nouveau compte de développeur. Il vous en coûtera 25 $ par heure.

Une fois que vous avez un compte développeur, vous pouvez cliquer sur "Publier une application Android sur Google Play" et suivre les instructions à l'écran.

Lire Du code à l'App Store - Android en ligne: https://riptutorial.com/fr/ionic2/topic/9659/du-code-a-l-app-store---android

Chapitre 9: Géolocalisation

Examples

Usage simple

Dans votre `package.json` assurez-vous d'inclure les dépendances:

```
{
    ...
    "dependencies": {
        ...
        "ionic-native": "^1.3.10",
        ...
    },
    ...
}
```

Pour utiliser la géolocalisation:

```
// custom-component.ts

import {Geolocation} from 'ionic-native';
import template from './custom-component.html';

@Component({
    selector: 'custom-component',
    template: template
})
export class CustomComponent {

    constructor() {

        // get the geolocation through a promise
        Geolocation.getCurrentPosition().then((position:Geoposition)=> {
            console.log(
                position.coords.latitude,
                position.coords.longitude);
        });
    }
}
```

Regarder la position

Pour une solution plus temps réel, vous pouvez utiliser la fonction watchPosition dans Geolocation qui avertit chaque fois qu'une erreur ou un changement de position se produit. Contrairement à la méthode getCurrentPosition, la fonction watchPosition renvoie un objet Observable.

```
import {Geolocation} from 'ionic-native';
import template from './custom-component.html';

@Component({
```

```
selector: 'custom-component',
template: template
})
export class CustomComponent {
constructor() {

    // get the geolocation through an observable
        Geolocation.watchPosition(<GeolocationOptions>{
            maximumAge: 5000, // a maximum age of cache is 5 seconds
            timeout: 10000, // time out after 10 seconds
            enableHighAccuracy: true // high accuracy
        }).subscribe((position) => {
            console.log('Time:' + position.timestamp);
            console.log(
                'Position:' + position.coords.latitude + ',' +
                position.coords.longitude);
            console.log('Direction:' position.coords.heading);
            console.log('Speed:' position.coords.speed);

        });
}
```

Lire Géolocalisation en ligne: https://riptutorial.com/fr/ionic2/topic/5840/geolocalisation

Chapitre 10: InAppBrowser

Introduction

Parfois, le client a juste besoin d'ouvrir une application Web dans une application mobile. Pour cela, nous pouvons utiliser InAppBrowser de manière à ce qu'il ressemble à une application. En revanche, nous ouvrons un site Web / une application Web sur mobile. au lieu d'ouvrir la première vue d'application, nous pouvons ouvrir directement InAppBrowser.

Examples

Un exemple concret de cette utilisation est cette application:

Dans cette application, j'ouvre directement InAppBrowser lorsque l'utilisateur touche l'icône de l'application au lieu de charger la première page de l'application. Cela donnerait l'impression aux utilisateurs qu'ils regardent l'application du même site Web ou de la même application Web.

Exemple de code pour utiliser InAppBrowser

```
  platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
      var url= "https://blog.knoldus.com/";
      var browserRef = window.cordova.InAppBrowser.open(url, "_self", "location=no",
"toolbar=no");
      browserRef.addEventListener("exit", (event) => {
          return navigator["app"].exitApp();
        }
      );
```

Lire InAppBrowser en ligne: https://riptutorial.com/fr/ionic2/topic/9801/inappbrowser

Chapitre 11: Modals

Examples

Utiliser des modaux

Les modaux se glissent en dehors de l'écran pour afficher une interface utilisateur temporaire, souvent utilisée pour les pages de connexion ou d'inscription, la composition des messages et la sélection des options.

```
import { ModalController } from 'ionic-angular';
import { ModalPage } from './modal-page';

export class MyPage {
  constructor(public modalCtrl: ModalController) {
  }

  presentModal() {
    let modal = this.modalCtrl.create(ModalPage);
    modal.present();
  }
}
```

REMARQUE: Un modal est un volet de contenu qui survole la page actuelle de l'utilisateur.

Transmission de données via un modal

Les données peuvent être transmises à un nouveau modal via `Modal.create()` comme second argument. Les données peuvent alors être consultées à partir de la page ouverte en injectant `NavParams` . Notez que la page, qui s'ouvre en tant que modal, n'a pas de logique "modale" spéciale, mais utilise `NavParams` différemment d'une page standard.

Première page:

```
import { ModalController, NavParams } from 'ionic-angular';

export class HomePage {

 constructor(public modalCtrl: ModalController) {

 }

 presentProfileModal() {
   let profileModal = this.modalCtrl.create(Profile, { userId: 8675309 });
   profileModal.present();
 }

}
```

Deuxième page:

```
import { NavParams } from 'ionic-angular';
export class Profile {

 constructor(params: NavParams) {
   console.log('UserId', params.get('userId'));
 }

}
```

Lire Modals en ligne: https://riptutorial.com/fr/ionic2/topic/6415/modals

Chapitre 12: Modals

Examples

Modal Simple

Modal est une interface utilisateur temporaire affichée en haut de votre page actuelle. Ceci est souvent utilisé pour la connexion, l'inscription, l'édition des options existantes et la sélection des options.

Examinons un exemple simple avec des modaux utilisés. Pour commencer, nous créons un projet vierge ionique. Laissez-nous créer un modal simple affichant un message et quitter le clic sur le bouton. Pour ce faire, nous créons d'abord une vue pour notre modal.

Message.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Modal
    </ion-title>
    <ion-buttons start>
      <button (click)="dismiss()">
        <span primary showWhen="ios">Cancel</span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <h1>Modal Without Params is created successfully.</h1>
  <button full (click)="dismiss()"> Exit </button>
</ion-content>
```

Message.ts

```
import { Component } from '@angular/core';
import { ViewController } from 'ionic-angular';
@Component({
  templateUrl: 'build/pages/message/message.html',
})
export class MessagePage {
  viewCtrl;
  constructor(viewCtrl: ViewController) {
    this.viewCtrl = viewCtrl;
  }
  dismiss(){
    this.viewCtrl.dismiss();
  }
}
```

Ce modal affiche un message. Le modal peut être fermé ou «supprimé» à l'aide de la méthode de

rejet View controllers.

Home.html

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Modal Example
    </ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <button full (click)="openModal()">ModalWithoutParams-Message</button>
</ion-content>
```

Home.ts

```
import { Component } from '@angular/core';
import { ModalController } from 'ionic-angular';
import {MessagePage} from '../message/message';
@Component({
  templateUrl: 'build/pages/home/home.html'
})
export class HomePage {
  modalCtrl;
  data;
  constructor(modalCtrl: ModalController) {
    this.modalCtrl = modalCtrl;
    this.data = [{name: "aaa", email: "aaa.a@som.com", mobile: "1234567890", nickname: "zzz"},
      {name: "bbb", email: "bbb.a@som.com", mobile: "1234567890", nickname: "yyy"},
      {name: "ccc", email: "ccc.a@som.com", mobile: "1234567890", nickname: "xxx"}]
  }
  openModal() {
    let myModal = this.modalCtrl.create(MessagePage);
    myModal.present();
  }
}
```

Nous créons maintenant notre page d'accueil pour importer **ModalController** et notre modèle de données MessagePage. La méthode de **création** de ModalController crée un modal pour notre modèle de données MessagePage enregistré pour contrôler la variable myModal. **La** méthode **actuelle** ouvre le modal en haut de notre page actuelle.

Modal avec paramètres sur le rejet:

Nous savons maintenant comment créer un modal. Mais que faire si nous voulons transmettre certaines données de modal à notre page d'accueil. Pour ce faire, examinons un exemple avec modal comme registre des paramètres de passage de page à la page parente.

Register.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>
```

```
      Login
    </ion-title>
    <ion-buttons start>
      <button (click)="dismiss()">
        <span primary showWhen="ios">Cancel</span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <ion-list>
    <ion-item>
      <ion-label>Name</ion-label>
      <ion-input type="text" [(ngModel)]="name"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Email</ion-label>
      <ion-input type="text" [(ngModel)]="email"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Mobile</ion-label>
      <ion-input type="number" [(ngModel)]="mobile"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Nickname</ion-label>
      <ion-input type="text" [(ngModel)]="nickname"></ion-input>
    </ion-item>
  </ion-list>
  <button full (click)="add()">Add</button>
</ion-content>
```

Register.ts

```
import { Component } from '@angular/core';
import { ViewController } from 'ionic-angular';
@Component({
  templateUrl: 'build/pages/register/register.html',
})
export class ResisterPage {
  viewCtrl;
  name;
  email;
  mobile;
  nickname;
  constructor(viewCtrl: ViewController) {
    this.viewCtrl = viewCtrl;
    this.name = "";
    this.email = "";
    this.mobile = "";
    this.nickname = "";
  }
  dismiss(){
    this.viewCtrl.dismiss();
  }
  add(){
    let data = {"name": this.name, "email": this.email, "mobile": this.mobile, "nickname":
this.nickname};
    this.viewCtrl.dismiss(data);
  }
```

```
}
```

Register modal obtient un objet de données avec les valeurs entrées par l'utilisateur et les paramètres sont transmis à notre page en cours sur la méthode de renvoi avec viewControllers. Maintenant, les paramètres sont envoyés.

Alors, comment allons-nous récupérer les paramètres dans la page d'accueil? Pour ce faire, nous créons un bouton sur la page d'accueil et appelons Register modal on click. Pour afficher l'utilisateur, nous affichons une liste.

Home.html

```
<ion-list>
  <ion-item *ngFor="let datum of data">
    <h1>{{datum.name}}</h1>
  </ion-item>
</ion-list>
<button full secondary (click)="openModalParams()">ModalWithParams-Register</button>
```

Home.ts

```
import {ResisterPage} from '../register/register';

  openModalParams(){
    let modalWithParams = this.modalCtrl.create(ResisterPage);
    modalWithParams.present();

    modalWithParams.onDidDismiss((result) =>{
      if(result){
        this.data.unshift(result);
      }
    });
  }
```

La méthode ViewController **onDidDismiss** est exécutée chaque fois qu'un modal est fermé. Si les données sont passées en paramètre à partir de modal, nous pouvons les récupérer en utilisant la méthode onDidDismiss. Ici, les données saisies par l'utilisateur sont ajoutées aux données existantes. Si aucune donnée n'est passée en paramètre, la valeur renvoyée sera nulle.

Modal avec paramètres sur create:

Passer des paramètres à un modal est similaire à la manière dont nous transmettons des valeurs à un NavController. Pour ce faire, nous modifions notre liste dans home.html pour ouvrir un modal en cliquant sur un élément de la liste et en transmettant les paramètres requis comme second argument à la méthode **create** .

Home.html

```
<ion-list>
   <ion-item *ngFor="let datum of data" (click)="openModalwithNavParams(datum)">
     <h1>{{datum.name}}</h1>
   </ion-item>
```

```
  </ion-list>
```

Home.ts

```
import {EditProfilePage} from '../edit-profile/edit-profile';

  openModalwithNavParams(data){
    let modalWithNavParams = this.modalCtrl.create(EditProfilePage,{Data: data});
    modalWithNavParams.present();
  }
```

Semblable à d'autres vues, nous utilisons NavParams pour récupérer les données envoyées depuis la vue précédente.

Edit-Profile.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Login
    </ion-title>
    <ion-buttons start>
      <button (click)="dismiss()">
        <span primary showWhen="ios">Cancel</span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <h2>Welcome {{name}}</h2>
  <ion-list>
    <ion-item>
      <ion-label>Email</ion-label>
      <ion-input type="text" value={{email}}></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Mobile</ion-label>
      <ion-input type="number" value={{mobile}}></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Nickname</ion-label>
      <ion-input type="text" value={{nickname}}></ion-input>
    </ion-item>
  </ion-list>
  <button full (click)="dismiss()">Close</button>
</ion-content>
```

Edit-Profile.ts

```
import { Component } from '@angular/core';
import { ViewController, NavParams } from 'ionic-angular';
@Component({
  templateUrl: 'build/pages/edit-profile/edit-profile.html',
})
export class EditProfilePage {
  viewCtrl;
```

```
  navParams;
  data;
  name;
  email;
  mobile;
  nickname;
  constructor(viewCtrl: ViewController, navParams: NavParams) {
    this.viewCtrl = viewCtrl;
    this.navParams = navParams;
    this.data = this.navParams.get('Data');
    this.name = this.data.name;
    this.email = this.data.email;
    this.mobile = this.data.mobile;
    this.nickname = this.data.nickname;

  }
  dismiss(){
    this.viewCtrl.dismiss();
  }
}
```

Lire Modals en ligne: https://riptutorial.com/fr/ionic2/topic/6612/modals

Chapitre 13: Notification Push envoyée et reçue

Remarques

Le SenderID qui est présent dans l'exemple d'initialisation est un identifiant d'expéditeur gcm qui vous est donné par Google. Il devrait également être présent lorsque vous installez le plugin

```
ionic plugin add phonegap-plugin-push --variable SENDER_ID="XXXXXXX"
```

Si vous souhaitez ajouter des données supplémentaires à vos notifications push, consultez ce lien en expliquant comment ajouter plus de caractères. Https://github.com/phonegap/phonegap-plugin-push/blob/master/docs/TYPESCRIPT.md

Examples

Initialisation

Le plug-in de notification push nécessite une initialisation d'initialisation qui indique au plug-in de commencer à s'exécuter à l'aide de l'ID de l'expéditeur fourni.

```
 let push = Push.init({
     android: {
       senderID: "------------",
     },
     ios: {
       alert: "true",
       badge: true,
       sound: "false",
     },
     windows: {},
   });
```

enregistrement

L'étape d'enregistrement enregistre l'application avec le système de l'appareil et renvoie un identifiant d'enregistrement

```
import { Push, RegistrationEventResponse} from "ionic-native";

       //the push element is created in the initialization example
       push.on("registration", async (response: RegistrationEventResponse) => {
                   //The registration returns an id of the registration on your device
                   RegisterWithWebApi(response.registrationId);

       });
```

Recevoir une notification push

Pour recevoir des notifications push, nous sommes supposés demander au plug-in d'écouter les notifications push entrantes. Cette étape est effectuée après l'initialisation et l'enregistrement

```
import { Push, NotificationEventResponse} from "ionic-native";

        //the push element is created in the initialization example
        push.on("notification", (response: NotificationEventResponse) => {
            let chatMessage: ChatMessage = <ChatMessage>{
              title: response.title,
              message: response.message,
              receiver: response.additionalData.replyTo,
              image: response.image
            };
            DoStuff(chatMessage));
       });
```

Lire Notification Push envoyée et reçue en ligne:
https://riptutorial.com/fr/ionic2/topic/5874/notification-push-envoyee-et-recue

Chapitre 14: Solution de contournement pour 'show-delete' dans désapprobation

Examples

Solution

Je développe une application mobile utilisant ionic 2 avec Angular 2.

J'ai une liste d'ions remplie d'ions. Je souhaite que ces éléments puissent être supprimés si nécessaire, comme présenté ici sur le site Web ionique.

Cependant, beaucoup de choses ont changé dans **ionic 2** depuis la première version et le style d'un bouton ouvrant tous les éléments **ioniques ci-** dessus n'est plus possible puisque **show-delete** et **show-reorder** ne sont plus pris en charge. La seule option disponible consiste à faire **glisser** les éléments ioniques en tant qu'élément ionique, ce qui nous permet de faire glisser chaque élément un par un afin de révéler le bouton de suppression.

Ce n'est pas ce que je voulais. Je voulais un bouton qui ouvre tous les éléments ioniques en même temps.

Après avoir passé un peu de temps là-dessus, j'ai trouvé une solution de travail et réussi à atteindre le résultat souhaité en utilisant ionic 2, et je vais le partager avec vous.

Voici ma solution:

Dans le fichier .html:

```
<ion-header>
  <ion-navbar>
    <ion-buttons start (click)="manageSlide()">
      <button>
        <ion-icon name="ios-remove"></ion-icon>
      </button>
    </ion-buttons>
    <ion-title>PageName</ion-title>
  </ion-navbar>
</ion-header>
```

et pour la liste:

```
<ion-list #list1>
  <ion-item-sliding #slidingItem *ngFor="let contact of contacts | sortOrder">
    <button #item ion-item>
      <p>{{ item.details }}</p>
      <ion-icon id="listIcon" name="arrow-forward" item-right></ion-icon>
    </button>
    <ion-item-options side="left">
      <button danger (click)="doConfirm(contact, slidingItem)">
```

```
        <ion-icon name="ios-remove-circle-outline"></ion-icon>
      Remove
      </button>
    </ion-item-options>
  </ion-item-sliding>
</ion-list>
```

Dans le fichier **.ts** , commencez par importer:

```
import { ViewChild } from '@angular/core';
import { Item } from 'ionic-angular';
import { ItemSliding, List } from 'ionic-angular';
```

puis se référer à l'élément html en déclarant un ViewChild:

```
@ViewChild(List) list: List;
```

Enfin, ajoutez vos classes pour gérer le travail:

```
public manageSlide() {

    //loop through the list by the number retreived of the number of ion-item-sliding in the
list
    for (let i = 0; i < this.list.getElementRef().nativeElement.children.length; i++) {

        // retreive the current ion-item-sliding
        let itemSlide = this.list.getElementRef().nativeElement.children[i].$ionComponent;

        // retreive the button to slide within the ion-item-sliding
        let item = itemSlide.item;

        // retreive the icon
        let ic = item._elementRef.nativeElement.children[0].children[1];

        if (this.deleteOpened) {
            this.closeSlide(itemSlide);
        } else {
            this.openSlide(itemSlide, item, ic);
        }
    }

    if (this.deleteOpened) {
        this.deleteOpened = false;
    } else {
        this.deleteOpened = true;
    }
}
```

Ensuite, la classe d'ouverture:

```
private openSlide(itemSlide: ItemSliding, item: Item, inIcon) {
  itemSlide.setCssClass("active-sliding", true);
  itemSlide.setCssClass("active-slide", true);
  itemSlide.setCssClass("active-options-left", true);
  item.setCssStyle("transform", "translate3d(72px, 0px, 0px)")
}
```

Et la classe de clôture:

```
private closeSlide(itemSlide: ItemSliding) {
  itemSlide.close();
  itemSlide.setCssClass("active-sliding", false);
  itemSlide.setCssClass("active-slide", false);
  itemSlide.setCssClass("active-options-left", false);
```

}

J'espère que cela vous aidera.

Bonne lecture et bon codage ...

Lire Solution de contournement pour 'show-delete' dans désapprobation en ligne: https://riptutorial.com/fr/ionic2/topic/6620/solution-de-contournement-pour--show-delete--dans--ion-list--desapprobation

Chapitre 15: Test d'unité

Introduction

Les tests unitaires en général apportent une sécurité supplémentaire à un produit pour éviter des problèmes lors de la modification / ajout de fonctionnalités. Un filet de sécurité qui dit "TOUT FONCTIONNE ENCORE". Les tests unitaires ne remplacent en aucune manière les tests utilisateur réels qu'un AQ peut effectuer.

Dans ce document, nous baserons les exemples sur ce dépôt: https://github.com/driftyco/ionic-unit-testing-example

Examples

Tests unitaires avec Karma / Jasmine

Les tests unitaires en mode ionique sont les mêmes que dans toute application angulaire.

Nous utiliserons quelques frameworks pour cela.

Karma - un cadre pour l'exécution de tests

Jasmine - un cadre pour écrire des tests

PhantomJS - une application qui exécute javascript sans navigateur

Tout d'abord, tout installer, alors assurez-vous que votre package.json inclut ces lignes dans les dépendances de développement. Je pense qu'il est important de noter que les dépendances de développement n'affectent pas du tout votre application et sont juste là pour aider le développeur.

```
"@ionic/app-scripts": "1.1.4",
"@ionic/cli-build-ionic-angular": "0.0.3",
"@ionic/cli-plugin-cordova": "0.0.9",
"@types/jasmine": "^2.5.41",
"@types/node": "^7.0.8",
"angular2-template-loader": "^0.6.2",
"html-loader": "^0.4.5",
"jasmine": "^2.5.3",
"karma": "^1.5.0",
"karma-chrome-launcher": "^2.0.0",
"karma-jasmine": "^1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"karma-sourcemap-loader": "^0.3.7",
"karma-webpack": "^2.0.3",
"null-loader": "^0.1.1",
"ts-loader": "^2.0.3",
"typescript": "2.0.9"
```

Pour dépasser un peu les paquets

```
"angular2-template-loader": "^0.6.2", - will load and compile the angular2 html files.

"ts-loader": "^2.0.3", - will compile the actual typescript files

"null-loader": "^0.1.1", - will not load the assets that will be missing, such as fonts and
images. We are testing, not image lurking.
```

Nous devrions également ajouter ce script à nos scripts package.json:

```
"test": "karma start ./test-config/karma.conf.js"
```

Notez également dans tsconfig que vous excluez les fichiers spec.ts de la compilation:

```
 "exclude": [
    "node_modules",
    "src/**/*.spec.ts"
  ],
```

Ok, maintenant, prenons la configuration de test réelle. Créez un dossier `test-config` dans votre dossier de projet. (Juste comme cela a été mentionné dans le script package.json) Dans le dossier, créez 3 fichiers:

`webpack.test.js` - qui indiquera au Webpack les fichiers à charger pour le processus de test

```
var webpack = require('webpack');
var path = require('path');

module.exports = {
  devtool: 'inline-source-map',

  resolve: {
    extensions: ['.ts', '.js']
  },

  module: {
    rules: [
      {
        test: /\.ts$/,
        loaders: [
          {
            loader: 'ts-loader'
          } , 'angular2-template-loader'
        ]
      },
      {
        test: /\.html$/,
        loader: 'html-loader'
      },
      {
        test: /\.(png|jpe?g|gif|svg|woff|woff2|ttf|eot|ico)$/,
        loader: 'null-loader'
      }
    ]
  },

  plugins: [
```

```
    new webpack.ContextReplacementPlugin(
      // The (\\|\/) piece accounts for path separators in *nix and Windows
      /angular(\\|\/)core(\\|\/)(esm(\\|\/)src|src)(\\|\/)linker/,
      root('./src'), // location of your src
      {} // a map of your routes
    )
  ]
};

function root(localPath) {
  return path.resolve(__dirname, localPath);
}
```

`karma-test-shim.js` - qui chargera les bibliothèques liées aux angles, telles que les bibliothèques de zones et de test, et configurera le module pour le test.

```
Error.stackTraceLimit = Infinity;

require('core-js/es6');
require('core-js/es7/reflect');

require('zone.js/dist/zone');
require('zone.js/dist/long-stack-trace-zone');
require('zone.js/dist/proxy');
require('zone.js/dist/sync-test');
require('zone.js/dist/jasmine-patch');
require('zone.js/dist/async-test');
require('zone.js/dist/fake-async-test');

var appContext = require.context('../src', true, /\.spec\.ts/);

appContext.keys().forEach(appContext);

var testing = require('@angular/core/testing');
var browser = require('@angular/platform-browser-dynamic/testing');

testing.TestBed.initTestEnvironment(browser.BrowserDynamicTestingModule,
browser.platformBrowserDynamicTesting());
```

`karma.conf.js` - définit la configuration du test avec le karma. Ici, vous pouvez passer de Chrome à PhantomJS pour rendre ce processus invisible et plus rapide, entre autres.

```
var webpackConfig = require('./webpack.test.js');

module.exports = function (config) {
  var _config = {
    basePath: '',

    frameworks: ['jasmine'],

    files: [
      {pattern: './karma-test-shim.js', watched: true}
    ],

    preprocessors: {
      './karma-test-shim.js': ['webpack', 'sourcemap']
    },
```

```
    webpack: webpackConfig,

    webpackMiddleware: {
      stats: 'errors-only'
    },

    webpackServer: {
      noInfo: true
    },

    browserConsoleLogOptions: {
      level: 'log',
      format: '%b %T: %m',
      terminal: true
    },

    reporters: ['kjhtml', 'dots'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false
  };

  config.set(_config);
};
```

Maintenant que nous avons configuré tout, nous allons écrire un test réel. Pour cet exemple, nous allons écrire un fichier de spécifications app.component. Si vous souhaitez voir des tests pour une page et non le composant principal, vous pouvez regarder ici: https://github.com/driftyco/ionic-unit-testing-example/blob/master/src/pages/page1/page1. spec.ts

Ce que nous devons faire d'abord, c'est tester notre constructeur. Cela va créer et exécuter le constructeur de notre app.component

```
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [MyApp],
      imports: [
        IonicModule.forRoot(MyApp)
      ],
      providers: [
        StatusBar,
        SplashScreen
      ]
    })
  }));
```

La déclaration inclura notre application ionique principale. Les importations seront les importations nécessaires pour ce test. Pas tout.

Les fournisseurs incluront les éléments injectés dans le constructeur mais ne feront pas partie de l'importation. Par exemple, app.component injecte le service Platform, mais depuis qu'il fait partie de l'IonicModule, il n'est pas nécessaire de le mentionner dans les fournisseurs.

Pour les prochains tests, nous aurons besoin d'une instance de notre composant:

```
  beforeEach(() => {
    fixture = TestBed.createComponent(MyApp);
    component = fixture.componentInstance;
  });
```

Après quelques tests pour voir que tout est en ordre:

```
  it ('should be created', () => {
    expect(component instanceof MyApp).toBe(true);
  });

  it ('should have two pages', () => {
    expect(component.pages.length).toBe(2);
  });
```

Donc à la fin nous aurons quelque chose comme ça:

```
import { async, TestBed } from '@angular/core/testing';
import { IonicModule } from 'ionic-angular';

import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';

import { MyApp } from './app.component';

describe('MyApp Component', () => {
  let fixture;
  let component;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [MyApp],
      imports: [
        IonicModule.forRoot(MyApp)
      ],
      providers: [
        StatusBar,
        SplashScreen
      ]
    })
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(MyApp);
    component = fixture.componentInstance;
  });

  it ('should be created', () => {
    expect(component instanceof MyApp).toBe(true);
  });

  it ('should have two pages', () => {
    expect(component.pages.length).toBe(2);
  });

});
```

Exécutez les tests par

```
npm run test
```

Et c'est à peu près tout pour les tests de base. Il y a plusieurs façons de raccourcir le test d'écriture, comme écrire votre propre TestBed et avoir l'héritage dans des tests qui pourraient vous aider à long terme.

Lire Test d'unité en ligne: https://riptutorial.com/fr/ionic2/topic/9561/test-d-unite

Chapitre 16: Utilisation des onglets

Remarques

N'oubliez jamais de consulter la documentation Ionic 2 Tab pour connaître les dernières modifications et mises à jour.

Examples

Modifier l'onglet sélectionné par programme à partir de la page enfant

Vous pouvez consulter le code complet de ce Plunker fonctionnel .

Dans cet exemple, j'utilise un service partagé pour gérer la communication entre les pages de l'onglet (pages enfants) et le conteneur d'onglets (composant contenant les onglets). Même si vous pouvez probablement le faire avec Events, j'aime l'approche de service partagé car elle est plus facile à comprendre et à maintenir lorsque l'application se développe.

TabService

```
import {Injectable} from '@angular/core';
import {Platform} from 'ionic-angular/index';
import {Observable} from 'rxjs/Observable';

@Injectable()
export class TabService {

  private tabChangeObserver: any;
  public tabChange: any;

  constructor(private platform: Platform){
    this.tabChangeObserver = null;
    this.tabChange = Observable.create(observer => {
        this.tabChangeObserver = observer;
    });
  }

  public changeTabInContainerPage(index: number) {
    this.tabChangeObserver.next(index);
  }
}
```

Ainsi, le `TabService` crée uniquement un `Observable` pour permettre au conteneur d'onglets de s'y abonner, et déclare également la méthode `changeTabInContainerPage()` qui sera appelée à partir des pages enfants.

Ensuite, dans chaque page enfant (celle à l'intérieur des onglets), nous ajoutons un bouton et lions l'événement `click` à une méthode qui appelle le service:

Page1.html

```
<ion-content class="has-header">
    <h1>Page 1</h1>
    <button secondary (click)="changeTab()">Select next tab</button>
</ion-content>
```

Page1.ts

```
import { Component } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { TabService } from 'tabService.ts';

@Component({
  templateUrl:"page1.html"
})
export class Page1 {

  constructor(private tabService: TabService) { }

  public changeTab() {
    this.tabService.changeTabInContainerPage(1);
  }
}
```

Et enfin, dans `TabsPage` , nous ne sommes abonnés qu'au service, puis nous modifions l'onglet sélectionné avec `this.tabRef.select(index);`

```
import { Component, ViewChild } from "@angular/core";
import { Page1 } from './page1.ts';
import { Page2 } from './page2.ts';
import { TabService } from 'tabService.ts';

@Component({
  templateUrl: 'tabs.html'
})
export class TabsPage {
  @ViewChild('myTabs') tabRef: Tabs;

  tab1Root: any = Page1;
  tab2Root: any = Page2;

  constructor(private tabService: TabService){
    this.tabService.tabChange.subscribe((index) => {
      this.tabRef.select(index);
    });
  }
}
```

Notez que nous obtenons une référence à l'instance Tabs en ajoutant `#myTabs` dans l'élément `ion-tabs` , et que nous l'obtenons depuis le composant avec `@ViewChild('myTabs') tabRef: Tabs;`

```
<ion-tabs #myTabs>
  <ion-tab [root]="tab1Root" tabTitle="Tab 1"></ion-tab>
  <ion-tab [root]="tab2Root" tabTitle="Tab 2"></ion-tab>
</ion-tabs>
```

Onglet Modifier avec selectedIndex

Au lieu d'obtenir une référence au DOM, vous pouvez simplement modifier l'index de l'onglet en utilisant l'attribut selectedIndex sur les onglets ion

HTML:

```
<ion-tabs [selectedIndex]="tabIndex" class="tabs-icon-text" primary >
     <ion-tab tabIcon="list-box"   [root]="tabOne"></ion-tab>
     <ion-tab tabIcon="contacts"  [root]="tabTwo"></ion-tab>
     <ion-tab tabIcon="chatboxes"  [tabBadge]="messagesReceived" [root]="tabFive"></ion-tab>
</ion-tabs>
```

TS:

```
import { Events} from "ionic-angular";

export class tabs {
 public tabIndex: number;
 constructor(e: Events) {
   tabs.mySelectedIndex = navParams.data.tabIndex || 0;
   e.subscribe("tab:change", (newIndex) => this.tabIndex = newIndex);
 }
}
```

Si vous souhaitez le modifier depuis un autre service de contrôleur, vous pouvez envoyer un événement:

```
e.publish("tab:change",2);
```

Lire Utilisation des onglets en ligne: https://riptutorial.com/fr/ionic2/topic/5569/utilisation-des-onglets

Chapitre 17: Utiliser les services

Remarques

Une chose très importante à propos de l'utilisation des services partagés est qu'ils doivent être inclus dans le tableau de `providers` du composant le plus élevé où ils doivent être partagés.

Pourquoi donc? Eh bien, supposons que nous `MyService` référence `MyService` dans le tableau des `providers` de chaque `Component` . Quelque chose comme:

```
@Component({
  templateUrl:"page1.html",
  providers: [MyService]
})
```

Et

```
@Component({
  templateUrl:"page2.html",
  providers: [MyService]
})
```

De cette manière, **une nouvelle instance du service sera créée pour chaque composant** afin que l'instance où une page enregistre les données sera différente de l'instance utilisée pour obtenir les données. Donc ça ne marchera pas.

Pour que l'application entière utilise la même instance (ce qui fait que le service fonctionne comme un service *singleton*), nous pouvons ajouter sa référence dans le `App Component` comme ceci:

```
@Component({
  template: '<ion-nav [root]="rootPage"></ion-nav>',
  providers: [MyService]
})
```

Vous pouvez également ajouter la référence `MyService` dans `ionicBootstrap(MyApp, [MyService]);` mais selon les guides de style Angular2

> Fournissez des services à l'injecteur Angular 2 dans le composant le plus élevé où ils seront partagés.
>
> Pourquoi? L'injecteur angulaire 2 est hiérarchique.
>
> Pourquoi? Lors de la fourniture du service à un composant de niveau supérieur, cette instance est partagée et disponible pour tous les composants enfants de ce composant de niveau supérieur.
>
> Pourquoi? C'est idéal lorsqu'un service partage des méthodes ou un état.

> Pourquoi? Cela n'est pas idéal lorsque deux composants différents ont besoin d'instances différentes d'un service. Dans ce scénario, il serait préférable de fournir le service au niveau des composants nécessitant une nouvelle instance distincte.

Et

> Ça va marcher. Ce n'est pas une bonne pratique. **L'option du fournisseur de bootstrap est destinée à la configuration et au remplacement des services préenregistrés d'Angular** , tels que sa prise en charge du routage.

... le `App Component` serait le meilleur choix.

Examples

Partager des informations entre différentes pages

L'un des exemples les plus simples d'utilisation *des services partagés* est le moment où nous souhaitons stocker des données à partir d'une page donnée de notre application, puis récupérer ces données à partir d'une autre page.

Une option pourrait consister à envoyer ces données en paramètre (par exemple, si une page appelle l'autre), mais si nous voulons utiliser ces données à partir d'une partie complètement différente de l'application, cela ne semble pas être le meilleur moyen de le faire. il. C'est à ce moment que *les services partagés* entrent en jeu.

Dans cet exemple, nous allons utiliser un service simple appelé `MyService` qui ne dispose que de deux méthodes simples: `saveMessage()` pour stocker une chaîne et `getMessage()` pour le récupérer à nouveau. Ce code fait partie de ce plunker de travail où vous pouvez le voir en action.

```
import {Injectable} from '@angular/core';

@Injectable()
export class MyService {

  private message: string;

  constructor(){ }

  public saveMessage(theMessage: string): void {
    this.message = theMessage;
  }

  public getMessage(): string {
    return this.message;
  }
}
```

Ensuite, lorsque nous voulons stocker un nouveau message, nous pouvons simplement utiliser le `saveMessage(theMessageWeWantToSave);` méthode de l'instance `MyService` (appelée juste `service`).

```
import { Component } from "@angular/core";
```

```
import { MyService } from 'service.ts';

@Component({
  templateUrl:"page1.html"
})
export class Page1 {

  message: string;

  // ...

  public saveSecretMessage(): void {
    this.service.saveMessage(this.message);
  }
}
```

De la même manière, lorsque nous voulons obtenir ces données, nous pouvons utiliser la méthode `getMessage()` de l'instance de service comme ceci:

```
import { Component } from "@angular/core";
import { MyService } from 'service.ts';

@Component({
  templateUrl:"page2.html"
})
export class Page2 {

  enteredMessage: string;

  constructor(private service: MyService) {
    this.enteredMessage = this.service.getMessage();
  }

  // ...
}
```

N'oubliez pas de vérifier la section *Remarques* pour savoir où la référence du service `MyService` être incluse et pourquoi.

Lire Utiliser les services en ligne: https://riptutorial.com/fr/ionic2/topic/4407/utiliser-les-services

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec ionic2	Akilan Arasu, Cameron637, carstenbaumhoegger, Community, FreeBird72, Guillaume Le Mière, Ian Pinto, Ketan Akbari, misha130, Raymond Ativie, sebaferreras, tymspy, Will.Harris
2	Ajouter une application ionique à la vue ionique	Saravanan Sachi
3	Angularfire2 avec Ionic2	Fernando Del Olmo
4	Composants Ionic2 CSS	Ketan Akbari
5	Configuration et débogage Ionic 2 dans le code Visual Studio	misha130, PRIYA PARASHAR
6	Connexion sociale avec Angularfire2 / Firebase	Cameron637, Gianfranco P.
7	Constructeur et OnInit	niks
8	Du code à l'App Store - Android	Luis Estevez, misha130
9	Géolocalisation	Matyas, misha130
10	InAppBrowser	niks
11	Modals	Raymond Ativie
12	Notification Push envoyée et reçue	misha130
13	Solution de contournement pour 'show-delete' dans désapprobation	Amr ElAdawy, Roman Lee

14	Test d'unité	misha130
15	Utilisation des onglets	misha130, sebaferreras
16	Utiliser les services	sebaferreras