UNIT 2

Angular

Exercise week 8

Client-side Web Development
2nd course – DAW
IES San Vicente 2020/2021
Author: Arturo Bernal Mayordomo

Index

Introduction....3
Listing the products....3
Transform the image to Base64....4
Submitting the form....4
Recommendations and tips....5

Introduction

We are going to implement, more or less, the exercise we did in week 2, with Angular. A single page with a list of products and a form to add a new one.

Create a project called **angular-sanvipop**. Use **ng new --strict -p sp angular-sanvipop** → It will use the prefix **sp** (SanviPop), instead of **app** for the components (more info https://github.com/angular/angular-cli/wiki/new).

```
arturo@arturo-desktop:~/Documentos/2020-2021$ ng new --strict -p sp angular-sanvipop
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
```

In this project create a component called **products-page**. This will have almost all the HTML we need for this exercise. The **nav** bar and the **div.container** will go in the AppComponent (and the **products-page** selector inside the container). Create an interface called Product that will have the needed properties:

```
export interface Product {
   id?: number;
   title: string;
   description: string;
   price: number;
   mainPhoto: string;
   category: number;
}
```

Don't forget to install **Bootstrap** and **FontAwesome** (we'll need this one in the future for icons) and include the CSS in the **angular.json** file!

npm i bootstrap @fortawesome/fontawesome-free

```
"styles": [
   "node_modules/bootstrap/dist/css/bootstrap.css",
   "node_modules/@fortawesome/fontawesome-free/css/all.css",
   "src/styles.scss"
]
```

In the component, create an empty array of Products called **products**, and a single Product object called **newProduct** (with empty fields).

In the HTML, reference each property of the **newProduct** object with the **[(ngModel)]** directive. Example:

```
<input type="text" class="form-control" name="title" [(ngModel)]="newProduct.title"  placeholder="Enter title">
```

Listing the products

By default in the product description, new line breaks won't have any effect (it will be shown in one line). You can address this in two ways (so it understands \n instead of
).

CSS approach:

```
.card-text {
  white-space: pre-wrap;
}
```

HTML approach. Use **innerText** attribute instead of normal interpolation:

```
<p class="card-text">{{product.description}}</p>

<!-- Use this instead → Translates \n to <br>-->
<p class="card-text" [innerText]="product.description"></p>
```

Generate the cards using **ngFor**.

Format the price using the currency pipe.

As the category is a number, create an array with the categories in the component and use category number – 1 to get the name of that category.

Don't forget to include the CSS provided in the week 2 exercise in the component (except the #errorMsg CSS, you can also delete that HTML element).

Transform the image to Base64

The concept is the same as in plain JavaScript, but the events now are handled differently. The HTML for the file input and the img element will look like this:

```
<div class="form-group">
  <label for="image">Image</label>
  <input type="file" class="form-control" #fileImage (change)="changeImage(fileImage)">
</div>
<img [src]="newProduct.mainPhoto" alt="" class="img-thumbnail">
```

Don't worry if you don't understand what #fileImage means (it's a reference to the HTML element). We'll see how to use that in the future.

This would be the component's method that converts the image to Base64.

```
changeImage(fileInput: HTMLInputElement) {
  if (!fileInput.files || fileInput.files.length === 0) { return; }
  const reader: FileReader = new FileReader();
  reader.readAsDataURL(fileInput.files[0]);
  reader.addEventListener('loadend', e => {
    this.newProduct.mainPhoto = reader.result as string;
  });
}
```

Submitting the form

To capture the submit event properly use this event in your HTML page:

```
<form class="mt-4" (ngSubmit)="addProduct()">
```

When submitting the form all you have to do is add the newProduct object to the array of products. Also, after adding the product, assign a new empty product object to

the newProduct property (reset the form).

Another way is to create a copy of the newProduct object {...this.newProduct}, and insert the copy in the array. After that restore the values of the newProduct object to empty.

Remove the **action** and **id** attributes from the form.

Recommendations and tips

- You don't need to validate the form (that's something we'll learn in the future to do with Angular). You can omit any error messages.
- You'll need to include Angular's **FormsModule** in your App Module or directives like **[(ngModel)]** or events like **ngSubmit** won't work!
- When using **[(ngModel)]** inside a <form> element, the element **must have a name property** (we'll learn more about forms in the future).
- If you want to be able to reset the <input type="file"> value when adding a product, create a string property in the component, bind it to the input using ngModel, and set it to empty string when you reset the form.

SanviPop

Title

Product you are selling

Description

Write something...

Price

0

Category

Choose a category...

Main photo

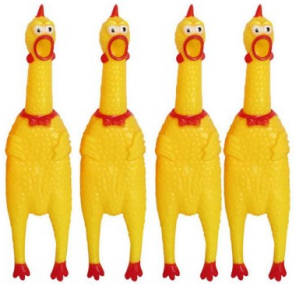
Seleccionar archivo Ningún archivo seleccionado

Create

Dinosaur custome

For programming in Cobol

Other 120 €



Rubber chicken

What is this for????

Sports and hobbies 19 €