Curso de Angular 6

Unidad Didáctica 03: Componentes

http://cursosdedesarrollo.com/

Ayuntamiento
de Vitoria-Gasteiz
Vitoria-Gasteizko
Udala

Índice de contenidos

- Introducción
- Creación de un componente
- Asociando el componente
- Modificando un componente
- Referencias
- Conclusiones

http://cursosdedesarrollo.com/

Introducción

Dentro de esta unidad veremos cómo crear nuevos componentes que podremos utilizar dentro de la aplicación

http://cursosdedesarrollo.com/

Creación de un Componente

Para crear un componente utilizaremos el comando

ng generate component nombre_componente

Esto modificará la configuración de la aplicación y generará los ficheros necesarios para el desarrollo del nuevo componente

Creación de un Componente

Por ejemplo ejecutaremos el comando:

ng generate component todos

Creación de un Componente

$ ng generate component todos

installing component

create src/app/todos/todos.component.css

create src/app/todos/todos.component.html

create src/app/todos/todos.component.spec.ts

create src/app/todos/todos.component.ts

update src/app/app.module.ts

http://cursosdedesarrollo.com/

Creación de un Componente

Como podemos ver nos ha creado un carpeta
src/app/todos

donde ha metido todos los ficheros del componente
y ha modificado el fichero app.module.ts para incluir
el componente en la cadena de compilación del
proyecto

http://cursosdedesarrollo.com/

Creación de un Componente

Dentro de la carpeta todos nos encontramos con el fichero todos.compontent.ts donde configuramos las importaciones el componente y la clase del componente

http://cursosdedesarrollo.com/

Creación de un Componente

todos.component.ts

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

@Component({
  selector: 'app-todos',
  templateUrl: './todos.component.html',
  styleUrls: ['./todos.component.css']
})
export class TodosComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

http://cursosdedesarrollo.com/

Asociando el Componente

Una vez creado el componente se asocia a la aplicación mediante el fichero app.module.ts

```
import { TodosComponent } from './todos/todos.component';
```

Asociando el Componente

También se incluye la declaración en el componente principal

```
@NgModule({
  declarations: [
    AppComponent,
    TodosComponent
  ],
})
```

http://cursosdedesarrollo.com/

Asociando el Componente

Luego deberemos modificar la plantilla del componente principal para que se modifique la vista

```
<h1>
  {{title}}
  <app-todos></app-todos>
</h1>
```

http://cursosdedesarrollo.com/

Modificando un Componente

El método ngOnInit es el que nos permite la modificación de datos en la carga de angular del componente

Así podremos modificar datos declarados en el componente

http://cursosdedesarrollo.com/

Modificando un Componente

```
todos;

ngOnInit() {

    this.todos=[
        {text:"hacer la compra”},

        {text: "revisar el coche"}

    ];

}
```

http://cursosdedesarrollo.com/

Modificando un Componente

Luego esos datos los podemos utilizar desde la plantilla:

```
<h3>Listado de tareas</h3>

  <ul>

    <li *ngFor="let todo of todos">

      {{todo.text}}

    </li>

  </ul>
```

http://cursosdedesarrollo.com/

Modificando un Componente

También podemos manejar acciones, por ejemplo el submit de un formulario:

```
<h4>Añadir tarea</h4>
    <form (submit)="addTodo()" >
      <input type="text" [(ngModel)]="todo.text"
          [ngModelOptions]="{standalone: true}"/>
    </form>
```

http://cursosdedesarrollo.com/

Modificando un Componente

Los eventos se encierran con ()

Las propiedades con []

Las expresiones con {{}}

Los data-binding son combinaciones de eventos y propiedades [()]

http://cursosdedesarrollo.com/

Modificando un Componente

Para que funcionen bien los databindings ngModel será necesario hacer unas inclusiones de código en el app.module.ts:

```
import { FormsModule } from '@angular/forms';

    imports: [

      BrowserModule,

      FormsModule

    ],
```

http://cursosdedesarrollo.com/

Modificando un Componente

Los eventos pueden actualizar modelo

Las propiedades pueden actualizar vista

Las expresiones se definen en la vista

Los data-binding pueden comunicar modelo y vista bidireccionalmente

http://cursosdedesarrollo.com/

Modificando un Componente

Deberemos añadir el método al componente:

```
todo;

addTodo(){
    //console.log(this);
    let mitodo={text:this.todo.text};
    this.todos.push(mitodo);
  }
```

http://cursosdedesarrollo.com/

Modificando un Componente

También podemos gestionar el click de un elemento desde la plantilla:

```
<button (click)="deleteTodo(todo.text)" >Borrar</
button>
```


Modificando un Componente

Y añadiendo el método al componente:

```
deleteTodo(texto){

    for(var i=0;i<this.todos.length;i++){

        var todo=this.todos[i];

        if(todo.text==texto){

          this.todos.splice(i,1);

        }

      }

    }
```

http://cursosdedesarrollo.com/

Referencias

Angular 2 Primeros Pasos: https://www.youtube.com/watch?v=Y7izsxhPpQY

Simple Angular 2 App with Angular CLI: https://www.youtube.com/watch?v=QMQbAoTLJX8

Conclusiones

Hemos visto cómo se crean
los distintos componentes
de una aplicación Angular

http://cursosdedesarrollo.com/

Datos de Contacto

http://www.cursosdedesarrollo.com
david@cursosdedesarrollo.com

http://cursosdedesarrollo.com/

Licencia

David Vaquero Santiago

Esta obra está bajo una
Licencia Creative Commons
Atribución-NoComercial-
CompartirIgual 4.0 Internacional

http://cursosdedesarrollo.com/