EBook Gratuito

APPRENDIMENTO ionic2

Free unaffiliated eBook created from
Stack Overflow contributors.

#ionic2

Sommario

Di

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

Capitolo 1: Iniziare con ionic2

Osservazioni

Ionic 2 è una tecnologia di sviluppo mobile multipiattaforma. Questo framework è progettato per la creazione di applicazioni mobili ibride e può essere utilizzato anche per applicazioni desktop. È una scrittura una volta, funziona ovunque con la tecnologia. Utilizza tecnologie web come JavaScript / Typescript, Angular 2, HTML e CSS (SCSS / LESS). Le app di Ionic2 funzionano bene su `>=android 4.4` , ma tu vuoi eseguire su `android 4.1` ad `android 4.3` devi usare cross walk .

Examples

Installazione o configurazione

Dato che Ionic 2 sta migliorando ogni giorno, ti preghiamo di controllare sempre la **documentazione ufficiale** per tenere traccia delle ultime modifiche e miglioramenti.

Prerequisiti: Avrai bisogno di NodeJS per costruire progetti Ionic 2. È possibile scaricare e installare il nodo qui e conoscere meglio NPM e pacchetti ionico 2 usa qui .

---

1. Installazione di Ionic 2

Come Ionic 1, è possibile utilizzare la CLI o la GUI Ionic per creare e testare rapidamente le app direttamente nel browser. Ha anche tutte le funzionalità per funzionare con le tue app Ionic 1, quindi non dovrai cambiare nulla!

Per usare Ionic 2 basta installare ionic da npm:

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

Se si verifica un errore EACCES, seguire le istruzioni qui per fornire al nodo le autorizzazioni necessarie.

2. Creazione della prima app

Una volta installata la CLI, esegui il seguente comando per avviare la tua prima app:

```
$ ionic start MyIonic2Project
```

Il modello di schede è utilizzato per impostazione predefinita, ma puoi scegliere un altro modello passando un flag. Per esempio:

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

Questo userà il modello del tutorial .

Per eseguire la tua app, cambia nella directory dei tuoi progetti ed esegui `ionic serve -lc` :

```
$ ionic serve -lc
```

L'opzione -l attiva il live ricaricamento della pagina, il -c visualizza i log della console. Se riscontri problemi nella creazione della tua app, assicurati che il tuo package.json corrisponda a quello della ionic2-app-base

Puoi giocare con la tua nuova app proprio lì nel browser!

3. Costruire su un dispositivo

Puoi anche costruire la tua nuova app su un dispositivo fisico o un emulatore di dispositivo. Avrai bisogno di Cordova per procedere.

Per installare Cordova, eseguire:

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

Consulta i documenti del simulatore iOS per la creazione di applicazioni iOS (NOTA: non è possibile creare su dispositivi iOS o emulatori su qualsiasi sistema operativo diverso da OSX) o i documenti Genymotion per creare un'applicazione Android.

Funzionando sul dispositivo iOS:

Per costruire un'app per iOS, è necessario lavorare su un computer OSX, perché avrai bisogno del framework di cacao per poterlo costruire per iOS, se è il caso devi prima aggiungere la piattaforma a cordova eseguendo il seguente comando:

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

Avrai bisogno di Xcode per compilare su un dispositivo iOS.

Infine, esegui la tua app con il seguente comando:

```
$ ionic cordova run ios
```

Funzionando su un dispositivo Android:

I passaggi per Android sono quasi identici. Innanzitutto, aggiungi la piattaforma:

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

Quindi installare l' SDK Android che consente di compilare su un dispositivo Android. Sebbene Android SDK sia dotato di un emulatore, è molto lento. Genymotion è molto più veloce. Una volta installato, basta eseguire il seguente comando:

```
$ ionic cordova run android
```

E questo è tutto! Congratulazioni per aver creato la tua prima app Ionic 2!

Anche Ionic ha una ricarica live. Quindi, se vuoi sviluppare la tua app e vedere i cambiamenti in atto sull'emulatore / dispositivo, puoi farlo eseguendo i seguenti comandi:

Per iOS:

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

Fai attenzione, su iOS 9.2.2 il fegato non funziona. Se vuoi lavorare con livereload, modifica il file config.xml aggiungendo quanto segue:

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

Quindi in `<platform name="ios">` :

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

Per Android:

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

`l` sta per live-reload, `c` per i log della console e `s` per i log del server. Questo ti permetterà di vedere se ci sono errori / avvertenze durante l'esecuzione.

Costruire per Windows

Se vuoi costruire il tuo progetto per Windows, devi lavorare su un computer Windows. Per iniziare, installa la piattaforma Windows sul tuo progetto ionic2 eseguendo il seguente comando:

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

Quindi esegui il seguente comando:

```
$ionic cordova run windows
```

Per eseguire nel browser

```
$ionic serve
```

per il browser Chrome controlla il dispositivo. (digitare nella barra degli indirizzi del browser Chrome)

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

Leggi Iniziare con ionic2 online: https://riptutorial.com/it/ionic2/topic/3632/iniziare-con-ionic2

Capitolo 2: Accesso sociale con Angularfire2 / Firebase

Examples

Accesso nativo a Facebook con 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));
      });
  }
}
```

Leggi Accesso sociale con Angularfire2 / Firebase online:
https://riptutorial.com/it/ionic2/topic/5518/accesso-sociale-con-angularfire2---firebase

Capitolo 3: Aggiungi l'app ionica alla vista ionica

introduzione

ionic view è un'app mobile che devi installare sul tuo cellulare, in modo che tu possa visualizzare la tua app senza creare file .apk. Condividendo il tuo ID app, gli altri utenti possono anche visualizzare la tua app sul cellulare utilizzando la vista ionica.

Sito: https://view.ionic.io/

Examples

Passi per aggiungere la tua app alla vista ionica

I seguenti passaggi devono essere eseguiti nell'app.ionic.io

1. Crea un account o accedi al tuo account ionico
2. Fai clic su "Nuova app" nella Dashboard e dai il nome alla tua app

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

3. Nella sezione panoramica di questa app appena creata, ci sarà un ID sotto il nome dell'app.

```
MyIonicApp ID is 4c5051c1
```

Di seguito sono riportati i passaggi nel prompt dei comandi **Node.js**

1. Accedi al tuo account ionico eseguendo

```
$ ionic login
```

2. Installa la tua cartella dell'app.
3. Per caricare la tua app in vista ionica, devi prima collegare la tua app con l'ID che hai creato nel sito ionico. Esegui il seguente comando per collegare,

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

Per MyIoincApp, il comando sarà,

```
$ ionic link 4c5051c1
```

Il comando precedente aggiornerà l'ID app nel file di configurazione di MyIonicApp.

4. Una volta terminato il collegamento, carica l'app eseguendo

```
$ ionic upload
```

Nota

Una volta che il caricamento ha avuto successo, apri la vista ionica sul tuo cellulare per visualizzare l'app.

Altri possono visualizzare la tua app, inviando l'ID app nella sezione 'Anteprima di un'app' in vista ionica.

Leggi Aggiungi l'app ionica alla vista ionica online:
https://riptutorial.com/it/ionic2/topic/10542/aggiungi-l-app-ionica-alla-vista-ionica

Capitolo 4: Angularfire2 con Ionic2

introduzione

Qui mal mostriamo come integrare AngularFire2 e utilizzare questo database in tempo reale nella nostra app Ionic.

Examples

Inizializzazione AngularFire

Prima di tutto è necessario inizializzare i moduli angularfire nel modulo dell'app in questo modo:

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

Puoi ottenere queste chiavi firmando firebase e creando un nuovo progetto.

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

Utilizzando AngularFire2

Una volta installato sulla tua app, è sufficiente importarlo:

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

Con questo elenco osservabile puoi accedere a un elenco di elementi sotto un percorso, ad esempio se hai root / articoli / cibo puoi ottenere articoli alimentari come questo:

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

E puoi semplicemente inserire un nuovo elemento qui e comparirà nel tuo database di Firebase, oppure puoi aggiornare un elemento e lo vedrai aggiornato sul tuo database. Puoi spingere e aggiornare in questo modo:

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

Oppure puoi anche oggetti remoti dalla tua lista di cibo:

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

Leggi Angularfire2 con Ionic2 online: https://riptutorial.com/it/ionic2/topic/10918/angularfire2-con-ionic2

Capitolo 5: Componenti CSS Ionic2

Examples

Griglia

Il sistema di griglia di Ionic è basato su flexbox, una funzione CSS supportata da tutti i dispositivi supportati da Ionic. La griglia è composta da tre unità: griglia, righe e colonne. Le colonne si espandono per riempire la loro riga e verranno ridimensionate per adattarle a colonne aggiuntive.

Classe	Larghezza
larghezza-10	10%
larghezza-20	20%
larghezza-25	25%
larghezza-33	33,3333%
larghezza-50	50%
larghezza-67	66,6666%
larghezza-75	75%
larghezza-80	80%
larghezza-90	90%

Esempio.

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

Carte

Le carte sono un ottimo modo per visualizzare parti importanti di contenuti e stanno rapidamente emergendo come un modello di progettazione principale per le app. Sono un ottimo modo per contenere e organizzare le informazioni, impostando anche aspettative prevedibili per l'utente. Con così tanti contenuti da mostrare in una sola volta, e spesso così poco spazio sullo schermo, le carte sono diventate rapidamente il modello di design preferito da molte aziende.

Esempio.

```
<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>
```

Leggi Componenti CSS Ionic2 online: https://riptutorial.com/it/ionic2/topic/8011/componenti-css-ionic2

Capitolo 6: Costruttore e OnInit

introduzione

Per quanto riguarda ionic2 il `constructor` : in termini semplici lo usiamo per creare istanze dei nostri plugin, servizi, ecc. Ad esempio: hai una pagina (vista) dove vuoi mostrare la lista di tutti gli studenti, e hai un file json che contiene tutti gli studenti (questo file è il tuo file di dati) quello che devi fare è creare un servizio in questo servizio, creerai un metodo e clicchi su una richiesta http.get per ottenere i dati json, quindi qui hai bisogno di cosa? http semplicemente in questo modo:

Examples

Esempio di Metodo Servizio Studente per l'utilizzo di Http nel costruttore

```
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)
        }
    }
```

Notare di nuovo il costruttore, se vogliamo utilizzare questo metodo di servizio, andremo alla nostra vista / pagina e:

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

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

Notate di nuovo il costruttore qui, stiamo creando un'istanza di StudentService nel costruttore e un'altra cosa, stiamo usando il plugin socialSharing così da usare che stiamo creando anche un'istanza di questo nel costruttore.

ngOnInit metodo per ottenere l'elenco degli studenti sul carico di visualizzazione

`OnInit` : questa è davvero una cosa incredibile in ionic2 o possiamo dire in AngularJs2. Con lo stesso esempio sopra possiamo vedere cosa è ngOnInit. Quindi sei pronto con il metodo di servizio, ora nella tua vista / pagina vuoi che i dati dell'elenco degli studenti siano disponibili non appena appare la tua vista, questa dovrebbe essere la prima operazione che avviene automaticamente al caricamento, perché quando la vista carica lo studente la lista dovrebbe

essere visibile Quindi la classe implementa OnInit e tu definisci ngOnInit. Esempio:

ngOnIn esempio per ottenere l'elenco degli studenti sulla pagina / vista

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

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

Leggi Costruttore e OnInit online: https://riptutorial.com/it/ionic2/topic/9907/costruttore-e-oninit

Capitolo 7: Dal codice all'app store - Android

introduzione

Troverai istruzioni passo passo su come preparare e caricare l'app di produzione ionica su Google Play.

Examples

Produzione pronta

Creazione del progetto dell'app

Quando crei un'app per Android pronta per l'app store è importante quando utilizzi `ionic start` che aggiungiamo i `--appname|-a` e `--id|-i` che vengono utilizzati da google play per identificare la tua app da altre app.

Se stai iniziando un nuovo progetto di app per dispositivi mobili, puoi utilizzare l'esempio cli di seguito.

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

1. File di configurazione dell'app

se si desidera impostare queste informazioni all'interno di un'app esistente, è possibile modificare `config.xml` . Raccomando a chi ha utilizzato il comando sopra per modificare anche `config.xml` .

Conferma / modifica `widget id` , `name` , `description` e attributi `author` .

Esempio:

```
<?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. icona e schermata iniziale

Entrambi i tipi di file supportati da icone e splash sono png, psd o ai e devono avere un nome file corrispondente a quello che è `icon` o `splash` e posizionato sotto la directory delle risorse nella radice del progetto. Le dimensioni minime dell'immagine dell'icona dovrebbero essere 192x192 px e non dovrebbero avere angoli arrotondati. e lo splash screen è molto più complicato, quindi clicca qui per saperne di più. Tuttavia, le dimensioni minime dovrebbero essere 2208x2208 px.

se si dispone di file di icone per generare utilizzare questo comando `ionic resources --icon` se si dispone di file splash per generare utilizzare questo comando `ionic resources --splash`

3. Costruire app di produzione

Prima di creare l'app di produzione, rimuovere i dati di registro sensibili.

Per creare una versione di rilascio con tutte le ottimizzazioni predefinite in posizione, utilizzare il tag --release e --prod

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

Per un elenco completo delle ottimizzazioni disponibili è possibile visitare il repository @ ionic / app-scripts

4. Crea una chiave privata

Ora, dobbiamo firmare l'APK non firmato (`android-release-unsigned.apk`) ed eseguire su di esso un programma di allineamento per ottimizzarlo e prepararlo per l'app store. Se hai già una chiave di firma, salta questi passaggi e usa quello.

Successivamente, individua il tuo file APK non firmato `android-release-unsigned.apk` all'interno di project dir `/platforms/android/build/outputs/apk/` e usa il comando `keytools` che verrà utilizzato per firmare il nostro file apk. Puoi usare l'esempio qui sotto:

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

puoi trovare `my-release-key.keystore` nella tua directory corrente.

Generiamo la nostra chiave privata usando il comando keytool fornito con JDK. Se questo strumento non viene trovato, consultare la guida all'installazione:

Prima verrà richiesto di creare una password per il keystore. Quindi, rispondi alle restanti domande degli strumenti utili e, una volta terminato, dovresti avere un file chiamato my-release-key.keystore creato nella directory corrente.

Nota: assicurati di salvare questo file in un posto sicuro, in caso di smarrimento non potrai inviare aggiornamenti alla tua app!

5. Firma l'APK

Per firmare l'APK non firmato, esegui lo strumento jarsigner che è anche incluso nel JDK:

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

Questo firma l'apk sul posto. Infine, abbiamo bisogno di eseguire lo strumento di allineamento zip per ottimizzare l'APK. Lo strumento zipalign può essere trovato in / percorso / a / Android / sdk / build-tools / VERSION / zipalign.

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

Ora abbiamo il nostro binario della versione finale chiamato HelloWorld.apk e possiamo pubblicarlo sul Google Play Store per far divertire tutto il mondo!

Pubblica la tua app su Google Play Store. Ora che abbiamo rilasciato APK per il Google Play Store, possiamo creare un elenco Play Store e caricare il nostro APK. Per iniziare, devi visitare la Console per gli sviluppatori di Google Play Store e creare un nuovo account sviluppatore. Costerà $ 25 una tantum.

Una volta che hai un account sviluppatore, puoi andare avanti e fare clic su "Pubblica un'app Android su Google Play" e seguire le istruzioni sullo schermo.

Leggi Dal codice all'app store - Android online: https://riptutorial.com/it/ionic2/topic/9659/dal-codice-all-app-store---android

Capitolo 8: geolocalizzazione

Examples

Uso semplice

Nel `package.json` assicurati di includere le dipendenze:

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

Per utilizzare la geolocalizzazione:

```
// 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);
        });
    }
}
```

Guardando la posizione

Per una soluzione più real time è possibile utilizzare la funzione watchPosition in geolocalizzazione che notifica ogni volta che si verifica un errore o un cambiamento di posizione. A differenza di getCurrentPosition, watchPosition restituisce un 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);

        });
}
```

Leggi geolocalizzazione online: https://riptutorial.com/it/ionic2/topic/5840/geolocalizzazione

Capitolo 9: InAppBrowser

introduzione

A volte il client richiede semplicemente di aprire un'app Web nell'app mobile, per questo possiamo usare InAppBrowser in modo tale che sembri un'app, invece stiamo aprendo un sito web / webApp in mobile, non appena l'utente toccherà l'icona dell'app che invece di aprire la prima vista dell'app, possiamo aprire direttamente InAppBrowser.

Examples

Un esempio vivo di questo utilizzo è questa app:

In questa app sto aprendo direttamente InAppBrowser quando l'utente tocca l'icona dell'app invece di caricare la prima pagina dell'app. In questo modo sembrerebbe all'utente che stia visualizzando l'app dello stesso sito web / webapp.

Esempio di codice per utilizzare 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();
        }
      );
```

Leggi InAppBrowser online: https://riptutorial.com/it/ionic2/topic/9801/inappbrowser

Capitolo 10: Installazione e debug di Ionic 2 in Visual Studio Code

introduzione

Visual Studio è un IDE open source che fornisce funzionalità intellisense e di modifica per il codice. Questo IDE supporta molti linguaggi come (Ionic, C, C #, AngularJs, TypeScript, Android e così via). Questi linguaggi sono in grado di eseguire tale codice aggiungendo le sue estensioni in VSCode. Usando VSCode siamo in grado di eseguire ed eseguire il debug del codice di diverse lingue diverse.

Examples

Installazione di VSCode

In primo luogo è necessario scaricare e installare il VSCode. Questa ultima versione di VSCode è disponibile per il download nel suo sito Web ufficiale . Dopo aver scaricato il VSCode, devi installarlo e aprirlo.

```
Introduction of Extensions in VSCode
```

VSCode è un editor aperto in modo da fornire l'editor per tutte le lingue, ma per eseguire un codice è necessario aggiungere l'estensione per quella particolare lingua. Per eseguire e modificare il tuo codice ionico devi aggiungere l'estensione **ionic2-vscode** nel tuo VSCode. Nella parte sinistra di VSCode Editor ci sono 5 icone in cui viene utilizzata l'icona più bassa per l'estensione. Le estensioni che potresti ottenere usando il **tasto di scelta rapida (ctrl + maiusc + X)** .

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

Premendo **ctrl + MAIUSC + X** hai mostrato la parte dell'estensione in cui sono indicati i primi **tre punti ...** questi punti sono noti come più icone. Cliccando su di esso si apre una finestra di dialogo che mostra il numero di opzioni da scegliere. scegli l'opzione secondo le tue necessità ma per ottenere tutta l'estensione dovresti selezionare **Mostra l'estensione consigliata** .iN l'elenco di tutte le estensioni che potresti installare Extension `(ionic2-vscode),npm`

Crea e aggiungi il tuo progetto Ionic in VSCode

VsCode non è in grado di creare il progetto ionico perché è un editor di codice. Puoi creare il tuo progetto ionico tramite **CLI** o **cmd** . crea il tuo progetto con il comando di sotto

```
$ ionic start appName blank
```

Sopra comando utilizzare per creare un'applicazione ionica modello vuoto. Ionic2 fornisce tre tipi di modelli **vuoti, tab e sidemenu** . Così. puoi sostituire il modello vuoto con altri due modelli secondo le tue necessità.

Ora, il tuo progetto Ionic è stato creato. Quindi, puoi aggiungere il tuo progetto in VSCode per modificarlo. Per aggiungere il tuo progetto segui i seguenti punti.

1. Vai al menu **File** in VScode.
2. Fare clic sulla **cartella Apri** nel menu File.
3. Trova e apri la cartella del tuo progetto.

   È possibile aprire direttamente la cartella utilizzando il tasto di scelta rapida **ctrl + O** o **ctrl + k**

`

Esegui e fai il debug del tuo progetto ionico

> Esegui ed esegui il debug in Chrome

Per eseguire il progetto ionico utilizzare sotto il comando in **terminale o cmd o CLI**

```
$ ionic serve
```

Per eseguire il **debug** del progetto ionico, in primo luogo è necessario aggiungere un'estensione **(Debugger per Chrome)** e quindi configurare il file launch.json in questo modo.

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

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

> ***Esegui ed esegui il debug in Android***

Per il progetto **Run** ionic in Android è necessario aggiungere la piattaforma Android tramite il comando di sotto nel terminale o cmd o CLI:

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

Costruisci Android con questo comando

```
$ ionic cordova build android
```

Esegui il comando per la piattaforma Android

```
$ ionic cordova run android
```

Ora, la tua applicazione funziona su un vero dispositivo Android.

Per eseguire il debug nel dispositivo Android è necessario aggiungere **Cordova o Android Extension** in VSCode. e configura il file launch.json in questo modo.

```
{
    "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}"
            }
]
}
```

Dopo la configurazione si seguono i seguenti passi o i tasti di scelta rapida per il debug:

1. Vai al menu di **debug** .
2. Fai clic su **Avvia debug** .

   o

```
Short keys
```

- Debugging - F5
- StepOver - F10
- Entrate e uscite - F11
- Arresta il debugging - Maiusc + F5
- Riavvia debug -ctrl + shift_F5

Leggi Installazione e debug di Ionic 2 in Visual Studio Code online: https://riptutorial.com/it/ionic2/topic/10559/installazione-e-debug-di-ionic-2-in-visual-studio-code

Capitolo 11: Invia notifica inviata e ricevi

Osservazioni

Il SenderID presente nell'esempio di inizializzazione è un ID mittente gcm che ti viene dato da google. Dovrebbe anche essere presente quando si installa il plugin

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

Se desideri aggiungere ulteriori dati alle tue notifiche push, guarda in questo link spiegando come aggiungere altri tipi di digitazione https://github.com/phonegap/phonegap-plugin-push/blob/master/docs/TYPESCRIPT.md

Examples

Inizializzazione

Il plugin di notifica push richiede un'inizializzazione di inizializzazione che indica al plug-in di iniziare a funzionare utilizzando l'id del mittente fornito.

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

Registrazione

La fase di registrazione registra l'app con il sistema del dispositivo e restituisce un ID di registrazione

```
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);

       });
```

Ricevere una notifica push

Per ricevere le notifiche push dovremmo dire al plugin di ascoltare le notifiche push in arrivo. Questo passaggio viene eseguito dopo l'inizializzazione e la registrazione

```
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));
       });
```

Leggi Invia notifica inviata e ricevi online: https://riptutorial.com/it/ionic2/topic/5874/invia-notifica-inviata-e-ricevi

Capitolo 12: Modals

Examples

Uso dei modali

Le modalitÃ scivolano fuori dallo schermo per visualizzare un'interfaccia utente temporanea, spesso utilizzata per le pagine di accesso o di registrazione, la composizione dei messaggi e la selezione delle opzioni.

```
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();
  }
}
```

NOTA: A Modal è un riquadro del contenuto che passa sopra la pagina corrente dell'utente.

Trasmissione dei dati attraverso un modale

I dati possono essere passati a una nuova modale tramite `Modal.create()` come secondo argomento. È possibile accedere ai dati dalla pagina aperta iniettando `NavParams` . Si noti che la pagina, che è stata aperta come modale, non ha una speciale logica "modale" al suo interno, ma utilizza `NavParams` diverso rispetto a una pagina standard.

Prima pagina:

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

export class HomePage {

 constructor(public modalCtrl: ModalController) {

 }

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

}
```

Seconda pagina:

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

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

}
```

Leggi Modals online: https://riptutorial.com/it/ionic2/topic/6415/modals

Capitolo 13: Modals

Examples

Modale semplice

Modale è un'interfaccia utente temporanea che viene visualizzata nella parte superiore della pagina corrente. Viene spesso utilizzato per l'accesso, l'iscrizione, la modifica delle opzioni esistenti e la selezione delle opzioni.

Esaminiamo un semplice esempio con le modali utilizzate. Per cominciare stiamo creando un progetto bianco e ionico. Cerchiamo di creare un modale semplice che mostri un messaggio e di uscire al clic del pulsante. Per farlo, per prima cosa creiamo la vista per il nostro 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();
  }
}
```

Questa modale mostra un messaggio. Il modale può essere chiuso o “respinto” utilizzando la

visualizzazione controller metodo di **respingere.**

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();
  }
}
```

Ora stiamo creando la nostra home page importando **ModalController** e il nostro modello di dati MessagePage. Il metodo di **creazione** di ModalController crea modali per il nostro modello di dati MessagePage che viene salvato per controllare la variabile myModal. **Il** metodo **attuale** apre la modale sopra la nostra pagina corrente.

Modale con parametri in chiusura:

Ora sappiamo come creare un modale. Ma cosa succede se vogliamo passare alcuni dati dalla modale alla nostra home page. Per fare ciò, esaminiamo un esempio con modale come Pagina di registro che passa i parametri alla pagina padre.

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);
  }
```

```
}
```

Il registro modale ottiene l'oggetto dati con i valori inseriti dall'utente e i parametri vengono passati alla nostra pagina corrente in congedo con il metodo di licenziamento viewControllers. Ora i parametri sono inviati.

Quindi, come recupereremo i parametri nella home page? Per fare ciò, stiamo creando un pulsante sulla home page e chiamiamo Register modal al click. Per visualizzare l'utente, stiamo visualizzando un elenco.

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);
      }
    });
  }
```

Il metodo **ViewController onDidDismiss** viene eseguito ogni volta che viene chiusa una modale. Se i dati vengono passati come parametro da modale, possiamo recuperarli utilizzando il metodo onDidDismiss. Qui i dati inseriti dall'utente vengono aggiunti ai dati esistenti. Se nessun dato viene passato come parametro, il valore restituito sarà nullo.

Modale con parametri su create:

Passare i parametri a una modale è simile a come si passano i valori a un NavController. Per fare ciò, stiamo modificando la nostra lista in home.html per aprire una modale quando si fa clic su una voce di elenco e si passano i parametri richiesti come secondo argomento al metodo di **creazione** .

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();
  }
```

Simile ad altre viste, usiamo NavParams per recuperare i dati inviati dalla vista precedente.

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();
  }
}
```

Leggi Modals online: https://riptutorial.com/it/ionic2/topic/6612/modals

Capitolo 14: Soluzione alternativa per "mostra-elimina" in deprecazione

Examples

Soluzione

Sto sviluppando un'applicazione mobile con ionic 2 con Angular 2.

Ho una lista di ioni riempita di ioni. Voglio che quegli oggetti ionici abbiano la possibilità di essere cancellati, se necessario, come presentato qui sul sito web ionico.

Tuttavia, molti sono cambiati in **ionico 2** poiché la prima versione e lo stile sopra di un pulsante che apre tutti gli elementi **ionici** in uno non è più possibile poiché lo **show-delete** e lo **show-reorder** non sono più supportati. L'unica opzione disponibile è avere **un oggetto ion-sliding** come elemento ionico, che ci dà la possibilità di far scorrere ogni oggetto uno alla volta per rivelare il pulsante cancella.

Non è quello che volevo. Volevo un pulsante che apra tutto l'oggetto ionico allo stesso tempo.

Dopo aver dedicato un po 'di tempo a questo, ho trovato una soluzione funzionante e sono riuscito a ottenere il risultato desiderato usando ionic 2, e lo condividerò con voi.

Ecco la mia soluzione:

Nel file .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>
```

e per la lista:

```
<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>
```

Nel file **.ts** , per prima cosa importa le tue importazioni:

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

quindi fai riferimento all'elemento html dichiarando un ViewChild:

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

Infine, aggiungi le tue classi per gestire il lavoro:

```
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;
    }
}
```

Quindi la classe di apertura:

```
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)")
}
```

E il corso di chiusura:

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

}

Spero che possa aiutarti un po 'là fuori.

Goditi e buona codifica ...

Leggi Soluzione alternativa per "mostra-elimina" in deprecazione online: https://riptutorial.com/it/ionic2/topic/6620/soluzione-alternativa-per--mostra-elimina--in--ion-list--deprecazione

Capitolo 15: Test unitario

introduzione

Il test unitario in generale offre maggiore sicurezza a un prodotto per prevenire problemi durante la modifica / aggiunta di funzionalità. Una rete di sicurezza che dice "TUTTO SEMPRE FUNZIONA". I test delle unità non sostituiscono in alcun modo l'utente effettivo che un QA appropriato può eseguire.

In questo documento baseremo gli esempi su questo repository: https://github.com/driftyco/ionic-unit-testing-example

Examples

Test unitari con Karma / Jasmine

Il test unitario in ionico è lo stesso di qualsiasi app angolare.

Useremo alcuni framework per farlo.

Karma: una struttura per eseguire test

Jasmine: una struttura per i test di scrittura

PhantomJS - un'applicazione che esegue javascript senza browser

Prima di tutto, installiamo tutto, quindi assicurati che il tuo pacchetto.json includa queste linee nelle dipendenze di sviluppo. Sento che è importante notare che le dipendenze degli sviluppatori non influenzano affatto la tua app e sono lì solo per aiutare lo sviluppatore.

```
"@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"
```

Ripassare un po 'i pacchetti

```
"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.
```

Dovremmo anche aggiungere questo script ai nostri script package.json:

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

Prendere nota anche in tsconfig che si stanno escludendo i file spec.ts dalla compilazione:

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

Ok, ora prendiamo la configurazione di test effettiva. Creare una cartella `test-config` nella cartella del progetto. (Proprio come è stato menzionato nello script package.json) All'interno della cartella crea 3 file:

`webpack.test.js` - che dirà al webpack quali file caricare per il processo di 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` - che caricherà le librerie angolari, come le librerie di zone e di test, e configurerà il modulo per il 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` - definisce la configurazione di come testare il karma. Qui puoi passare da Chrome a PhantomJS per rendere questo processo invisibile e più veloce tra le altre cose.

```
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);
};
```

Ora che abbiamo configurato tutto, scriviamo alcuni test effettivi. Per questo esempio scriveremo un file spec app.component. Se desideri vedere i test per una pagina e non il componente principale, puoi guardare qui: https://github.com/driftyco/ionic-unit-testing-example/blob/master/src/pages/page1/page1. spec.ts

Quello che dobbiamo fare prima è testare il nostro costruttore. Questo creerà ed eseguirà il costruttore del nostro app.component

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

La dichiarazione includerà la nostra app ionica principale. Le importazioni saranno le importazioni necessarie per questo test. Non tutto.

I provider includeranno le cose che sono state iniettate nel costruttore ma che non fanno parte dell'importazione. Ad esempio, app.component inietta il servizio Piattaforma ma, poiché è parte di IonicModule, non è necessario menzionarlo nei provider.

Per i prossimi test avremo bisogno di ottenere un'istanza del nostro componente:

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

Successivamente alcuni test per vedere che tutto è in ordine:

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

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

Quindi alla fine avremo qualcosa di simile a questo:

```
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);
  });

});
```

Esegui i test di

```
npm run test
```

E questo è tutto per il test di base. Ci sono alcuni modi per collegare la scrittura del test come scrivere il tuo TestBed e avere ereditarietà nei test che potrebbero aiutarti nel lungo periodo.

Leggi Test unitario online: https://riptutorial.com/it/ionic2/topic/9561/test-unitario

Capitolo 16: Utilizzando le schede

Osservazioni

Ricordarsi sempre di controllare i documenti di Ionic 2 Tab per essere al corrente delle ultime modifiche e aggiornamenti.

Examples

Cambia la scheda selezionata in modo programmatico dalla Pagina secondaria

Puoi dare un'occhiata al codice completo in questo Plunker funzionante .

In questo esempio, utilizzo un servizio condiviso per gestire la comunicazione tra le pagine all'interno della scheda (pagine figlio) e il contenitore delle schede (il componente che contiene le schede). Anche se probabilmente potresti farlo con Eventi, mi piace l'approccio al servizio condiviso perché è più facile da capire e anche da mantenere quando l'applicazione inizia a crescere.

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);
  }
}
```

Quindi, in pratica `TabService` crea solo un `Observable` per consentire al contenitore di schede di sottoscriverlo e dichiara anche il metodo `changeTabInContainerPage()` che verrà chiamato dalle pagine `changeTabInContainerPage()` .

Quindi, in ogni pagina figlio (quelle all'interno delle schede) aggiungiamo solo un pulsante e associamo l'evento `click` a un metodo che chiama il servizio:

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);
  }
}
```

E infine, in `TabsPage` , sottoscriviamo solo il servizio, quindi cambiamo la scheda selezionata con `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);
    });
  }
}
```

Si prega di notare che stiamo ottenendo un riferimento all'istanza Tabs aggiungendo `#myTabs` nell'elemento `ion-tabs` , e lo otteniamo dal componente con `@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>
```

Cambia scheda con selectedIndex

Invece di ottenere un riferimento al DOM, puoi semplicemente cambiare l'indice della scheda usando l'attributoIndice selezionato nelle schede 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);
 }
}
```

Se vuoi cambiarlo da qualche altro servizio controller puoi inviare un evento:

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

Leggi Utilizzando le schede online: https://riptutorial.com/it/ionic2/topic/5569/utilizzando-le-schede

Capitolo 17: Utilizzo dei servizi

Osservazioni

Una cosa molto importante sull'utilizzo dei servizi condivisi è che devono essere inclusi nella matrice dei `providers` del componente più in alto in cui devono essere condivisi.

Perché? Bene, supponiamo di includere il riferimento a `MyService` nell'array `providers` da ciascun `Component` . Qualcosa di simile a:

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

E

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

In questo modo **verrà creata una nuova istanza del servizio per ciascun componente,** pertanto l'istanza in cui una pagina salverà i dati sarà diversa dall'istanza utilizzata per ottenere i dati. Quindi non funzionerà.

Per fare in modo che l'intera app utilizzi la stessa istanza (facendo in modo che il servizio funzioni come servizio *singleton*), possiamo aggiungere il suo riferimento nella `App Component` questo modo:

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

È anche possibile aggiungere il riferimento `MyService` in `ionicBootstrap(MyApp, [MyService]);` ma secondo le guide di stile Angular2

> Fornisci servizi all'iniettore Angular 2 nella parte più alta in cui saranno condivisi.
>
> Perché? L'iniettore Angular 2 è gerarchico.
>
> Perché? Quando si fornisce il servizio a un componente di livello superiore, tale istanza è condivisa e disponibile per tutti i componenti figlio di quel componente di livello superiore.
>
> Perché? Questo è l'ideale quando un servizio sta condividendo metodi o stato.

Perché? Questo non è l'ideale quando due diversi componenti richiedono istanze diverse di un servizio. In questo scenario sarebbe meglio fornire il servizio a livello di componente che necessita della nuova istanza separata.

E

Funzionerà. Non è solo una buona pratica. **L'opzione del provider di bootstrap è intesa per la configurazione e l'override dei servizi preregistrati di Angular** , come il supporto del routing.

... la `App Component` sarebbe la scelta migliore.

Examples

Condividi le informazioni tra pagine diverse

Uno degli esempi più semplici di utilizzo dei *servizi condivisi* è quando vogliamo archiviare alcuni dati da una determinata pagina della nostra applicazione, e quindi ottenere di nuovo quei dati ma da un'altra pagina.

Un'opzione potrebbe essere quella di inviare quei dati come parametro (ad esempio, se una pagina chiama l'altro) ma se vogliamo usare quei dati da una parte completamente diversa dell'applicazione, questo sembra non essere il modo migliore di fare esso. Questo è quando *i servizi condivisi* vengono a giocare.

In questo esempio, useremo un servizio semplice chiamato `MyService` che ha solo due semplici metodi: `saveMessage()` per memorizzare una stringa e `getMessage()` per ottenerlo di nuovo. Questo codice fa parte di questo plunker funzionante in cui è possibile vederlo in azione.

```
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;
  }
}
```

Quindi, quando vogliamo memorizzare un nuovo messaggio, possiamo semplicemente usare `saveMessage(theMessageWeWantToSave);` metodo `MyService` (chiamato solo `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);
  }
}
```

Allo stesso modo, quando vogliamo ottenere quei dati possiamo usare il metodo `getMessage()` del servizio in questo modo:

```
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();
  }

  // ...
}
```

Ricordarsi di controllare la sezione *Osservazioni* per vedere dove dovrebbe essere incluso il riferimento per il servizio `MyService` e perché.

Leggi Utilizzo dei servizi online: https://riptutorial.com/it/ionic2/topic/4407/utilizzo-dei-servizi

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con ionic2	Akilan Arasu, Cameron637, carstenbaumhoegger, Community, FreeBird72, Guillaume Le Mière, Ian Pinto, Ketan Akbari, misha130, Raymond Ativie, sebaferreras, tymspy, Will.Harris
2	Accesso sociale con Angularfire2 / Firebase	Cameron637, Gianfranco P.
3	Aggiungi l'app ionica alla vista ionica	Saravanan Sachi
4	Angularfire2 con Ionic2	Fernando Del Olmo
5	Componenti CSS Ionic2	Ketan Akbari
6	Costruttore e OnInit	niks
7	Dal codice all'app store - Android	Luis Estevez, misha130
8	geolocalizzazione	Matyas, misha130
9	InAppBrowser	niks
10	Installazione e debug di Ionic 2 in Visual Studio Code	misha130, PRIYA PARASHAR
11	Invia notifica inviata e ricevi	misha130
12	Modals	Raymond Ativie
13	Soluzione alternativa per "mostra-elimina" in deprecazione	Amr ElAdawy, Roman Lee
14	Test unitario	misha130
15	Utilizzando le schede	misha130, sebaferreras

16	Utilizzo dei servizi	sebaferreras