INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

Web Programming

2rd Activity - Intro to Angular

2020/2021

1. Outputting posts

The goal is to be able to output the posts, so perform the following steps:

a) Create a **new component** inside posts folder and name it **post-list** and in there, add post-list.component.ts and the post-list.component.html

Use angular cli to create the component:

```
ng g c posts/post-list --spec=false
```

b) Verify if **app.module.ts** dependencies are all set

c) Add post-list selector in the app.component.html page to have the post list appear just after the post create

d) run with **ng serve**

e) Delete the <p> tag in the post-create.component.html

f) Use the expansion panel which is a collapsible, from angular material and import all the files needed (MatExpansionModule) in order to have something like this:

`post-list.component.ts` `post-list.component.html` `post-list.component.css`

```
import { Component, Input } from
"@angular/core";

@Component({
  selector: "app-post-list",
  templateUrl: "./post-list.component.html",
  styleUrls: ["/post-list.component.css"]
})
export class PostListComponent {}
```

```
<mat-accordion>
    <mat-expansion-panel>
        <mat-expansion-panel-header>
      The expansion title here!
        </mat-expansion-panel-header>
        <p>Extension panel</p>
    </mat-expansion-panel>
</mat-accordion>
```

```
:host{
    display: block;
    margin-top:1rem;
}
```

f) Add the module needed in app.module.ts

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { MatCardModule } from '@angular/material/card';
import { MatInputModule } from '@angular/material/input';
import { MatButtonModule } from '@angular/material/button';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatExpansionModule } from '@angular/material/expansion';

import { AppComponent } from './app.component';
import { PostCreateComponent } from './posts/post-create/post-create.component';
import { NoopAnimationsModule } from '@angular/platform-browser/animations';
import { HeaderComponent } from './header/header.component';
import { PostListComponent } from './posts/post-list/post-list.component';
// import { PostCreateBootstrapComponent } from
'./posts/post-create-bootstrap/post-create-bootstrap.component';

@NgModule({
          declarations: [ AppComponent, PostCreateComponent, HeaderComponent,
PostListComponent ],
          imports: [
                    BrowserModule,
                    FormsModule,
                    NoopAnimationsModule,
                    MatCardModule,
                    MatInputModule,
                    MatButtonModule,
                    MatToolbarModule,
                    MatExpansionModule
          ],
          providers: [],
          bootstrap: [ AppComponent ]
})
export class AppModule {}
```

g) Do a stylish step

post-create.component.css

```
mat-form-field,textarea{
    width: 100%;
}
```

app.component.html

```
<app-header></app-header>
<main>

<app-post-create></app-post-create>
    <app-post-list></app-post-list>
</main>
```

app.component.css

```
main {
  width: 80%;
  margin: 1rem auto;
}
```

2. Structural directives

a) *ngFor directive

We want to create an array of objects to simulate posts content and add a way to go through all the elements of the array. We do this adding the content in the ts file, and angular has an instruction that change the HTML DOM, that allow us to repeat an element as often as required - *ngFor . Here we have a new array of posts with 3 objects.

post-list.component.ts

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

@Component({
  selector: 'app-post-list',
  templateUrl: './post-list.component.html',
  styleUrls: [ '/post-list.component.css' ]
})
export class PostListComponent {

posts = [
  { title: 'First Post', content: 'This is the First post' },
  { title: 'Second Post', content: 'This is the Second post' },
  { title: 'Third Post', content: 'This is the Third post' }
```

```
];

}
```

post-list.component.html

```
<mat-accordion multi="true">
    <mat-expansion-panel *ngFor="let post of posts">
        <mat-expansion-panel-header>
       {{post.title}}
        </mat-expansion-panel-header>
        <p>{{post.content}}</p>
    </mat-expansion-panel>
</mat-accordion>
```

The:

```
let post of posts
```

is a javascript way to go through arrays. We are creating a variable post that we can access later in the HTML. In each iteration, a post will be an object that has two properties: title and content. We can access to these values through string interpolation

```
{{post.title}}
```

The multi=true is to allow to open all accordions,

a)*ngIf directive

And if the posts array is empty, how can we handle that case?

Well, Angular has another directive that allows us to test conditions the - *ngIf. In this case we test if we have any post, and we only show that data if the condition is true. So, if we don't we make an alternative HTML. In this case, our alternate HTML is

```
<p class="info-text mat-body-1" *ngIf="posts.length <= 0”>There are no posts, yet!</p>
```

post-list.component.ts

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

@Component({
  selector: 'app-post-list',
  templateUrl: './post-list.component.html',
  styleUrls: [ '/post-list.component.css' ]
})
export class PostListComponent {

posts = [];

}
```

post-list.component.html

```
<mat-accordion multi="true" *ngIf="posts.length > 0">
    <mat-expansion-panel *ngFor="let post of posts">
        <mat-expansion-panel-header>
       {{post.title}}
        </mat-expansion-panel-header>
        <p>{{post.content}}</p>
    </mat-expansion-panel>
</mat-accordion>
<p *ngIf="posts.length <= 0”>There are no posts, yet!</p>
```

And now, adding some style to the paragraph

post-list.component.ts

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

@Component({
  selector: 'app-post-list',
  templateUrl: './post-list.component.html',
  styleUrls: [ '/post-list.component.css' ]
})
export class PostListComponent {

posts = [];

}
```

post-list.component.html

```
<mat-accordion multi="true"
*ngIf="posts.length > 0">
    <mat-expansion-panel *ngFor="let post of
posts">
        <mat-expansion-panel-header>
       {{post.title}}
        </mat-expansion-panel-header>
        <p>{{post.content}}</p>
    </mat-expansion-panel>
</mat-accordion>
<p class="info-text mat-body-1"
*ngIf="posts.length <= 0”>There are no posts,
yet!</p>
```

post-list.component.css

```
:host{
    display: block;
    margin-top:1rem;
}


.info-text{
    text-align: center;
}
```

3. Creating Posts with Property and Event Binding

Now we want to send our post created to the post-list, in order to see all the posts listed. For this we have to code the following activities:

a) save the user input and create a method to send it to another component.

b) gather the user input, save it in an array with all the posts and create a method to send it to other component

c) receive the data from an external component. For this, we'll make some changes, to create a post title and post content. We create another input field for the posts Title and delete the {{newPost}} and the <p> element. We also made some styling changes, just following the rules.

@Output decorator

Save the user input and create a method to send it to another component.

post-create.component.ts

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

@Component({
selector: 'app-post-create',
templateUrl: './post-create.component.html',
styleUrls: [ './post-create.component.css' ]
})

export class PostCreateComponent {

enteredTitle = '';
enteredContent = '';

@Output() postCreated = new EventEmitter();

onAddPost() {
    const post={
              title: this.enteredTitle,
              content: this.enteredContent
              };
    this.postCreated.emit(post);
}
}
```

post-create.component.html

```
<mat-card>
  <mat-form-field>
    <input
    matInput
    type="text"
    [(ngModel)]="enteredTitle">

  </mat-form-field>
    <mat-form-field>
      <textarea
      matInput
      rows=6
      [(ngModel)]="enteredContent">
      </textarea>
    </mat-form-field>

<button mat-raised-button
    color="accent"
    (click)="onAddPost()">Save Post</button>
    <hr>
</mat-card>
```

The method **onAddPost()** will create an object with the user input data, and will send it via a special kind of variable, an **event Emitter** with a special attribute **@Output()**, which is called **decorator**.

```
@Output()  postCreated = new EventEmitter();
```

This decorator allows a data emitter to send its data to his father (the component that calls it) - app.component.ts

Linking data between components

Gather the user input, save it in an array with all the posts and create a method to send it to another component. The component that will receive the data sent is the father component. So in app.component will receive the post created and will stored the data in an array. The way we do this is by running

```
(postCreated)="onPostAdded($event)"
```

where postCreated is the event from the post-create component that is sending data, to our new function onPostAdded in app.component. The way we receive our data is accessing the object of the event, which is **$event**.

We then save the post in an array that will be available from other components. In this case, post-list, but first let's put it ready for receiving data from other components. Go to the **@Input decorator** section..

app.component.ts

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

@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: [ './app.component.css' ]
})
export class AppComponent {

storedPosts = [];

onPostAdded(post) {
 this.storedPosts.push(post);
}
}
```

app.component.html

```
<app-header></app-header>
<main>
<app-post-create
(postCreated)="onPostAdded($event)"></app
-post-create>
<app-post-list
[posts]="storedPosts"></app-post-list>
</main>
```

app.component.css

```
main {
          width: 80%;
          margin: 1rem auto;
}
```

@Input decorator

In our template we need the **post.title** and **post.conten**t of all posts. For that we need to receive the **posts** array and that's why we create an array named **posts**. But it's a special array that we'll be fed by external data, and to inform angular of this feature we use **@Input()**.

Having this array loaded, we now have all the data we need to traverse the array and list all the posts with our ***ngFor**.

post-list.component.ts

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

@Component({
selector: 'app-post-list',
templateUrl: './post-list.component.html',
styleUrls: [ './post-list.component.css' ]
})
export class PostListComponent {
@Input() posts = [];
}
```

post-list.component.html

```
<mat-accordion multi="true"
*ngIf="posts.length > 0">
    <mat-expansion-panel *ngFor="let
post of posts">
    <mat-expansion-panel-header>
        {{post.title}}
    </mat-expansion-panel-header>
    <p>{{post.content}}</p>
    </mat-expansion-panel>
</mat-accordion>
<p class="info-text mat-body-1"
*ngIf="posts.length <= 0">No posts
added yet!</p>
```

post-list.component.css

```
:host {
          display: block;
          margin-top: 1rem;
}

.info-text {
          text-align: center;
}
```

3. Exercise

Create another input field named **authorName**, and show it in the list of posts.