



► 5. Plantillas, directivas, tuberías, e interceptores


Objetivos

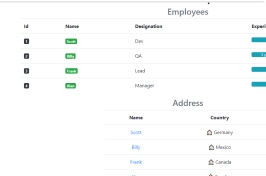
- Disponer de herramientas para reutilizar aspectos visuales y de infraestructura
- Usar tuberías personalizadas
- Directivas que aporten funcionalidad

Referencia

Building angular reusable components using ng-template, ngTemplateOutlet & ngTemplateOutletContext

Recently, I was working on an interesting but very common problem in my project that motivated me to write this article. The problem was like that, I had a re-usable data-table dumb component that I wanted to share across my app & the requirement was to make it configurable from its parent component.

 <https://medium.com/null-exception/building-angular-reusable-components-using-ng-template-ngtemplateoutlet-ngtemplateoutletcontext-f8813b67cd1f>




ID	Name	Designation	Email
1	John	Software Engineer	john.doe@company.com
2	Jane	Product Manager	jane.smith@company.com
3	Mike	Marketing Specialist	mike.brown@company.com
4	Sarah	HR Manager	sarah.white@company.com
5	David	Finance Analyst	david.black@company.com

Name	Country
John	Germany
Jane	Mexico
Mike	Canada
Sarah	USA
David	UK

<https://www.tektutorialshub.com/angular/ng-content-content-projection-in-angular/>

Creating Custom Structural Directive in Angular

TechnoFunnel presents another article on Custom Structural Directives in Angular. In this article, we will talk about What are Structure Directives and How to create Custom Directives with Angular. Structural Directive in Angular are responsible for manipulating, modifying and removing elements

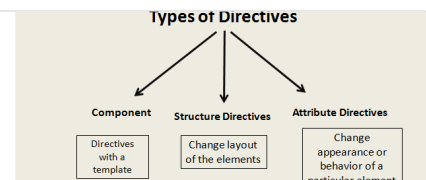
 <https://medium.com/technofunnel/creating-custom-structural-directive-in-angular-79a2862cc169>



Custom Directives in Angular


For those who do a lot of angular, you might have used ng-bind, ng-init somewhere in your project. These are the built-in directives that are provided to us by the angular team. We'll see how to create our very own directive in angular. So to create one, we need to understand different types of custom directives in angular.

<https://nishugoel.medium.com/custom-directives-in-angular-5a843f76bb96>



Creating custom pipes in Angular 11

Pipes are an amazing, useful and clean way to transform data in Angular templates. Angular offers a lot of built-in pipes for data formatting that can be leveraged in our application. I will give you four common pipes examples used in Angular, but if you want to know more about it see the pipes API

 <https://medium.com/nerd-for-tech/creating-custom-pipes-in-angular-11-d4468da38168>



danrevah/ngx-pipes

 Useful pipes for Angular with no external dependencies! - danrevah/ngx-pipes

 <https://github.com/danrevah/ngx-pipes>

danrevah/ngx-pipes

 Useful pipes for Angular with no external dependencies!

22 Contributors 2k Used by 1k Stars 153 Forks



A Comprehensive Guide On Angular HTTP Interceptors - eSparkBiz


Angular is used to build JavaScript-heavy single-page based web applications. Single page apps or SPA's load the entire content of a site within a single page. This single page is usually an index.html file. Today, we will talk about Angular HTTP Interceptors in detail.

 <https://www.esparkinfo.com/angular-http-interceptors.html>



Top 10 ways to use Interceptors in Angular - Angular inDepth

Find out which superpowers you can start using today > Just like Batman develops his gadgets we can use interceptors to gain superpowers. There are many ways to use an interceptor, and I'm sure most of us have only scratched the surface. In this article, I will present my ten favorite ways to use

 <https://indepth.dev/posts/1051/top-10-ways-to-use-interceptors-in-angular>



▶▶ 1 - Plantillas

Card Template component

```
<article class="card mb-4">
  <header class="card-header">
    <p class="card-header-title">
      <a *ngIf="card.link" [routerLink]="[card.link]">{{ card.title }}</a>
      <a *ngIf="card.href" [href]="card.href">{{ card.title }}</a>
      <span *ngIf="!card.link && !card.href">{{ card.title }}</span>
    </p>
  </header>
  <main class="card-content">
    <div *ngIf="!!card.description; else noDescription" class="content">
      {{ card.description }}
    </div>
  </main>
</article>
<ng-template #noDescription>
  <ng-content select=".content">
</ng-content>
</ng-template>
```

Uso en ResourcePage

```
<main>
  <ab-ui-header [header]="header"></ab-ui-header>
  <ab-ui-card>
    <pre class="content">{{ resource | json }}</pre>
  </ab-ui-card>
</main>
```



Las plantillas sirven para homogeneizar la presentación.

▶▶ 2 - Directivas

 Track

```
@Directive({
  selector: '[abUiTrack]',
```

```

})
export class TrackDirective {
  @Input('abUiTrack') label?: string;

  @HostListener('click') onClick() {
    this.trackUserInteraction('CLICK');
  }
  constructor(private el: ElementRef, private tracker: TrackerStoreService) {}

  private trackUserInteraction(action: string) {
    this.tracker.trackEntry({
      category: 'BUSINESS',
      event: action,
      label: this.getLabel(),
    });
  }
  private getLabel() {
    if (this.label) return this.label;
    const native = this.el.nativeElement;
    return (
      native.id || native.name || native.value || native.innerHTML || 'unknown'
    );
  }
}

```

Uso en Card Template

```

<a *ngIf="card.link" [routerLink]="[card.link]" abUiTrack>{{ card.title }}</a>
<a *ngIf="card.href" [href]="card.href" [abUiTrack]="card.href" export class TruncatePipe implements PipeTransform {
  transform(value: unknown, ...args: unknown[]): unknown {
    const source = value as string;
    if (!source) return '';

    const defaultLimit = 20;
    const limitArg = args[0] as string;
    const limit = limitArg ? parseInt(limitArg) : defaultLimit;

    if (source.length <= limit) return source;

    const defaultTrail = '...';
    const trailArg = args[1];
    const trail = trailArg || defaultTrail;
    const truncated = source.substring(0, limit) + trail;

    return truncated;
  }
}>{{ card.title }}</a>

```

3 - Tuberías custom

```
ng g @schematics/angular:pipe pipes/truncate --project ui --export
```

```

@Pipe({
  name: 'truncate',
})
export class TruncatePipe implements PipeTransform {
  transform(value: unknown, ...args: unknown[]): unknown {
    const source = value as string;
    if (!source) return '';

    const defaultLimit = 20;
    const limitArg = args[0] as string;
    const limit = limitArg ? parseInt(limitArg) : defaultLimit;

    if (source.length <= limit) return source;

    const defaultTrail = '...';

```

```

const trailArg = args[1];
const trail = trailArg || defaultTrail;
const truncated = source.substring(0, limit) + trail;

return truncated;
}
}

```

Uso en la misma CardTemplate

```

<div *ngIf="!!card.description; else noDescription" class="content">
  {{ card.description | truncate:50}}
</div>@Injectable()
export class AdapterInterceptor implements HttpInterceptor {
  constructor() {}

  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    return next.handle(request).pipe(
      filter((event) => event instanceof HttpResponse),
      map((eventResponse) => eventResponse as HttpResponse<any>),
      map((httpResponse) => this.adaptResponse(httpResponse))
    );
  }

  adaptResponse(httpResponse: HttpResponse<any>) {
    const body = httpResponse.body;
    const adaptedBody = body['data'] || [];
    const adaptedResponse = httpResponse.clone({ body: adaptedBody });
    return adaptedResponse;
  }
}

```

4 - Interceptores

Otros ejemplos de interceptores.

Adaptador

```

@Injectable()
export class AdapterInterceptor implements HttpInterceptor {
  constructor() {}

  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    return next.handle(request).pipe(
      filter((event) => event instanceof HttpResponse),
      map((eventResponse) => eventResponse as HttpResponse<any>),
      map((httpResponse) => this.adaptResponse(httpResponse))
    );
  }

  adaptResponse(httpResponse: HttpResponse<any>) {
    const body = httpResponse.body;
    const adaptedBody = body['data'] || [];
    const adaptedResponse = httpResponse.clone({ body: adaptedBody });
    return adaptedResponse;
  }
}

```

Retry

```

const RETRY_MAX = 3;
const DELAYED_RETRY_MS = 3000;

@Injectable()
export class RetryInterceptor implements HttpInterceptor {
  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    return next
      .handle(request)
      .pipe(retryWhen((error$) => this.tryRetry(error$)));
  }

  tryRetry(error$: Observable<HttpErrorResponse>) {
    return error$.pipe(
      concatMap((error, count) => {
        if (this.canRetry(error, count)) {
          return of(error);
        } else {
          return throwError(error);
        }
      }),
      delay(DELAYED_RETRY_MS)
    );
  }

  private canRetry(error: HttpErrorResponse, count: number) {
    return (error.status == 0 || error.status >= 500) && count < RETRY_MAX;
  }
}

```

Seguridad

```

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  constructor(private service: AuthService) {}
  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    const authorization = 'Bearer ' + 'session token from AuthService';
    request = request.clone({
      setHeaders: {
        Authorization: authorization,
      },
    });
    return next.handle(request);
  }
}

```

Registro de proveedores

```

@NgModule({
  imports: [CommonModule, HttpClientModule],
  exports: [HttpClientModule],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: RetryInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: AdapterInterceptor, multi: true },
  ],
})
export class CoreModule {}

```



Para más ejemplos ver implementación del `TrackerInterceptor`



Alberto Basalo