Guía de migración de aplicaciones angular de RC4 a RC5

24 de Agosto de 2016

Inscripción Registro Mercantil de Pontevedra, Tomo 3116, Libro 3116, Folio 30, Hoja PO-38276
C.I.F.: B-36.499.960 · contact@imatia.com · www.ontimize.com

Índice de contenidos

1 Introducción

En este documento se explica paso a paso como actualizar una aplicación web realizada con Ontimize Web con la versión angular2 RC4 (se corresponde con la versión 0.0.3 del framework) a la versión angular2 RC5.

1. Actualizar el número de versión en el fichero *package.json* (2.0.0-rc.5).
2. Actualizar el proceso de 'bootstrapping' y envolver tu aplicación en un *NgModule*.
3. Mover todas las declaraciones de *directives* y *pipes* de tus componentes (@Component) al módulo principal de aplicación (@NgModule).
4. Actualizar el router.

Puedes obtener información complementaria en los siguientes enlaces:

- https://angular.io/docs/ts/latest/cookbook/rc4-to-rc5.html
- https://www.barbarianmeetscoding.com/blog/2016/08/13/updating-your-angular-2-app-from-rc4-to-rc5-a-practical-guide/

2 Actualización de versión

Actualizar en el fichero **`package.json`** las siguientes versiones:

- Módulos estándar angular2 de `2.0.0-rc.4` a `2.0.0-rc.5`
- Módulos de componentes de `v2.0.0-alpha.6` a `v2.0.0-alpha.7-3` (añadir módulos nuevos de componentes).
- Módulo angular2/forms de `0.2.0` a `0.3.0`
- Módulo angular2/router de `3.0.0-beta.2` a `3.0.0-rc.1`
- Módulo ng2-translate de 2.2.2 a `2.4.1`
- Módulo ng2-material (tag versión v0.6.1)

El resultado del archivo debe ser como se muestra en el siguiente cuadro.

```
"@angular/common": "2.0.0-rc.5",
    "@angular/compiler": "2.0.0-rc.5",
    "@angular/core": "2.0.0-rc.5",
    "@angular/forms": "0.3.0",
    "@angular/http": "2.0.0-rc.5",
    "@angular/platform-browser": "2.0.0-rc.5",
    "@angular/platform-browser-dynamic": "2.0.0-rc.5",
    "@angular/router": "3.0.0-rc.1",
    "@angular/upgrade": "2.0.0-rc.5",
    "@angular2-material/button": "v2.0.0-alpha.7-3",
    "@angular2-material/button-toggle": "v2.0.0-alpha.7-3",
    "@angular2-material/card": "v2.0.0-alpha.7-3",
    "@angular2-material/checkbox": "v2.0.0-alpha.7-3",
    "@angular2-material/core": "v2.0.0-alpha.7-3",
    "@angular2-material/grid-list": "v2.0.0-alpha.7-3",
    "@angular2-material/icon": "v2.0.0-alpha.7-3",
    "@angular2-material/input": "v2.0.0-alpha.7-3",
    "@angular2-material/list": "v2.0.0-alpha.7-3",
    "@angular2-material/menu": "v2.0.0-alpha.7-3",
    "@angular2-material/progress-bar": "v2.0.0-alpha.7-3",
    "@angular2-material/progress-circle": "v2.0.0-alpha.7-3",
    "@angular2-material/radio": "v2.0.0-alpha.7-3",
    "@angular2-material/sidenav": "v2.0.0-alpha.7-3",
    "@angular2-material/slider": "v2.0.0-alpha.7-3",
    "@angular2-material/slide-toggle": "v2.0.0-alpha.7-3",
    "@angular2-material/tabs": "v2.0.0-alpha.7-3",
    "@angular2-material/toolbar": "v2.0.0-alpha.7-3",
    "@angular2-material/tooltip": "v2.0.0-alpha.7-3",
    "ng2-translate": "2.4.1",
    "ng2-material": "git+http://37.139.121.79:8082/gitbucket/git/daniel.grana/ng2-
material.git#v0.6.1",
```

NOTA:

Los nuevos componentes de módulos (`button-toggle, menú, slider y tooltip`) se deben añadir en el archivo de configuración **`system-config.ts.`**

```
'@angular2-material/button-toggle': {
    main: 'button-toggle.js',
    defaultExtension: 'js'
  },
'@angular2-material/menu': {
    main: 'menu.js',
    defaultExtension: 'js'
  },
'@angular2-material/slider': {
    main: 'slider.js',
    defaultExtension: 'js'
  },
'@angular2-material/tooltip': {
    main: 'tooltip.js',
```

Para realizar las actualizaciones de versiones se recomienda ejecutar la consola con permisos de administrador y previamente borrar las carpetas de las versiones antiguas contenidas en *node_modules*.

3 Bootstraping y NgModule

Se debe modificar el archivo **main.ts** para adaptarlo al nuevo proceso de 'bootstrapping'. El contenido del fichero será:

```
import { enableProdMode } from '@angular/core';
import { ontimizeBootstrap } from 'ontimize-web-ng2/ontimize/MainLauncher';
import { AppModule, environment, CONFIG } from './app/';

if (environment.production) {
  enableProdMode();
}

// Boostraping app...
ontimizeBootstrap(AppModule, CONFIG).then(appRef => {
  console.log('initialized... ');
});
```

Se debe añadir un nuevo fichero **app.module.ts** en la ruta *src/app*. Este fichero define el módulo de aplicación.
El decorador @NgModule tiene los siguientes parámetros:

- `imports`: Array de módulos que se van a utilizar en la aplicación. El framework proporciona una constante (`ONTIMIZE_MODULES`) que contiene todos los módulos estándar de angular2 (`FormsModule, HttpModule, etc`) así como los componentes material (`Button, Card, etc.`)
- `declarations`: Los componentes y directivas de la aplicación. El framework proporciona una constante (`ONTIMIZE_DIRECTIVES`) con todas las directivas por defecto. Además se deberán añadir todas las directivas de la aplicación.
- `bootstrap`: El componente raíz de la aplicación (`AppModule`).
- `providers`: Servicios que el usuario desea que estén disponibles en toda la aplicación. El framework proporciona un método que permite obtener los 'providers' estándar a partir de la configuración de aplicación (ver cuadro de texto). Además el usuario puede añadir sus propios providers a este array.

El contenido de este fichero es el siguiente:

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

import {
  ONTIMIZE_MODULES,
  ONTIMIZE_DIRECTIVES,
  ontimizeProviders } from 'ontimize-web-ng2/ontimize';

import { CONFIG } from './app.config';
import { AppComponent }  from './app.component';
import { routing } from './app.routes';
import { APP_DIRECTIVES } from './app.directives';

// Standard providers...
let standardProviders = ontimizeProviders({
  'config': CONFIG
});
// Defining custom providers (if needed)...
// let customProviders = [
// ];

@NgModule({
  imports: [ ONTIMIZE_MODULES, routing],
  declarations: [
    AppComponent,
    ONTIMIZE_DIRECTIVES,
    ...APP_DIRECTIVES
  ],
  bootstrap: [ AppComponent ],
  providers: [
    ...standardProviders
    // ...customProviders
  ]
})
export class AppModule { }
```

4 Directives y Pipes

Todas las declaraciones de `directives` y `pipes` que se hacían en los componentes (@Component) de la aplicación se tienen que eliminar y pasan a declararse en el módulo principal de la aplicación. Para ello es necesario crear un fichero de directivas de la aplicación ***app.directives.ts*** en la ruta *src/app*.
Un ejemplo del contenido de ese fichero es el siguiente:

```
import { LoginComponent } from './+login';
import { MainComponent } from './+main';
import { AboutComponent } from './+main/+about';
import { ACCOUNTS_DIRECTIVES } from './+main/+accounts';
import { CUSTOMERS_DIRECTIVES } from './+main/+customers';
import { EMPLOYEES_DIRECTIVES } from './+main/+employees';
import { HelpComponent } from './+main/+help';
import { HomeComponent } from './+main/+home';
import { SHARED_DIRECTIVES } from './shared';

// All directives of the application
export const APP_DIRECTIVES: any = [
  LoginComponent,
  MainComponent,
  AboutComponent,
  ACCOUNTS_DIRECTIVES,
  CUSTOMERS_DIRECTIVES,
  EMPLOYEES_DIRECTIVES,
  HelpComponent,
  HomeComponent,
  SHARED_DIRECTIVES
];
```

El resultado de este fichero es que tiene que devolver una constante (`APP_DIRECTIVES`) que contendrá todas las directivas de la aplicación.
Como se puede ver en el ejemplo, se pueden importar los componentes del módulo (por ej. *LoginComponent* del módulo *+login*) o, para mayor comodidad y organización del código, se pueden importar constantes que contengan todas las directivas del módulo (por ej. `CUSTOMERS_DIRECTIVES` del módulo *+customers*).

Se recomienda realizar el segundo modo de importación. Para ello se debe modificar el archivo ***index.ts*** del módulo en concreto y añadir:

```
export {CUSTOMERS_DIRECTIVES} from './customers.directives';
```

Y crear el archivo ***customers.directives.ts***.

```
import {
  CustomersComponent,
  CustomersHomeComponent,
  NewCustomerComponent,
  CustomersEditComponent,
  CustomersDetailComponent
} from '../+customers';

// For testing inheritance on forms...
import { CustomersDetailFormComponent } from './detail/customers-detail-form.component';
import { CustomersDetailSubFormComponent } from './detail/customers-detail-
subform.component';
import { CustomersEditFormComponent } from './edit/customers-edit-form.component';
import { NewCustomerFormComponent } from './new/new-customer-form.component';

export const CUSTOMERS_DIRECTIVES: any = [
  CustomersComponent,
  CustomersHomeComponent,
  NewCustomerComponent,
  CustomersEditComponent,
  CustomersDetailComponent,
  CustomersDetailFormComponent,
  CustomersDetailSubFormComponent,
  CustomersEditFormComponent,
  NewCustomerFormComponent
];
```

5 Router

Por último, hay que actualizar el fichero/s de definición de rutas de la aplicación. El fichero principal de definición de rutas ***app.routes.ts*** queda definido del modo que muestra el siguiente cuadro de texto.

Se deben renombrar las variables *RouterConfig* a *Routes* y el módulo debe exportar un módulo que deberá ser añadido en el parámetro `imports` del módulo de la aplicación (@NgModule).

```
import { Routes, RouterModule } from '@angular/router';

import { LoginComponent } from './+login';
import { HomeComponent } from './+main/+home';
import { HelpComponent } from './+main/+help';
import { AboutComponent } from './+main/+about';
import { PrivateAppRoutes } from './+main/main.routes';

const LoginRoutes: Routes = [
  { path: 'login', component: LoginComponent}];

const PublicAppRoutes: Routes = [
    { path: 'home', component: HomeComponent },
    { path: 'help', component: HelpComponent },
    { path: 'about', component: AboutComponent }];

// All routes of the application
const routes: Routes = [
  ...LoginRoutes,
  ...PublicAppRoutes,
  ...PrivateAppRoutes,
  { path: '', redirectTo: 'main', pathMatch: 'prefix' }
];

export const routing = RouterModule.forRoot(routes, { enableTracing: true });
```

NOTA:

En todos los ficheros parciales de definición de rutas (por ej. *main.routes.ts*) se debe renombrar la variable *RouterConfig* por *Routes*.

---

6 Otros cambios

Se debe modificar el archivo **app.component.ts** y eliminar la declaración de providers.

```
@Component({
  moduleId: module.id,
  selector: 'o-app',
  templateUrl: 'app.component.html',
  styleUrls: ['app.component.css'],
  ~~providers: [ DialogService ]~~
})
```