Guía del Principiante para Angular 4: Componentes

Original de: Roy Agasthyan
Traducción al español: Rafael Chavarría

Angular es un marco de trabajo popular para crear front ends para aplicaciones web y móviles. Todo comenzó con AngularJS 1.x y después AngularJS 2, y ahora finalmente Angular, con las últimas actualizaciones y reparación de errores siendo trabajadas por el equipo de Angular.

Los componentes son una parte importante de una aplicación web Angular. En este tutorial, verás cómo comenzar con la creación de una aplicación web usando Angular, y también conocerás componentes.

Comenzando

Comienza instalando Angular CLI usando el administrador de paquetes node (npm).

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

Una vez que tienes el CLI Angular instalado, crea una nueva aplicación Angular usando CLI.

```
ng new angular-app
```

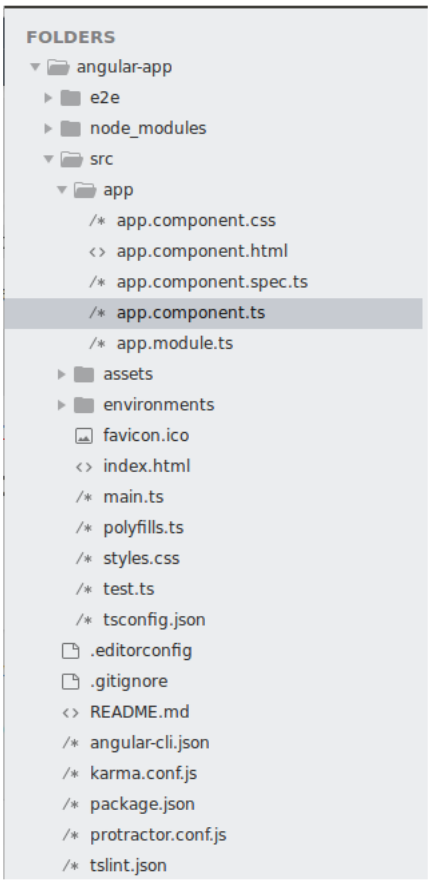
Navega a la carpeta de aplicación e inicia el servidor.

```
cd angular-app
ng serve
```

Apunta a tu navegador a http://localhost:4200/ y deberías tener la aplicación por defecto ejecutándose.

Estructura de Aplicación Angular

Navega a la carpeta de aplicación `angular-app` y echa un vistazo a la estructura del proyecto. Aquí está cómo luce:



Toda aplicación de angular tiene un módulo raíz en donde defines el componente principal a cargar. En la aplicación Angular por defecto, el módulo raíz es definido dentro de `app.module.ts`. Cuando carga el `AppModule`, este revisa cuál componente está incluido y carga ese módulo. Como se vio en el `app.module.ts`, el módulo que está incluido es `AppComponent`. El componente `AppComponent` es definido en el archivo `app.component.ts`.

Un componente es definido usando el decorador `@Component`. Dentro del decorador `@Component`, puedes definir el componente `selector`, el componente `template`, y el `style` relacionado.

¿Qué Es un Componente Angular?

Los componentes son como el bloque básico de construcción en una aplicación Angular. Los componentes son definidos usando el decorador `@component`. Un componente tiene un `selector`, `template`, `style` y otras propiedades,

usando el que especifican los metadatos requeridos para procesar el componente.

Desde la documentación oficial:

"Los componentes son el bloque de construcción más básico de un UI en una aplicación Angular. Una aplicación Angular es un árbol de componentes Angular. Los componentes Angular son un sub-conjunto de directivas. A diferencia de las directivas, los componentes siempre tienen una plantilla y solo un componente puede ser instanciado por un elemento en una plantilla".

La mejor manera de entender alfo relacionado a programar es hacerlo de verdad. Así que comencemos a crear un componente Angular para sumar dos números. Llamémoslo `CalculatorComponent`.

Creando el Componente Calculator

Comencemos creando un componente para nuestra calculadora. Dentro de la carpeta `src/app`, crea una carpeta llamada `calc`. Aquí es donde nuestro componente `calc` residirá. Dentro de la carpeta `calc`, crea un archivo llamado `calc.component.html`. Esta será la plantilla para nuestro componente de calculadora. Aquí está cómo luce:

```
1  <h1>
2      Calculator component
3  </h1>
```

Crea un archivo llamado `calc.component.ts`. Aquí es donde definirás el componente `calc` y especificarás los metadatos relacionados. Estarás definiendo el componente usando el decorador `@component`. Para definir el componente, necesitas importar el módulo `component` desde el núcleo de angular.

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

Define el componente especificando el `template`, `style`, y `selector`. También definirás una `class` para administrar la plantilla especificada por el decorador `@component`. Aquí está cómo luce:

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

@Component({
  selector: 'calc',
  templateUrl: 'calc.component.html',
  styleUrls: ['calc.component.css']
})

export class CalcComponent {
}
```

Todos los estilos relacionados al componente de la plantilla debería ser definido dentro del archivo especificado en el decorador de componente. Así que crea un archivo llamado `calc.component.css` dentro de la carpeta `calc`. Pondrás el sistema para el componente de la calculadora dentro de este archivo.

Ahora que tienes tu componente listo, definamos el componente dentro del módulo raíz `app.module.ts`.

Primero importa el componente dentro de `app.module.ts file` y después inclúyelo en la sección `declarations`. Aquí está cómo luce después de agregar el `CalcComponent`:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { CalcComponent } from './calc/calc.component'

@NgModule({
  declarations: [
    AppComponent,
    CalcComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})

export class AppModule { }
```

Como puedes ver en el módulo root `app.module.ts`, el `AppComponent` es el módulo incluido y por lo tanto de generará por defecto. Así qué, para ver nuestro componente calculadora, defínelo dentro de `app.component.html`. Aquí está cómo luce el archivo `app.component.html`.

```
<div style="text-align:center">
  <calc></calc>
</div>
```

Guarda los cambios de arriba y comienza el servidor. Deberías poder ver el contenido HTML del componente calculadora mostrado.

Agregando la Funcionalidad Calculadora

Comencemos agregando una plantilla para nuestra calculadora Angular. Agrega el siguiente código al archivo `calc.component.html`:

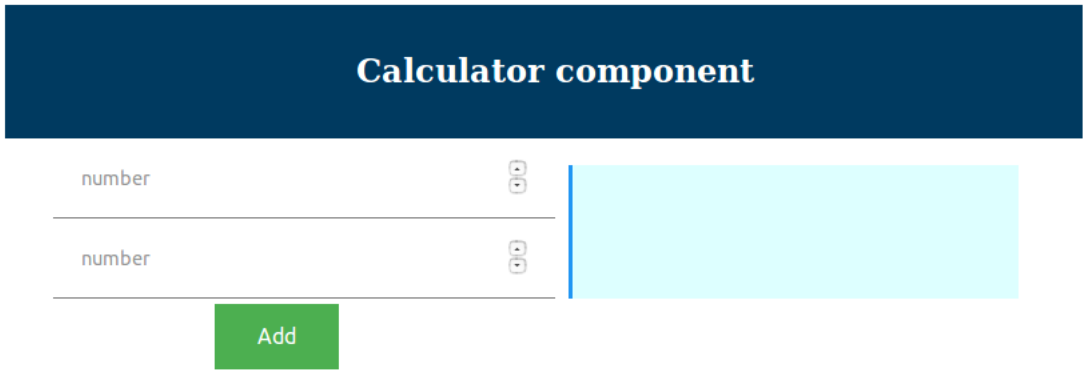
```
<div class="container">
    <div class="header">
      <h2>
            Calculator component
        </h2>
    </div>

    <div class="grid">
        <div class="row">
            <div class="col-6">
                <div class="operation">
                    <div class="row">
                        <div class="col-12">
                            <input type="number" name="" placeholder="number">
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-12">
                            <input type="number" name="" placeholder="number">
                        </div>
                    </div>
                    <div>
                        <div class="col-12">
                            <button class="button">
                                Add
                            </button>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-6">
                <div class="result">
                    <span>
                        Result
                    </span>
                </div>
            </div>
        </div>
    </div>
</div>
```

Agrega el siguiente estilo al archivo `calc.component.css`.

```
.grid{
    width: 100%
}
.row{
    width: 100%;
    display: flex;
}
.col-1 {
    width: 8.33%;
}
.col-2 {
    width: 16.66%;
}
.col-3 {
    width: 25%;
}
.col-4 {
    width: 33.33%;
}
.col-5 {
    width: 41.66%;
}
.col-6 {
    width: 50%;
}
.col-7 {
    width: 58.33%;
}
.col-8 {
    width: 66.66%;
}
.col-9 {
    width: 75%;
}
.col-10 {
    width: 83.33%;
}
.col-11 {
    width: 91.66%;
}
.col-12 {
    width: 100%;
}
.header{
    width: 100%;
    background-color: #003A60;
    height: 100px;
}
.header h2{
    line-height: 100px;
    color: #fff;
}
.button {
    background-color: #4CAF50; /* Green */
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
```

```
    margin: 4px 2px;
    cursor: pointer;
}
input{
    border: none;
    border-bottom: 1px solid grey;
    width: 80%;
    margin: 0% 10%;
    padding: 5%;
}
.result{
    background-color: #ddffff;
    width: 80%;
    margin: 20px 10px 10px 10px;
    height: 100px;
    border-left: 3px solid #2196F3;
}
.result span{
    line-height: 100px;
}
```

Guarda los cambios de arriba y deberías poder visualizar la siguiente interfaz de usuario.



Agreguemos la directiva `ngModel` a las cajas de texto de entrada mostradas arriba. Modifica el código de `calc.component.html` como se muestra abajo:

```
<div class="row">
    <div class="col-12">
        <input [(ngModel)]="number1" type="number" name="" placeholder="number">
    </div>
</div>
<div class="row">
    <div class="col-12">
        <input [(ngModel)]="number2" type="number" name="" placeholder="number">
    </div>
</div>
```

Como se vio arriba, has establecido el `ngModel` para las cajas de entrada de texto a las variables `number1` y `number2`.

Definamos las variables dentro del `CalcComponent` en el archivo `calc.component.ts`.

```
export class CalcComponent {
  public number1 : number;
  public number2 : number;
}
```

Ahora, cuando el usuario teclea en las cajas de texto, la variable `ngModel` correspondiente se actualiza. Puedes revisar mostrando la variable en el archivo de plantilla del componente.

```
    <div class="result">
        <span>
            Number 1 : {{number1}}
            Number 2 : {{number2}}
        </span>
    </div>
```

Guarda los cambios e ingresa valores dentro de las cajas de texto, y deberías tener los datos actualizados dentro del span.

Agreguemos un clic de botón al botón `Add` el cuál calculará la suma del `number1` y `number2` cuando se da clic en el botón.

Modifica el código HTML como se muestra para incluir la directiva clic.

```
    <button (click)="add()" class="button">
```

```
        Add
    </button>
```

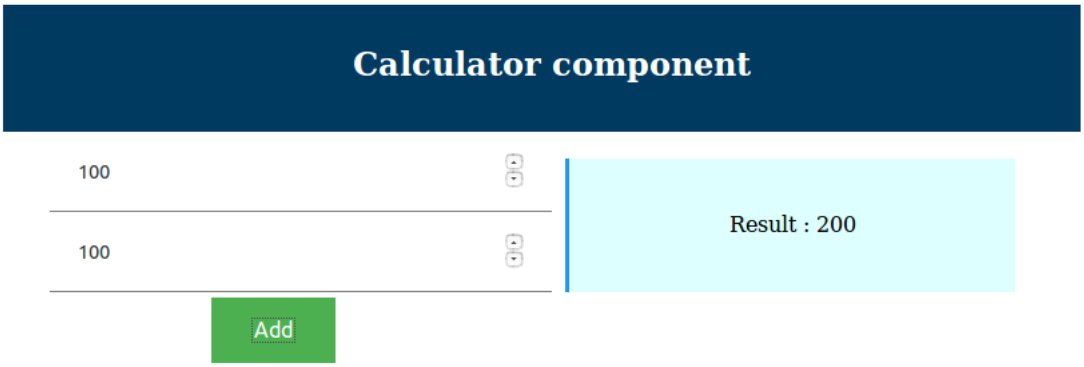
Define la función `add` dentro del `CalcComponent` como se muestra:

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

@Component({
  selector: 'calc',
  templateUrl: 'calc.component.html',
  styleUrls: ['calc.component.css']
})

export class CalcComponent {
  public number1 : number;
  public number2 : number;
  public result : number;

  public add(){
      this.result = this.number1 + this.number2
  }
}
```

Como se vio en el código de arriba, el resultado de la suma está siendo colocado en una variable llamada result
Modifiquemos la plantilla HTML para mostrar el resultado una vez que la variable es establecida.

```
<div class="result">
    <span>
        Result : {{result}}
    </span>
</div>
```

Guarda los cambios de arriba e intenta sumar dos números dando clic sobre el botón **Add**. El resultado se mostrará en la interfaz de usuario.



Terminando

En este tutorial, viste cómo comenzar con crear una aplicación web usando Angular 4. Aprendiste sobre componentes Angular y cómo crear uno. Creaste un simple componente Angular simple para sumar dos números.

El código fuente de este tutorial está disponible en GitHub (https://github.com/royagasthyan/AngularComponent).