Framework Design project

Tripa Adrian-Daniel

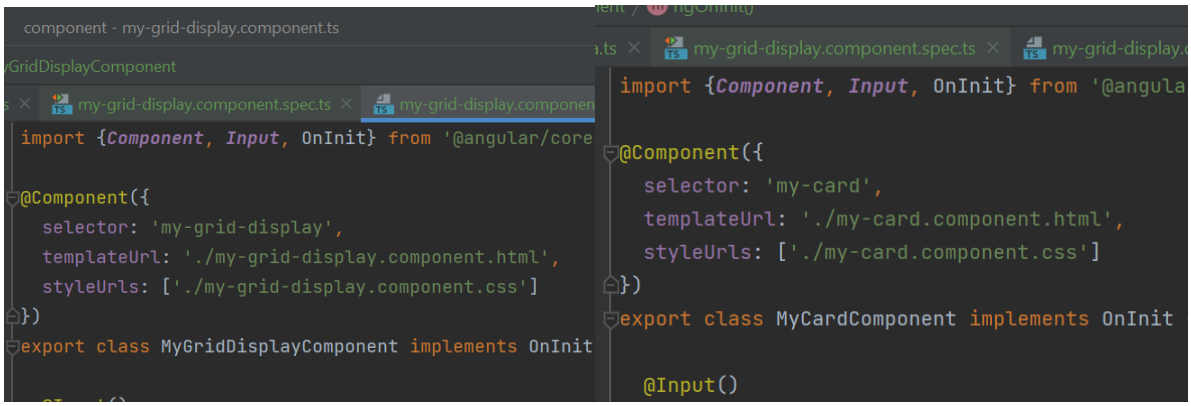
248/2

1. Project: Angular Custom Element – Web Component

Angular custom elements are a way to define HTML tags in a framework-agnostic way. The custom elements are used to encapsulate components that can be reused throughout an application or page. In this way, the code is written only once and encapsulated in a friendly manner and reused whenever needed.

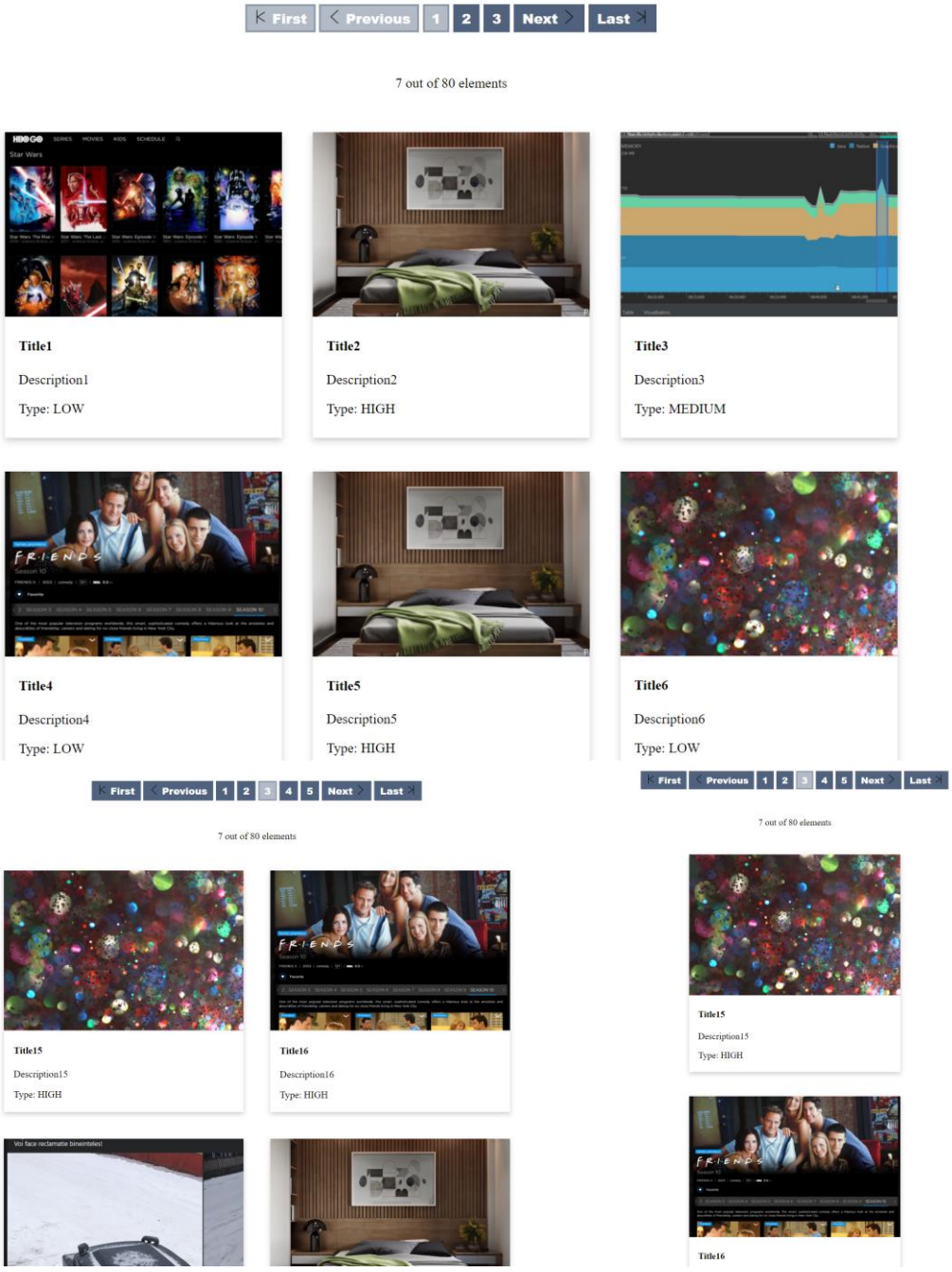
The element has two parts: the html view and the typescript code that represents the data and the logic behind the component. The view elements can be bound to the data, using {{}}, which is the interpolation operator. When the data changes, the view that uses that data also changes and displays the current values.

Apart from this, there is a CSS file that can be used for a component which will apply only to that component, and not the rest of the project. The files are related to each other, as well as the name of the custom tag, in the typescript file of the component, in the "*@Component*" decorator that tells Angular this is a custom element, not a regular class.



2. Component description

The main component is a grid that uses paging to display elements. Another component is the card that is used to display each element in the grid. It shows a photo, a title and a description, with the possibility to redirect on click. It can be used in many different scenarios, such as a page of courses where the users can see all the courses displayed and choose from them, or a page with posts or articles that are behind this clean and easily navigable representation. The content is responsive and arranges the grid in one, two or three columns based on the width of the page, and even the cards themselves change their size accordingly.



a. Navigation bar

The navigation bar contains the First, Prev, Next, Last buttons but also five buttons (or less if the current page is closer to one end) for individual pages centred around the current page. The First button shows the first page, the Prev button shows the previous page, the Next button shows the next page, the Last button shows the last page, and the numbered buttons show the page with that number. The greyed-out buttons are disabled (First, Prev and 1 on the first page; Next, Last and the last page on the last page; the current page on the rest of the pages).

3. Input data and integration

The element receives the page size and a list of elements. The elements must contain at least the following fields: “title”, “imagePath”, “description”, “type” and “onClickHref”. The element can be integrated using the name “my-grid-display”.

```
<my-grid-display [page_size]="7" [elements]="posts"></my-grid-display>
```

```
{
  title: "Title1",
  imagePath: "../../assets/img/image2.png",
  description: "Description1",
  type: "LOW",
  onClickHref: ""
},
```

a. The data validation

The list must not be empty, and the page number must be at least 1 and at most the size of the list.

Incorrect input data: The page size must be at least 1!

Incorrect input data: The page size exceeds the number of elements in the list!

Incorrect input data: The list is empty!

4. Methods and data

The "*@Input*" decorator specifies which variables will hold external data.

The element uses variables to store data related to pages and to the elements that are shown in the current page. The logic in the methods decides what range of list positions are on the current page, how many pages are, what buttons should be visible in the navigation bar and what buttons should be active or disabled.

The "*ngOnInit()*" method is called when the component is initialized. At that point, it already has received the input data which can be processed in this step for the initial state of the component that is going to be displayed. Here, the input data is validated, and the error message is set in case of invalid data, and the values are calculated for the number of pages and the current page is set to the first page.

The following methods are utility methods that are called when the data needs to change. The methods make the necessary modifications and the view that depends on that data changes automatically according to the changes made. The "*computeIdsAndButtons()*" method sets the range of ids of the elements that will be displayed on the current page, and the buttons that must be enabled or disabled for the current page. The "*firstPage()*","*previousPage()*","*nextPage()*" and "*lastPage()*" methods set the current page to be the first, previous, next or last page. They are called when pressing the First, Prev, Next and Last buttons in the navigation bar. The "*moveToPage(page)*" method is called when the page buttons are pressed on the navigation bar and sets the current page to be the selected page. The "*arrayOfIndexes()*" and "*arrayOfPages()*" methods are used to reach all the ids of the elements in the list that are going to be displayed in the current page; and the numbers of the pages that would have buttons in the navigation bar in the current page. The "*isPageCurrentPage()*" method decides if a page button should or should not be disabled. The current page button is disabled always because the component already shows the current page.

5. The View

The display component shows a grid of card components, a navigation bar and the number of elements in a page and the total number of elements. The logic behind the buttons is accessed through "*[disabled]="buttonFirst" (click)="firstPage()"*", which decides when a button should be enabled or disabled and what happens when a button is clicked. For the page buttons, the structure "**ngFor="let **page** of arrayOfPages()"*" gives the page numbers, and the structure "**ngFor="let **index** of arrayOfIndexes()"*" gives the ids of the elements that will be displayed in the current page.
Each card that is displayed in the grid is represented by another type of custom component, accessed by the name *my-card*.

6. The card component

The card component receives one element from the list of elements in the display component and shows a picture, a title, a description and a type, and if clicked it redirects to another page. All the details must be in the input element. This component has the purpose of encapsulating the code that can be very easily reused this way.

```
<div class="grid-container">
  <my-card *ngFor="let index of arrayOfIndexes()" [element]="elements[index]"></my-card>
</div>
```