Práctica Angular – Parte 4

Películas

Refactorizando el componente *peliculas*

Si analizamos con detenimiento la plantilla de `PeliculasComponent` podríamos llegar a la conclusión de que se están llevando a cabo dos responsabilidades en un único componente:

1. Iteración sobre cada objeto `Pelicula` del array de películas.
2. Creación de la "Card" junto con el pintado de cada propiedad del objeto `Pelicula`.

Convendría separar estas dos responsabilidades, en lugar de tenerlas en un mismo componente. En nuestro caso no es muy preocupante, porque no hay mucho código que tratar, pero si queremos seguir las recomendaciones y buenas prácticas en eel diseño de componentes, sería más correcto que la segunda responsabilidad correspondiera a un componente especializado, llamado, por ejemplo, `CardComponent`.

Este cambio nos aportará como ventajas un código más limpio, fácil de mantener y más sencillo de reutilizar. Aunque también nos añade inconvenientes: nos obliga a establecer una jerarquía padre-hijo, donde `PeliculasComponent` será el padre y `CardComponent` será el hijo. Por tanto, será necesario pasar información entre estos dos componentes. El paso de información entre componentes es sencillo cuando el sentido es de padre a hijo pero resulta más tedioso cuando es hijo-padre o entre hermanos (en este último caso tiene que ser hijo1 → padre → hijo2...los hermanos nunca se ven).

Para nuestro caso, la comunicación será:

- Del padre al hijo, ya que el objeto `Pelicula` actual (en curso) lo extraerá el padre del array en cada iteración y se lo pasará al hijo para que éste confeccione la "Card".
- Del hijo al padre, pues el botón que el usuario pulsa para ir a la página que muestra el detalle de una película (con toda la sinopsis) se encuentra ahora en `CardComponent` (aquí es donde sabemos el ID de película de la que queremos mostrar los detalles), mientras que la función que enruta hacia la página de detalle se encuentra en `PeliculasComponent`.

Por tanto, vamos a crear el nuevo componente: abrimos un terminal y desde nuestra carpeta de proyectos Angular ejecutamos lo siguiente para crear la estructura base del proyecto:

```
ng g c components/peliculas/card --flat --spec=false -is
```

Una vez finalizado el proceso anterior, vamos a extraer de la plantilla de `PeliculasComponent` aquello que ahora debe ir en la plantilla del nuevo componente. A continuación se resalta en amarillo lo que tenemos que extraer:

peliculas.component.html

```
<h1>Películas disponibles</h1>
<hr>
<div class="row animated fadeIn fast" *ngIf="peliculas.length == 0">
  <div class="col-md-12">
    <div class="alert alert-info" role="alert">
      No se han encontrado peliculas
    </div>
  </div>
</div>
<div class="card-columns">
  <div class="card animated fadeIn fast" *ngFor="let pelicula of peliculas">
    <img class="card-img-top" [src]="pelicula.img" [alt]="pelicula.titulo">
    <div class="card-body">
      <h4 class="card-title">{{ pelicula.titulo }}</h4>
      <p class="card-text">{{ pelicula.sinopsis | smartTruncate: 100 }}</p>
      <p class="card-text">{{ pelicula.direccion }}</p>
      <p class="card-text">
        <small class="text-muted">{{ pelicula.anio }}</small>
      </p>
       <button  (click)="verPelicula(pelicula.id)"  type="button"  class="btn
btn-outline-primary btn-block">
        Ver más...
      </button>
    </div>
  </div>
</div>
```

Pegamos lo anterior en la plantilla de `card.component.html`, eliminando el código de marcado que viene por defecto:

card.component.html

```
<div class="card animated fadeIn fast" *ngFor="let pelicula of peliculas">
    <img class="card-img-top" [src]="pelicula.img" [alt]="pelicula.titulo">
    <div class="card-body">
      <h4 class="card-title">{{ pelicula.titulo }}</h4>
      <p class="card-text">{{ pelicula.sinopsis | smartTruncate: 100 }}</p>
      <p class="card-text">{{ pelicula.direccion }}</p>
      <p class="card-text">
        <small class="text-muted">{{ pelicula.anio }}</small>
      </p>
       <button  (click)="verPelicula(pelicula.id)"  type="button"  class="btn
btn-outline-primary btn-block">
        Ver más...
      </button>
    </div>
  </div>
```

A continuación tenemos que hacer algunos cambios en este código que acabamos de pegar:

- Eliminamos del primer `div` la directiva estructural `*ngFor`, ya que ahora no tiene sentido en este componente.
- Cambiamos la signatura del método que se ejecutará cuando el usuario pulse el botón para ver el detalle de la película.

card.component.html

```
<div class="card animated fadeIn fast" *ngFor="let pelicula of peliculas">
  <img class="card-img-top" [src]="pelicula.img" [alt]="pelicula.titulo">
  <div class="card-body">
    <h4 class="card-title">{{ pelicula.titulo }}</h4>
    <p class="card-text">{{ pelicula.sinopsis | smartTruncate: 100 }}</p>
    <p class="card-text">{{ pelicula.direccion }}</p>
    <p class="card-text">
      <small class="text-muted">{{ pelicula.anio }}</small>
    </p>
```

```
        <button  (click)="emitirId()"  type="button"  class="btn  btn-outline-
primary btn-block">
       Ver más...
     </button>
   </div>
</div>
```

Ahora necesitamos modificar la clase de `CardComponent` para definir una propiedad `pelicula` e indicar que su valor vendrá dada por el componente padre `PeliculasComponent` (le pasará un objeto `Pelicula` en cada iteración del array). Esto se establece mediante el decorador `@Input()`:

card.component.ts

```
import {Component, Input, OnInit} from '@angular/core';
import {Pelicula} from '../../models/peliculas/pelicula';

@Component({
  selector: 'app-card',
  templateUrl: './card.component.html',
  styles: []
})
export class CardComponent implements OnInit {
  @Input() pelicula: Pelicula;
  constructor() { }
  ngOnInit() {
  }
}
```

A continuación, añadimos lo siguiente a la plantilla de `PeliculasComponent`:

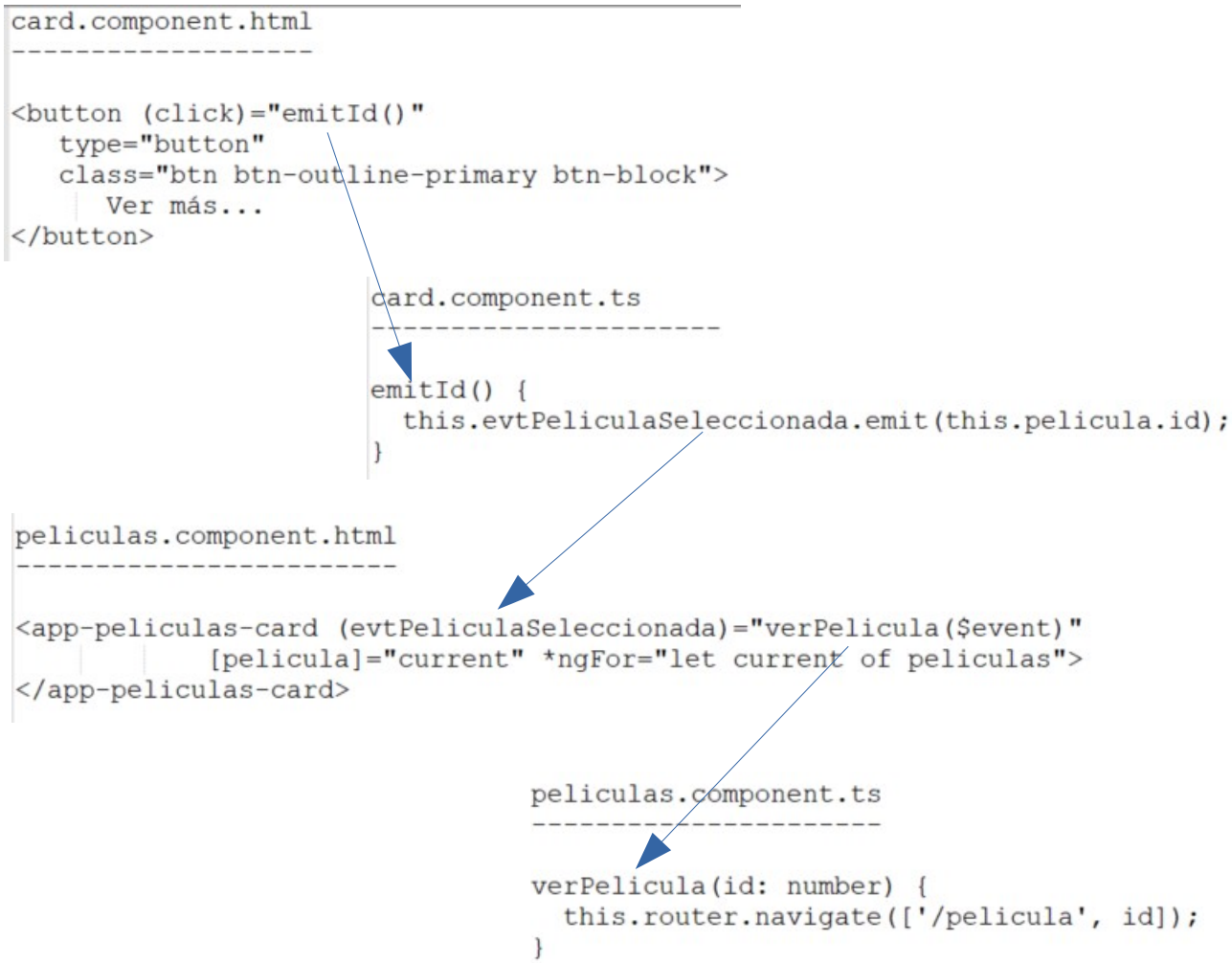
peliculas.component.html

```
<h1>Películas disponibles</h1>
<hr>
<div class="row animated fadeIn fast" *ngIf="peliculas.length == 0">
  <div class="col-md-12">
    <div class="alert alert-info" role="alert">
```

```
      No se han encontrado peliculas
    </div>
  </div>
</div>
<div class="card-columns">
  <app-card [pelicula]="current" *ngFor="let current of peliculas">
  </app-card>
</div>
```

El código anterior en cada iteración le pasa a `CardComponent` cada película (current le hemos llamado aquí). En `CardComponent`, la propiedad se tiene que llamar `pelicula`.

En estos momentos nos debería mostrar todas las películas sin problemas. No obstante, si intentáramos ver el detalle de alguna de ellas, el programa fallaría, ya que nos queda aún trabajo que hacer. Vamos a solucionar esto pero antes vamos a mirar el siguiente diagrama:



Modificamos la clase de `CardComponent`. Los cambios ha realizar son los siguientes:

- Añadir una propiedad de tipo `EventEmitter` para enviar un evento de tipo `number` (el ID de la película de la que se quiere ver el detalle). Esta propiedad será de *salida*, es decir, del hijo hacia el padre, por lo que tenemos que usar el decorador `Output()`.
- Crear un método `emitId()` que será el encargado de enviarle al componente padre el ID de la película. Este método se ejecuta cuando el usuario pulsa el botón "Ver más…" sobre una "Card".

card.component.ts

```
import {Component, EventEmitter, Input, OnInit, Output} from '@angular/core';
import {Pelicula} from '../../models/peliculas/pelicula';

@Component({
  selector: 'app-card',
  templateUrl: './card.component.html',
  styles: []
})
export class CardComponent implements OnInit {

  @Input() pelicula: Pelicula;
  @Output() evtPeliculaSeleccionada: EventEmitter<number>;

  constructor() {
    this.evtPeliculaSeleccionada = new EventEmitter<number>();
  }
  ngOnInit() {
  }
  emitirId() {
    this.evtPeliculaSeleccionada.emit(this.pelicula.id);
  }
}
```

Por último, hemos de modificar la plantilla de `PeliculasComponent` para indicar que este componente quiere manejar el evento del tipo `evtPeliculaSeleccionada`, producido por su componente hijo. y que cuando tal evento se produzca se ejecutará el método `verPelicula()`,

definido en la clase de `PeliculasComponent`. El parámetro `$event` contendrá, en este caso, el ID de la película:

peliculas.component.html

```
<h1>Películas disponibles</h1>
<hr>
<div class="row animated fadeIn fast" *ngIf="peliculas.length == 0">
  <div class="col-md-12">
    <div class="alert alert-info" role="alert">
      No se han encontrado peliculas
    </div>
  </div>
</div>
<div class="card-columns">
  <app-card (evtPeliculaSeleccionada)="verPelicula($event)"
            [pelicula]="current" *ngFor="let current of peliculas">
  </app-card>
</div>
```

Ahora deberíamos poder ir a la página de detalle de una película sin problemas.

Regresar del detalle a la página de resultados de búsqueda

Hay un caso de uso que no funciona de la manera esperada: si hacemos una búsqueda, obtenemos resultados y vamos al detalle de una de estas películas, comprobaremos que al regresar nos devuelve a la página donde aparecen todas las películas, en lugar de donde estábamos, que era la página con los resultados de la búsqueda.

Para solucionar lo anterior, tenemos que hacer unos pocos cambios:

Primero: en la plantilla de `PeliculaComponent` reemplazamos el enlace que teníamos por un botón que capture el evento `clic` y llame a una función llamada `volver()`:

pelicula.component.html

```
<h1 class="animated fadeIn">{{ pelicula.titulo | uppercase }}
<small>({{ pelicula.anio }})</small></h1>
<hr>

<div class="row animated fadeIn fast">
  <div class="col-md-4">
    <img [src]="pelicula.img" class="img-fluid" [alt]="pelicula.titulo">
    <br><br>
          <button (click)="volver()" class="btn btn-outline-danger btn-
block">Regresar</button>
  </div>
  <div class="col-md-8">
    <h3>{{ pelicula.titulo }}</h3>
    <hr>
    <p>{{ pelicula.sinopsis }}</p>
    <hr>
    <p>{{ pelicula.direccion}}</p>
  </div>
</div>
```

Segundo: En la clase de `PeliculaComponent`:

- Inyectamos un objeto del tipo `Location` de Angular
- Creamos el método `volver()`, cuya implementación hace una llamada al método `back()` del objeto `Location`

pelicula.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import {PeliculasService} from '../../services/peliculas.service';
import {Pelicula} from '../../models/peliculas/pelicula';
import {Location} from '@angular/common';

@Component({
  selector: 'app-pelicula',
  templateUrl: './pelicula.component.html',
  styles: []
```

```
})
export class PeliculaComponent implements OnInit {
  ...
  constructor(private activatedRoute: ActivatedRoute,
              private peliculasService: PeliculasService,
              private location: Location) {
    this.activatedRoute.params.subscribe(params => {
      this.pelicula = this.peliculasService.getPelicula(params['id']);
    });
  }
  ngOnInit() {
  }
  volver() {
    this.location.back();
  }
}
```

Ahora ya podemos probar el funcionamiento de este caso de uso.

Conclusión

En parte final de la práctica hemos refactorizado el código para dejarlo más orientado a componentes. No ha sido una tarea sencilla pero ha valido la pena para ver cómo funciona la comunicación entre componentes en Angular.