Programação Móvel

Atualização Ionic

Prof. Dr. Diego R. Antunes
drantunes@utfpr.edu.br

Departamento de Computação
Universidade Tecnológica Federal do Paraná

Updates

Updates

- RC.0 (28/09/16) : diversas breaking changes;
- RC.1 (13/10/16) : correção de bugs;
- RC.2 (03/11/16) : correção de bugs e performance;
- RC.3 (17/11/16) : correção de bugs e performance;

Atualização

Atualização

A principal atualização, com breaking changes, consiste em transformar a versão Beta para RC.0

https://github.com/driftyco/ionic/blob/master/CHANGELOG.md

Mudançcas

Estrutura de Pastas

A pasta app foi movida para dentro de src

Estrutura de Pastas

Na pasta src foi adicionada a pasta assets. Esta pasta serve para adicionar arquivos como imagens e ícones. No processo de compilação ela é movida para o www.

Estrutura de Pastas

Na pasta src foi adicionada a pasta theme. Esta pasta agora só disponibiliza um arquivo css. Você pode fazer imports ou sobrecrever as cores padrão do Ionic.

CSS de Componentes

Agora, o Ionic carrega automaticamente o CSS de cada nova Page. Ou seja, não é necessário fazer imports manualmente.

app.ts

O arquivo app.ts foi transformado no arquivo app.modules.ts e app.component.ts dentro da pasta app.

TS app.component.ts

```ts
import { Component } from '@angular/core';
import { Platform } from 'ionic-angular';
import { StatusBar, Splashscreen } from 'ionic-native';

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


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

  constructor(platform: Platform) {
    platform.ready().then(() => {
      // Okay, so the platform is ready
      // Here you can do any higher lev
      StatusBar.styleDefault();
      Splashscreen.hide();
    });
  }
```

O arquivo app.component é o componente inicial que o ionic irá carregar.

TS app.component.ts

```
1  import { Component } from '@angular/core';
2  import { Platform } from 'ionic-angular';
3  import { StatusBar, Splashscreen } from 'ionic-native';
4
5  import { HomePage } from '../pages/home/home';
6
7
8  @Component({
9    templateUrl: 'app.html'
10 })
11 export class MyApp {
12   rootPage = HomePage;
13
14
15
16
17
18
19
20
21   }
```

Outra mudança importante: agora basta informar o nome do arquivo de template, que deve estar na mesma pasta.

TS app.module.ts

```
1  import { NgModule, ErrorHandler } from '@angular/core';
2  import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
3  import { MyApp } from './app.component';
4  import { HomePage } from '../pages/home/home';
5
6  @NgModule({
7    declarations: [
8      MyApp,
9      HomePage
10   ],
11   imports: [
12     IonicModule.forRoot(MyApp)
13   ],
14   bootstrap: [IonicApp],
15
16
17
18
19
20
21 export class AppModule {}
```

O arquivo app.module.ts serve para declarar e importar todos os módulos e componentes usados no sistema.

TS app.module.ts

```
1  import { NgModule, ErrorHandler } from '@angular/core';
2  import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
3  import { MyApp } from './app.component';
4  import { HomePage } from '../pages/home/home';
5
6  @NgModule({
7    declarations: [
8      MyApp,
9      HomePage
10   ],
11   imports: [
12     IonicModule.forRoot(MyApp)
13   ],
14   bootstrap: [IonicApp],
15
16
17
18
19
20
21 export class AppModule {}
```

Assim, o Angular conhece os módulos que precisa instanciar e tratar na injeção de dependência.

TS app.module.ts

```ts
import { NgModule, ErrorHandler } from '@angular/core';
import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
import { MyApp } from './app.component';
import { HomePage } from '../pages/home/home';

@NgModule({
  declarations: [
    MyApp,
    HomePage
  ],
  imports: [
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],

export class AppModule {}
```

O arquivo importa classes padrão do sistema e o novo sistema de erros.

TS app.module.ts

```
1  import { NgModule, ErrorHandler } from '@angular/core';
2  import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
3  import { MyApp } from './app.component';
4  import { HomePage } from '../pages/home/home';
5
6  @NgModule({
7    declarations: [
8      MyApp,
9      HomePage
10   ],
11   imports: [
12     IonicModule.forRoot(MyApp)
13   ],
14   bootstrap: [IonicApp],
15
16
17
18
19
20
21 export class AppModule {}
```

Em seguida, você deve importar todas as páginas e módulos do seu sistema.

Além disso, você deve declarar as páginas em

declarations e *entryComponents*

```
@NgModule({
  declarations: [
    MyApp,
    HomePage
  ],
  imports: [
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage
  ],
  providers: [{provide: ErrorHandler, useClass: IonicErrorHandler}]
})
export class AppModule {}
```

Neste arquivo também é possível declarar todos os seus providers (e.g. Firebase).

```
@NgModule({
  declarations: [
    MyApp,
    HomePage
  ],
  imports: [
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage
  ],
  providers: [{provide: ErrorHandler, useClass: IonicErrorHandler}]
})
export class AppModule {}
```

Pages

Algumas mudanças nas páginas geradas pelo Ionic:

color de componentes e tratamento de erros.

Agora ficou mais simples alterar a cor de componentes.

Basta usar o atributo color e especificar a cor.

home.html | TS home.ts

```
1  <ion-header>
2    <ion-navbar color="primary">
3      <ion-title>
4        Ionic RC3
5      </ion-title>
6    </ion-navbar>
7  </ion-header>
8
9  <ion-content padding>
10 </ion-content>
11
```

Ou seja, agora é possível realizar o binding de uma variável definida na classe.

home.html

```
<ion-header>
  <ion-navbar [color]="variavel">
    <ion-title>
      Ionic RC3
    </ion-title>
  </ion-navbar>
</ion-header>
```

home.ts

```
export class HomePage {

  public variavel = 'secondary';

}
```

Color

Este atributo pode ser usado nos componentes de Badge, Button, Checkbox, Chip, FAB, Icon, Item, Label, Navbar, Spinner, Tabs, Toggle, Toolbar e outros

Error

A maioria de erros de sintaxe agora são exibidos no navegador, resolvendo parcialmente a questão da ”tela branca do mal"

Error

Close

Typescript Error

Cannot find name 'NavControlle'.

src/pages/home/home.ts

```
12
13    constructor(public navCtrl: NavControlle) {
14
```

```
Ionic Framework: 2.0.0-rc.3
Ionic Native: 2.2.3
Ionic App Scripts: 0.0.45
Angular Core: 2.1.1
Angular Compiler CLI: 2.1.1
Node: 6.8.0
OS Platform: macOS Sierra
Navigator Platform: MacIntel
User Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, li
```

Buttons

Nesta versão o Ionic introduziu a diretiva ion-button.

Agora é possível usar os estilos padrão do ionic ou customizar os botões com um estilo próprio.

Buttons

Ou seja, todo button deve adicionar ion-button.

```
9   <ion-content padding>
10
11    <button ion-button block color="primary">
12      Botão
13    </button>
14
15  </ion-content>
16
```

Icon Buttons

Também foram criadas algumas diretivas para botões com ícones: icon-only, icon-left, icon-right. Para exemplos e mais detalhes acessar este link

Storage

Storage

O Storage agora foi movido para um pacote separado.

Para instalar você deve usar:

```
npm install @ionic/storage --save --save-exact
```

Storage

E deve importar no app.modules.ts e adicionar no array de providers.

Storage

PORÉM: Este módulo não é mais compatível com o SQLite. Agora ele tem o propósito de salvar dados no formato chave-valor somente.

Storage

Assim, para usar o SQLite, você deve importar o módulo Nativo ou criar um provider que mapeie o WebSQL ou SQLite Nativo. Dica neste link.

Como atualizar?

Duas Maneiras

1) Criar um novo projeto e copiar os arquivos antigos;

2) Modificar um projeto existente;

Criar um novo projeto

Esta é a maneira mais rápida e simples de atualizar seu projeto para a versão mais atual. O tutorial passo a passo encontra-se neste link.

Criar um novo projeto

Basicamente você precisa atualizar o ionic em seu computador para a versão mais atual. Em seguida, criar um projeto normalmente e fazer a migração.

Criar um novo projeto

As versões RC.1, RC.2 e RC.3 são compatíveis com o tutorial apresentado para migração, ou seja, não há breaking change.

Dicas Finais

Dicas

1) Acompanhe o blog oficial do Ionic

2) Acompanhe o github oficial e o changelog.

Avisos