Curso de Desarrollo Web con Angular

Artículo

Creando nuestro primer componente en Angular

Uriel Ramírez 23 PlatziRank Feb. 17, 2017

Primeros pasos con Angular

Angular 2 nació de un equipo de ingenieros de Google, como un framework para construir aplicaciones universales, esto quiere decir aplicaciones que funcionen en web, desktop y móviles.

¿Qué cambio desde Angular.js a Angular 2?

A pesar que el enfoque sigue siendo Single Page Applications, Angular sufrió un par de cambios:

a) Su core ahora está construido con RxJS y TypeScript (Recuerda que tenemos una guía sobre este superset), que asegura su velocidad y modularidad. Ya no tendremos cambios grandes como lo fue de Angular.js a Angular 2, ahora tendremos *semantic versions* que serán retrocompatibles.

b) El enfoque son los componentes, haciendo que tengas UI retutilizables.

Hola Mundo con Angular CLI

Los CLI (command line interface) son herramientas que nos ayudan a simplificar ciertas tareas que tendríamos que hacer de manera manual con algún framework / librería / software. En el caso de Angular, nos proporciona herramientas que van desde crear un proyecto desde cero hasta un servidor local para pruebas.

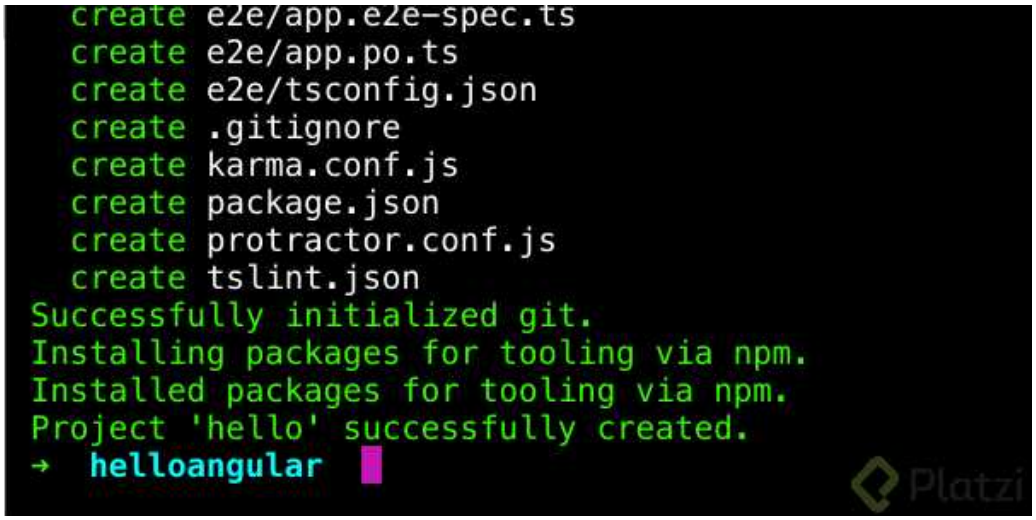
Para instalar Angular CLI debemos tener previamente instalado Node.js y ejecutar el siguiente comando:

```
npm install -g @angular/cli
```

Ya instalado Angular CLI podemos crear proyecto de manera más sencilla sin tener que crear todos los archivos de manera manual. Vamos a crear nuestro Hola Mundo, lo primero que tenemos que hacer es pararnos en la carpeta donde deseamos tener nuestro proyecto y después ejecutar la siguiente linea:

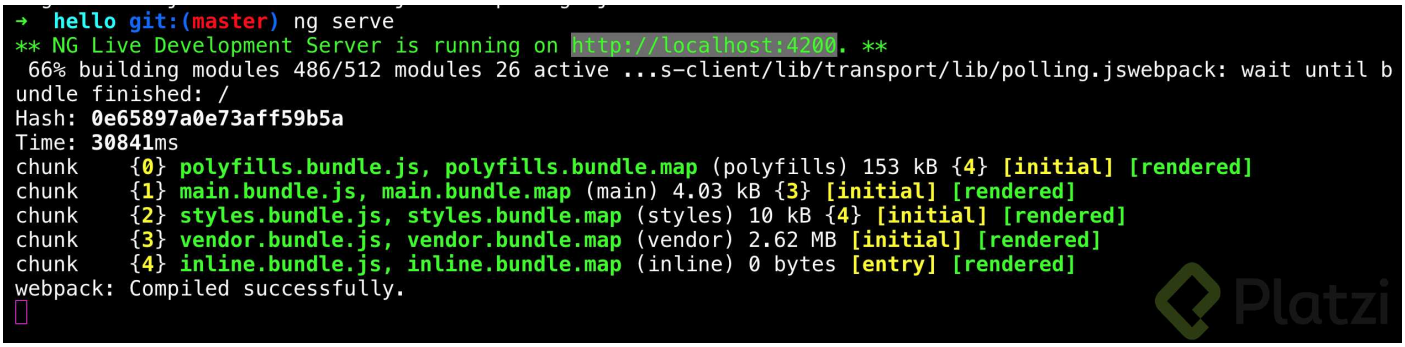
```
ng new NombreDelProyecto
```

La espera dependerá entre muchas cosas de tu conexión a internet, pero no te preocupes, el angular cli nos notificará cuando el proyecto esté listo para usar.

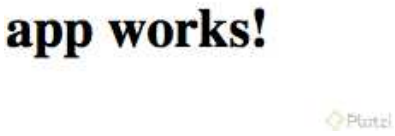


Pero, ¿Cómo ejecutamos un Hola Mundo?, eso es muy sencillo, dentro de angular cli contamos con una herramienta que nos va a ayudar a lenvantar un servidor local con nuestro proyecto. Para hacer uso de la herramientas necesitamos esta dentro de la carpeta de proyecto (/NombreDeProyecto) y ejecutar lo siguiente:

```
ng serve
```

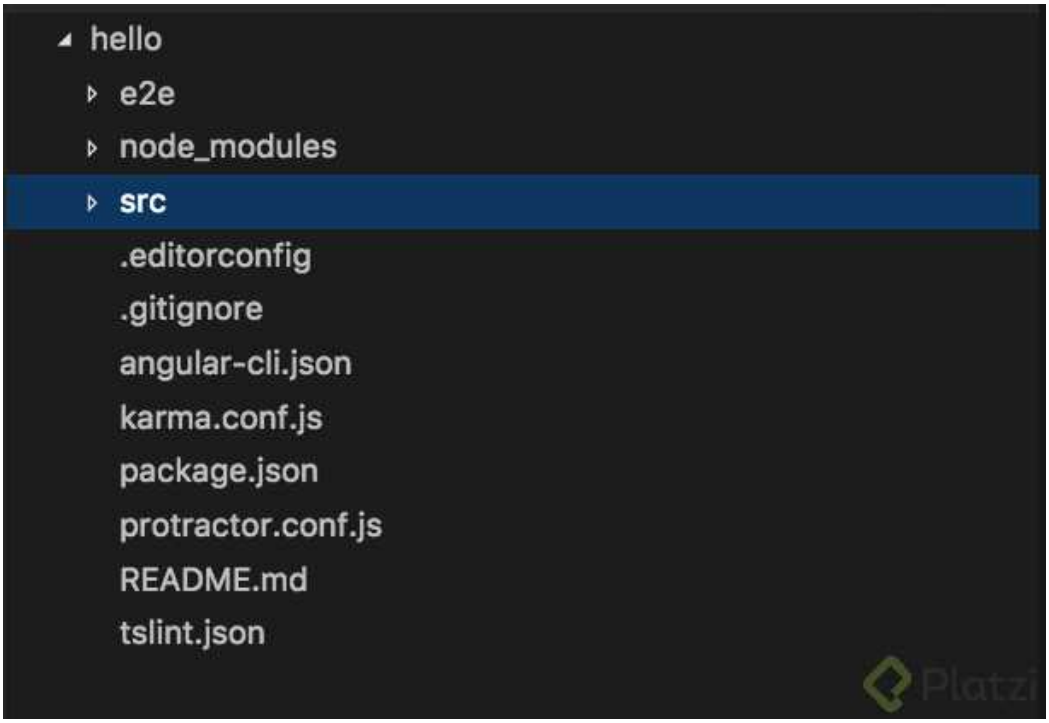


Y listo, al ingresar la dirección local propuesta por el CLI observaremos que nuestro proyecto se creo correctamente y podemos empezar a trabajar con el. **“Hola al Mundo de Angular 2”**



Estructura de una aplicación en Angular

Para iniciar a entender como funciona una aplicación en Angular iniciaremos por entender cuál es la estructura de un proyecto, notarás que desde el CLI se crearon algunos archivos y carpetas que quizás nunca habías visto, repasaremos desde los archivos más externos a los más internos.



Archivos de configuracion:

.editorconfig: Es la configuración para nuestro editor, pero realmente esto no se modifica
.gitignore: Angular CLI nos genera un cliente de git para poder trabajar de manera versionada en nuestro proyecto y después subirlo a Github por ejemplo, existen archivos que no valen la

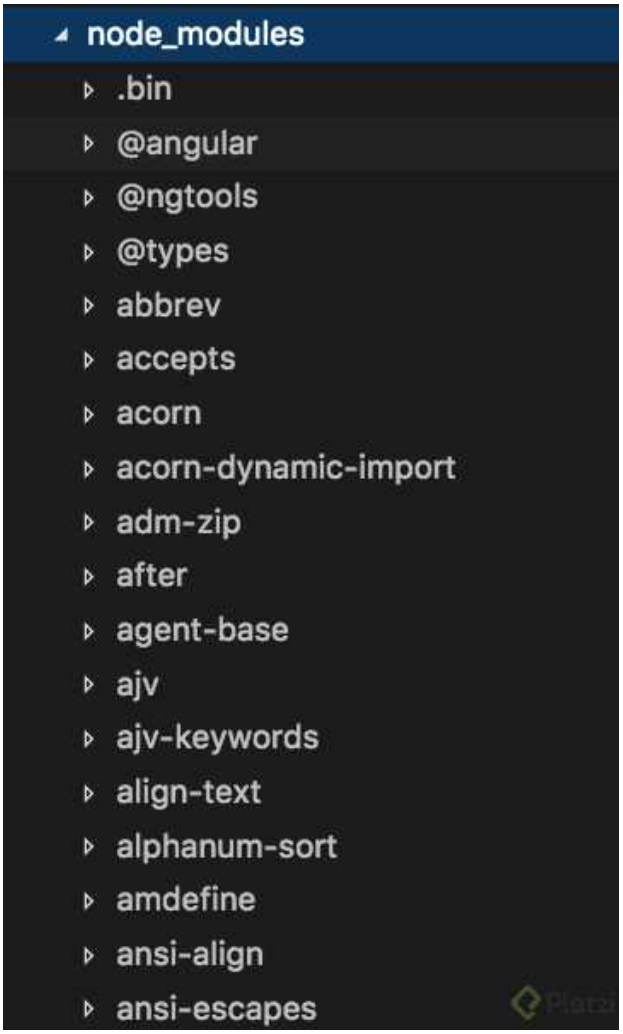
pena tenerlos en cada versión local y es aquí donde ignoramos esos archivos.
angular-cli.json: Aquí es donde consultamos la configuración del CLI.
karma.conf.js: Es el archivo de configuración para los test
package.json: Es la configuración que describe las dependencias que necesitamos de Node.js
protractor.conf.js: Es la configuración de los test e2e con Jasmine
README.md: Es el readme que tiene cualquie proyecto compartido en github, pero en este caso puedes encontrar una guia de los comandos de Angular CLI
tslint.json:Es la configuración del Linter para typescript, realmente no es necesario modificarlo.

/e2e

Los archivos dentro de e2e describen un tipo de testing llamado End to End, esto se ejecuta con la ayuda de Jasmine.

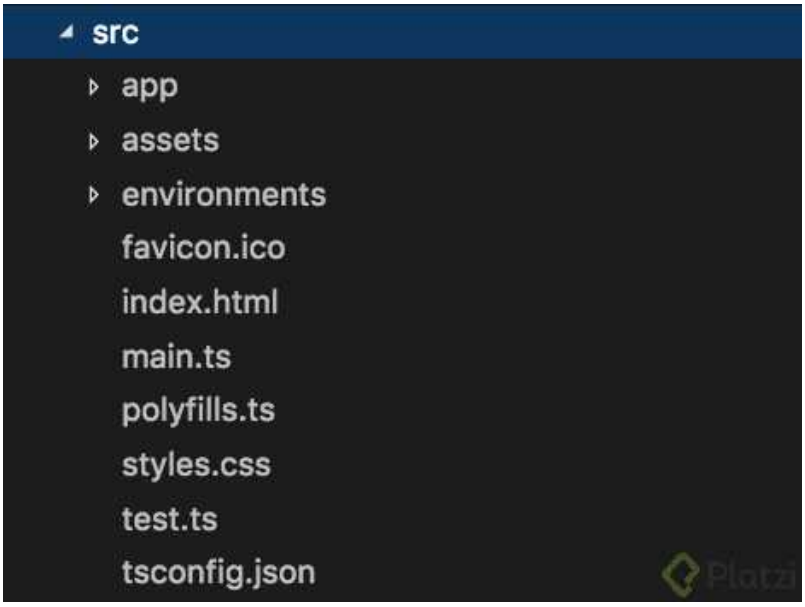
/node_modules

Es donde se encuentran todas nuestras dependencias de Node.js de nuestro proyecto



/src

Está quizás es la carpeta más importante de todas y donde estarás la mayor parte de tu tiempo ya que en ella se encuentra todo el código de la aplicación



Vamos a detalles todo lo que compone la carpeta /src.

/app

En esta carpeta y por carpetas, es donde vamos a tener todos nuestros componentes, servicios y aquellos elementos que tengan que ver con las vista de nuestra aplicación, no te preocupes, más adelante explicaremos todos los elementos que tienen un componente en Angular.

/assets

En esta carpeta tendremos aquellos assets que no sean propios de un componente en Angular como imágenes, videos o archivos de cualquier tipo.

/environments

Nos permite modificar el entorno de trabajo en el que estamos, por lo tanto, cuando lleves algo a producción, modificaremos los respectivos archivos de TypeScript

favicon.icon

Puedes sustituirlo por el icono de tu preferencia para obtener un resultado como:

index.html

Este es el HTML principal y es donde va a existir nuestra Single Web Application

main.ts

Es el primer archivo en ejecutarse, en el se encuentran todos los parametros de configuración la de aplicación como el entorno en el que trabajamos, en que archivo tenemos declarados todos nuestros componentes, etc.

polyfills.ts

Este archivo nos asegura que tendremos compatibilidad en todos los navegadores modernos.

styles.css

Son los estilos generales del proyecto

test.ts

Aquí podemos seguir agregando tests unitarios a nuestros componentes

tsconfing.json

Si recuerdas el artículo de typescript, este es el archivo de configuración propio del superset.

¿Cómo funciona un componente en Angular?

Nos faltó entender que hay dentro de /app, pero esto va muy mano de como esta compuesto un componente en Angular. Un componente en Angular tiene 3 partes:

- Un template o estructura HTML
- Una hoja de estilos CSS propia del componente

- Toda la funcionalidad del componente usando TypeScript (aunque tambien podría usar JS)

Es importante que los nombres de los archivos sean del tipo *nombre.component.extensión*:

Aclarado lo anterior quizás ya te diste cuenta que /app es un componente. Veamos el archivo que describe la funcionalidad *app.component.ts* :

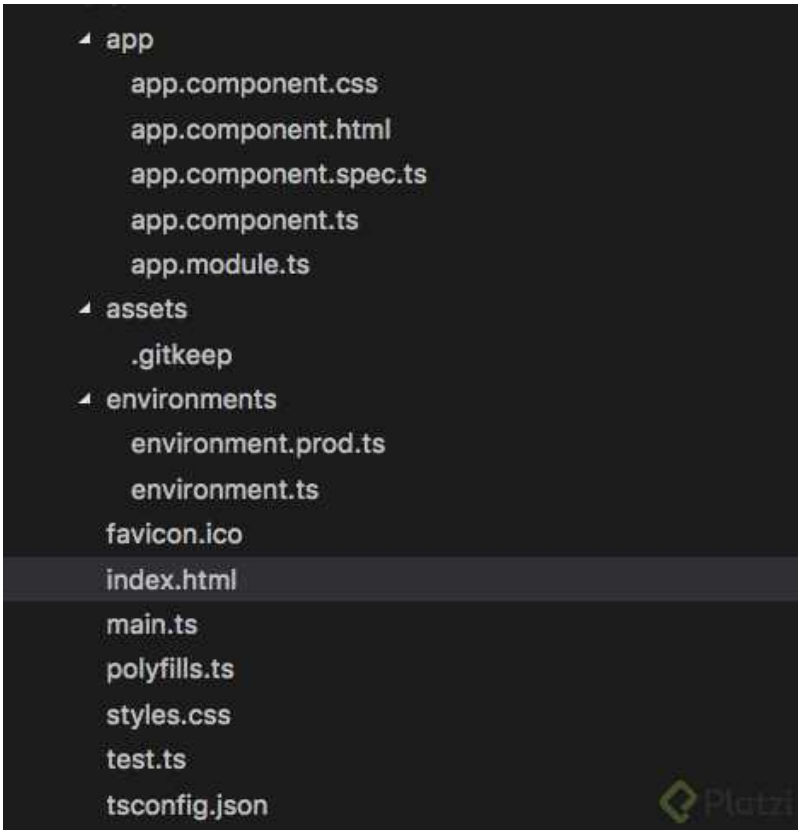
```
import { Component } from '@angular/core';  // Importamos la funcionalidad de compontente
del core de Angular

@Component({             // Este annotation nos ayuda a la configuración del componente,
entre ellos:
  selector: 'app-root',          // En que selector se va a renderizar
  templateUrl: './app.component.html',       // En que ruta esta su template
  styleUrls: ['./app.component.css']            // En que ruta se encuentra su hoja de
estilos
})
export class AppComponent {                  // La clase que vamos a exportar le dará
vida a nuestros componente, podemos tener incluso variables y
  title = 'app works!';                              // funciones queramos usar en
nuestros HTML ya montado
}
```

Nuestro template en este ejemplo no tiene más que:

```
<h1>
  {{title}}
</h1>
```

Y la hoja de estilos no tienen nada por el momento. Bueno, entonces: ¿Dónde termina nuestro componente?, recordemos que tenemos nuestro index.html:




```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Hello</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
    <app-root>Loading...</app-root>   // ¿Recuerdas el selector que declaramos en el componente?, es para que ahí se haga el render
</body>
</html>
```

¿Qué otro archivo nos faltó? Nuestro app.module.ts

Este archivo es el encargado de entender que componentes y dependencias tenemos en nuestra aplicación. Por cierto, cada vez que tu creas un componente nuevo, debes declararlo en este archivo.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Y listo, así aprendiste a hacer tu primer componente en Angular. Te voy a dejar un desafío: Sin usar el Angular CLI, deberás crear un nuevo componte en esta aplicación que se llame Platzi y debe estar debajo de <app-root>Loading...</app-root>

Compartelo en los comentarios

B I U | Insertar código | Enlace | Imagen

Deja tu comentario

Suma tu comentario

Top Nuevas

1  ArmandoVEN 0 Puntos 3 meses

Excelente explicación introductoria

Responder

1  edwintorresmec 0 Puntos 3 meses

Hola, excelente explicación, pero soy novato y me costó descubrir que también tenía que imporatr el módulo Platzi en el main.ts
Logré hacer que ejecute bien con mi icono, mi título y mi nuevo componente 😊

1

ThespianArtist 3 meses

Ese era el truco 😉 que bien que lo logra te 😉

Responder

1

pixelciosa 104 Puntos un día

Logré implementarlo bien, pero no edité el main.ts, solo el app.module.ts que quedó asi:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
import { PlatziComponent } from './platzi.component/platzi.component';


@NgModule({
  declarations: [
    AppComponent,
```

Ver más...

Responder

1

ismain 0 Puntos 3 meses

Buenas, pude crear el componente sin problemas. Al no poder usar Angular CLI tuve que hacer todo a mano, pero bueno básicamente era copiar y pegar lo de app y modificarlo. También había que importar el módulo en el main.ts, eso me di cuenta probando.

Saludos.

0

ThespianArtist 3 meses

lo de importar el componente en main.ts era justo lo que quería que aprendiéramos y creo que se logró, bien 😉

Responder

1

willybardales 2 Puntos 3 meses

El mejor framework Angular CLI !

2

ThespianArtist 3 meses

♥

Responder

0

cccadeveloper 4 Puntos 2 meses

Alguien me podria ayudar diciendo cual es la modificacion del main.ts, gracias

0

ThespianArtist 2 meses

Para agregar un nuevo componente es justo agregarlo en el main.ts. Lo haces por ejemplo:

```
import { Component2 } from '../app/component2/component2';
```

Tambien lo agregas en el declarations:

```
declarations: [ AppComponent, Component2 ]
```

0

cccadeveloper 2 meses

ya tengo bien el declarations, pero en el main.ts es que tengo problemas

mi code esta asi:

```
import { AppPlatzi } from'./app/app.component';
```

el archivo app.module.ts esta asi:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
```

Ver más...

Responder

0

johandymduran 0 Puntos un mes

Excelente explicación uriel!

Responder

0

Jenchi_Medina 42 Puntos 3 meses

Si fue un gran reto. le batalle ya que me marcaba error pero al final de cuenta lo solucione.

Nota: un pequeño espacio puede ser la diferencia de que funcione o no. créame a mi paso, revise el código como 20 veces.

0

ThespianArtist 3 meses

Wow, ¿En donde un espacio afecto a que funcionara?

Responder

0

Elpagano 0 Puntos 4 días

Que hace el archivo Specs, es necesario crearlo ya que el NG component lo crea automáticamente

Responder

0

CristianVega 46 Puntos 3 meses

Estoy siguiendo los pasos para crear el proyecto, cuando ejecuto ng new miComponente, al terminar el proceso en la terminar me aparece que fue creado, pero voy y reviso en donde lo cree y no aparece nada en blanco.Alguien sabe que pasa?

0

ThespianArtist 3 meses

¿Qué te aparece en la terminal?

0

CristianVega 3 meses

Project 'miComponente' successfully created.

CristianVega 3 meses

En ese momento doy cd miComponente pero no existe esa carpeta

0

Ver todas las respuestas

Responder

0

osvas 5 Puntos — 2 meses

Tried to find bootstrap code, but could not. Specify either statically analyzable bootstrap code or pass in an entryModule to the plugins options.

esto pasa cuando doy ng serve.

aiuda :v

Responder

0

_dvdgonzalez 0 Puntos — 2 meses

Estuvo difícil el reto, pero con algo de documentación se logró 😄

Responder

0

jancarlo 11 Puntos — 3 meses

¿A qué se refiere al decir “sin usar Angular CLI”? 😯

ya que la idea sería agregar una nueva carpeta en src y dentro de ella los archivos del componente (html, css, etc) y en el html hacer referencia al selector que coloques, pero no sé si eso sea estar usando CLI, o si CLI es cuando corres el server 😯

0

ThespianArtist — 3 meses

Se puede crear un componente desde la teminal usando:

ng -g component “nombredelcomponente”

Entonces es correcto o que dices, crear html, css y .ts.

Si puedes usar cli para el server 😄

1

jancarlo — 3 meses

Ah! vale, desconocía ese comando, oks, gracias por la aclaración! Genial el curso por cierto (y)

Responder