Introduction à IONIC

Développement d'un chronomètre

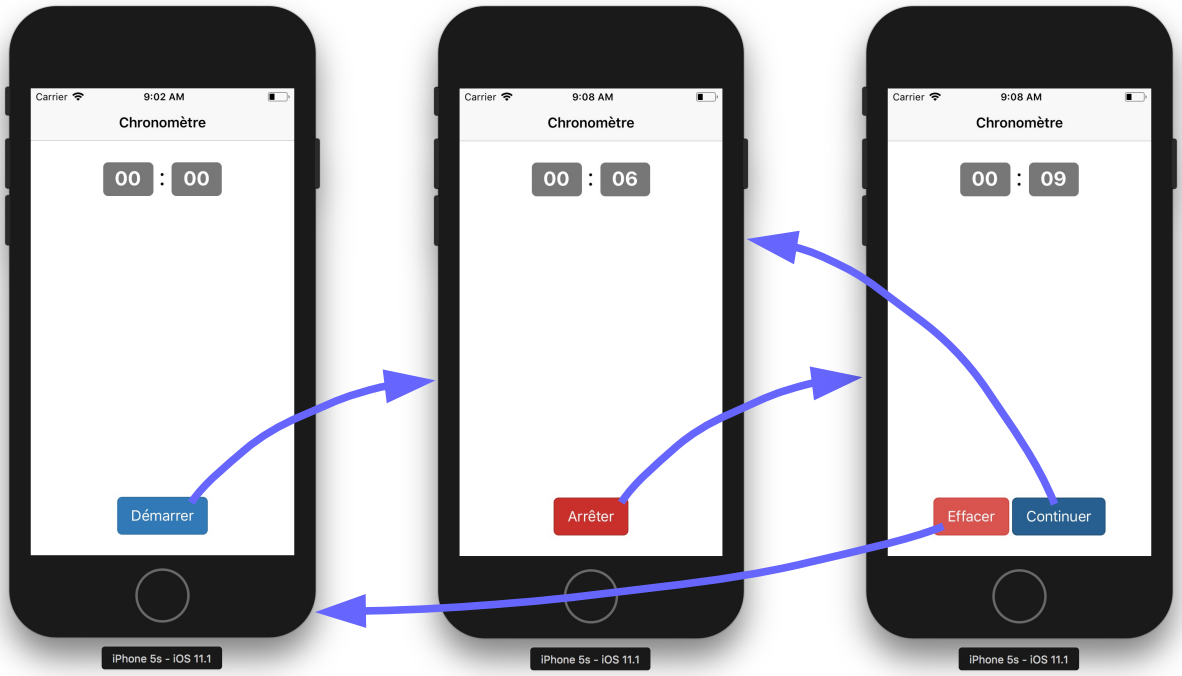
Introduction

Ionic est un framework qui permet, avec apache cordova, la création d'applications multi-plateforme (Android, IOS, Windows Phone) en utilisant les standards du Web comme HTML, CSS, Javascript.

Ionic 2 est basé sur Angular 2.

Cahier des charges

Ce TP consiste en la réalisation d'un chronomètre qui compte les minutes et les secondes. Il sera possible de démarrer le chronomètre, de l'arrêter et de reprendre le comptage ou de revenir à zéro.



Installations

Pré-requis

La version 8.x.x de Node.js doit être installée et la version 5.x.x pout npm. Pour le déploiement sur Android, il faut avoir installé Android Studio. Pour le déploiement sur IOS, il faut un MAC avec Xcode d'installé.

Test de Node.js et de npm

```
nodejs --version
v8.9.1
```

```
npm --version
5.5.1
```

Installation de ionic

Ionic s'installe via le Pacquage Manager de NodeJS.

Tapez la commande suivante dans une console en mode administrateur :

```
npm install -g ionic cordova
```

Pour tester l'installation, tapez la commande :

```
ionic --version
3.17.0
```

Ici, le système nous dit que la version installée est la 3.17.0.

Création et test d'un projet vide

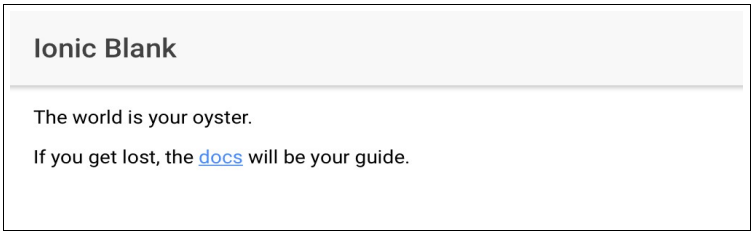
Nous allons créer un nouveau projet avec une page blanche. Pour cela, tapez la commande suivante :

```
ionic start stopwatch blank
```

Pour tester que le projet est bien initialisé, déplacez vous dans le répertoire créer par Ionic, puis lancez le serveur :

```
cd stopwatch
ionic serve
```

Enfin, dans un navigateur, ouvrez la page : http://localhost:8100. Vous devriez voir cette page :

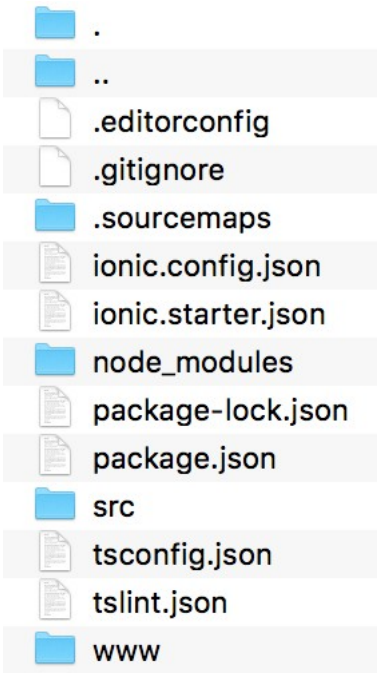


Prise en main d'Ionic

Examen de l'arborescence des fichiers

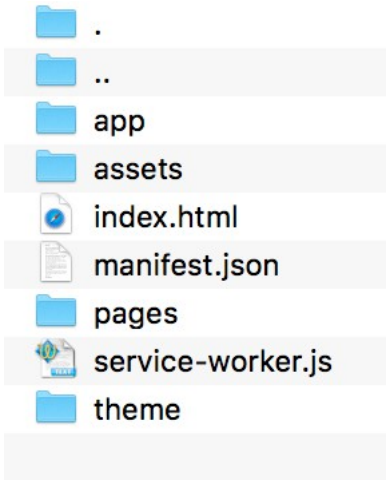
La racine du projet

A la racine du projet, nous trouvons tous les fichiers de configuration (.json) , les modules pré-installés dans le répertoire node_modules et les sources du projet dans le répertoire src. Le répertoire www contient l'application Web du projet.

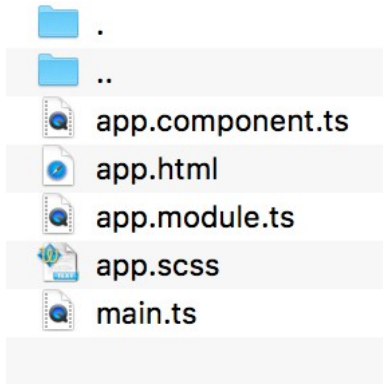


Le répertoire /src

Le répertoire src contient les sources du projet. Le fichier index.html est le fichier principal. Celui-ci n'est pas utilisable directement. Le répertoire pages contient les sources des différentes pages qui seront affichées. Le répertoire app contient les sources de l'application. Dans le répertoire assets, nous pourrons déposer toutes les ressources (images, vidéo, audio, etc.).



Le répertoire /src/app

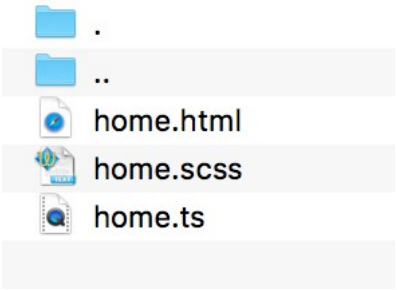


app.module.ts contient la déclaration du module.

app.component.ts, app.html et app.scss sont les sources du composant racine de l'application.

Le répertoire /src/pages/home

Dans le répertoire /src/pages, nous trouvons toutes les pages à afficher. A la création de l'application, nous trouvons dans le sous-répertoire home les fichiers de définition de la page d'accueil.



Architecture d'une application ionic

Toute la documentation se trouve sur le site ionic à l'adresse suivante : http://ionicframework.com/docs/

Le fichier /src/index.html

```
<ion-app></ion-app>
```

Ce fichier est le point d'entrée d'une application Ionic. Il contient les liens vers les fichier CSS et Javascript de Ionic ainsi que la balise ion-app.

Le fichier /src/app/app.module.ts

Ce fichier est le point d'entrée de l'application.

Le fichier /src/app/app.html

C'est le template principal de l'application.

```
<ion-nav [root]="rootPage"></ion-nav>
```

Le fichier /src/app/app.scss

Ce fichier contient les styles CSS qui s'appliquent globalement à l'application.

Le fichier /src/app/app.component.ts

C'est le fichier Typescript du composant principal.

Le répertoire /src/pages/home

Ce répertoire contient les fichiers de définition de la page d'accueil de l'application.

Fichier home.html par défaut :

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Ionic Blank
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  The world is your oyster.
  <p>
    If you get lost, the <a href="http://ionicframework.com/docs/v2">docs</a> will be
your guide.
  </p>
</ion-content>
```

Fichier home.scss par défaut :

```
page-home {

}
```

Fichier home.ts par défaut :

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  constructor(public navCtrl: NavController) {

  }

}
```

Application stopwatch

Ajouter bootstrap

Dans le <head> du fichier /src/index.html, ajouter un lien vers le CSS :

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

La class Timer

La class Timer (src/app/timer.ts) permet d'incrémenter les secondes et les minutes.

```
export class Timer {

    private minutes: number = 0;
    private secondes: number = 0;
    private totalSecondes: number = 0;
    private timer;

    start() {
        this.timer = setInterval(() => {
            this.minutes = Math.floor(++this.totalSecondes / 60);
            this.secondes = this.totalSecondes - this.minutes * 60;
        }, 1000);
    }

    stop() {
        clearInterval(this.timer);
    }

    reset() {
        this.totalSecondes = this.minutes = this.secondes = 0;
    }
}
```

La classe State

La class State (/src/app/state.ts) permet de démarrer, suspendre, reprendre et arrêter le chronomètre.

```
export class State {

    private play: boolean = true;
    private stop: boolean = false;
    private backward: boolean = false;

    setPlay() {
        this.stop = true;
        this.play = this.backward = false;
    }

    setStop() {
        this.stop = false;
        this.play = this.backward = true;
    }

    setBackward() {
        this.play = true;
        this.stop = this.backward = false;
    }
}
```

La page principale

Le typescript

Fichier /src/pages/home/home.ts

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';

import { Timer } from '../../app/timer';
import { State } from '../../app/state';

@Component({
     selector: 'page-home',
     templateUrl: 'home.html'
})
export class HomePage {

     private btnPlay: string = 'Démarrer';

     private timer: Timer = new Timer();
     private state: State = new State();

     constructor(public navCtrl: NavController) {}

     play() {
          this.timer.start();
          this.state.setPlay();
          this.btnPlay = 'Continuer';
     }

     stop() {
          this.timer.stop();
          this.state.setStop();
     }

     backward() {
          this.timer.reset();
          this.state.setBackward();
          this.btnPlay = 'Démarrer';
     }
}
```

Le fichier HTML

Fichier /src/pages/home/home.html

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

<ion-content text-center>
  <div class="digit">
    <span class="label label-default">{{timer.minutes | twoDigit}}</span> :
    <span class="label label-default">{{timer.secondes | twoDigit}}</span>
  </div>
</ion-content>

<ion-footer text-center>
  <div class="action">
    <button class="btn btn-danger btn-lg"
      *ngIf="state.backward" (click)="backward()">Effacer</button>
    <button class="btn btn-danger btn-lg"
      *ngIf="state.stop" (click)="stop()">Arrêter</button>
    <button class="btn btn-primary btn-lg"
      *ngIf="state.play" (click)="play()">{{btnPlay}}</button>
  </div>
</ion-footer>
```

Le style CSS

Fichier /src/pages/home/home.scss

```
page-home {

.digit {
     font-size: xx-large;
     text-align: center;
     margin: 15px;
     padding: 5px;
     border: 1px solid blak;
}

.action {
     text-align: center;
}

}
```

Afficher les minutes et les secondes sur 2 digits

Utilisation d'un pipe pour modifier l'affichage dynamiquement.

Créer le répertoire /src/app/pipes

Créer le fichier /src/app/pipes/two-digit.ts

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

@Pipe({
  name: 'twoDigit'
})
export class TwoDigit {
     transform(value, args) {
          return this.pad(value, 2, 0);
     }

     pad(n, width, z) {
          z = z || '0';
          n = n + '';
          return n.length >= width ? n : new Array(width - n.length + 1).join(z) + n;
     }
}
```

Mettre à jour le module (/src/app/app.module.ts)

```
import { TwoDigit } from './pipes/two-digit';

@NgModule({
  declarations: [
     TwoDigit
  ],
```

Utiliser le pipe dans /src/pages/home/home.html

```
<p class="digit">
  <span class="label label-default">{{timer.minutes | twoDigit}}</span> :
  <span class="label label-default">{{timer.secondes | twoDigit}}</span>
</p>
```

Internationalisation

Nous allons utiliser le module Angular ng2-translate. Voir la documentation à l'adresse suivante :

https://www.npmjs.com/package/ng2-translate

Ajout du module :

```
npm install ng2-translate --save
```

Modifications dans app.module.ts

Imports des modules :

```
import {HttpModule, Http } from '@angular/http';
import {TranslateModule, TranslateStaticLoader, TranslateLoader, TranslateService }
from 'ng2-translate/ng2-translate';
```

Fonction de création du loader :

```
export function createTranslateLoader(http: Http) {
  return new TranslateStaticLoader(http, './assets/i18n', '.json');
}
```

Ajouts dans @ngModule :

```
@NgModule({
  imports: [
    HttpModule,
    TranslateModule.forRoot({
      provide: TranslateLoader,
      useFactory: createTranslateLoader,
      deps: [Http]
    })
  ],

  exports: [TranslateModule],

  providers: [
    TranslateService
  ],
```

Modifications dans app.component.ts

```
import { Component } from '@angular/core';
import { Platform } from 'ionic-angular';
import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';
import {TranslateService } from 'ng2-translate/ng2-translate';


import { HomePage } from '../pages/home/home';


@Component({
  templateUrl: 'app.html'
})
export class MyApp {
  rootPage:any = HomePage;


  constructor(platform: Platform, statusBar: StatusBar, splashScreen: SplashScreen,
private translate: TranslateService) {
    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();
      splashScreen.hide();
    });


    this.initTranslation();
  }


  initTranslation() {
    var userLang = navigator.language.split('-')[0]; // use navigator lang if available
    userLang = /(fr|en)/gi.test(userLang) ? userLang : 'en';
    // this language will be used as a fallback when a translation isn't found in the
current language
    this.translate.setDefaultLang('en');
    // the lang to use, if the lang isn't available, it will use the current loader to
get them
    this.translate.use(userLang);
  }
}
```

Création des fichier de langues

/www/assets/i18n/en.json :

```
{
     "TITLE": "stopwatch",
     "START" : "Start",
     "STOP": "Stop",
     "RESET": "Reset",
     "CONTINUE": "Continue"
}
```

/www/assets/i18n/fr.json :

```
{
     "TITLE": "Chronomètre",
     "START" : "Démarrer",
     "STOP": "Arrêter",
     "RESET": "Effacer",
     "CONTINUE": "Continuer"
}
```

Utilisation dans home.html

```
<ion-header>
  <ion-navbar>
    <ion-title>
      {{'TITLE' | translate}}
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content text-center>
  <div class="digit">
    <span class="label label-default">{{timer.minutes | twoDigit}}</span> :
    <span class="label label-default">{{timer.secondes | twoDigit}}</span>
  </div>
</ion-content>

<ion-footer text-center>
  <div class="action">
    <button class="btn btn-danger btn-lg"
      *ngIf="state.backward" (click)="backward()">{{'RESET' | translate}}</button>
    <button class="btn btn-danger btn-lg"
      *ngIf="state.stop" (click)="stop()">{{'STOP' | translate}}</button>
    <button class="btn btn-primary btn-lg"
      *ngIf="state.play" (click)="play()">{{btnPlay | translate}}</button>
  </div>
</ion-footer>
```

Utilisation dans home.ts, dans la méthode backward() :

```
    private btnPlay: string = 'START';
    backward() {
         this.timer.reset();
         this.state.setBackward();
         this.btnPlay = 'START';
    }
```

Configuration des plateformes destination

```
ionic cordova platform add ios
```

```
ionic cordova platform add android
```

Icone de l'application

Placez l'icone et le splash screen dans le répertoire /resources.

Pour IOS dans le sous-répertoire ios. Et pour Android dans le sous répertoire android.

Créer un icone au format PNG d'au moins1024x1024 pixels. Cordova se charge de générer les différents formats pour IOS et Android.

Même chose pour le splash screen au format PNG d'au moins 2732x2732

Voir la documentation :

Icons: https://cordova.apache.org/docs/en/latest/config_ref/images.html

Splash Screens: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-splashscreen/

Lancer la commande :

```
ionic cordova resources
```

pour générer les différents format d'image pour chaque plateforme.

Déployer l'application

Pour Android :

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

Pour IOS

```
ionic cordova build ios --prod --release
```

Compiler l'application

Les projets se trouvent dans le répertoire /platforms.

Il faut utiliser AndroidStudio et Xcode pour générer les fichiers d'installation.

Pour le Web

Pour l'application Web, vous devez déposer sur le serveur Web le contenu du répertoire /www.