# 0 settings.json

{

*"workbench.colorTheme"*: "Monokai Night",

*"workbench.iconTheme"*: "material-icon-theme",

*"files.autoSave"*: "afterDelay",

*"editor.tabSize"*: 2,

*"prettier.printWidth"*: 100,

*"prettier.tabWidth"*: 2,

*"prettier.useTabs"*: false,

*"prettier.singleQuote"*: true,

*"prettier.jsxSingleQuote"*: true,

*"prettier.trailingComma"*: "none",

*"prettier.bracesSpacing"*: true,

*"prettier.bracketSpacing"*: true,

*"prettier.breakProperty"*: true,

*"prettier.arrowParens"*: true,

*"prettier.semi"*: false,

*"prettier.spaceBeforeFunctionParen"*: true,

*"prettier.alignObjectProperties"*: false,

*"[typescript]"*: {

*"editor.formatOnSave"*: true,

*"editor.defaultFormatter"*: "esbenp.prettier-vscode"

    }

}

# 1 Instalaciones

Npm i bootstrap

npm i @fortawesome/fontawesome-free

npm i --save-dev json-server

npm i --save-dev jsonwebtoken

**angular.json**

*"styles"*: [

"src/styles.css",

"node\_modules/@fortawesome/fontawesome-free/css/all.min.css",

"node\_modules/bootstrap/dist/css/bootstrap.min.css"

],

# 2 Preparando el servicio web RESTful

**Package.json**

*"scripts"*: {

*"ng"*: "ng",

*"start"*: "ng serve",

*"build"*: "ng build",

*"watch"*: "ng build --watch --configuration development",

*"test"*: "ng test",

*"json"*: "json-server restData.js -p 3500"

  },

**restData.js**

*module*.*exports* = *function* (){

*var* data = {

    products:[

      {id:1,name:'kayak',category:'watersports',price:275,details:{supplier:'acme',keywords:['boat','small']}},

      {id:2,name:'lifejacket',category:'watersports',price:48.95,details:{supplier:'smoot co',keywords:['safety']}},

      {id:3,name:'soccer ball',category:'soccer',price:19.50},

      {id:4,name:'corner flags',category:'soccer',price:34.95},

      {id:5,name:'stadium',category:'soccer',price:79500},

      {id:6,name:'thinking cap',category:'chess',price:16},

      {id:7,name:'unsteady chair',category:'chess',price:29.95},

      {id:8,name:'human chess board',category:'chess',price:75},

      {id:9,name:'bling bling king',category:'chess',price:1200}

    ]

  }

  return data;

}

**Para ejecutar este REST 🡪 npm run json**

**http://localhost:3500/products**

[**http://localhost:3500/products/1**](http://localhost:3500/products/1)

# 3 el modelo de datos

* 1 o más clases que describen la data en el modelo
* 1 data source, que carga y graba la data en un servidor
* 1 repository, que permite manipular la data del modelo

Model/producto.model.ts 🡪 Product

Model/datasource.model.ts 🡪 SimpleDataSource

Model/repository.model.ts 🡪 Model

# 4 GITHUB

https://github.com/manolinlao/mlao-shop.git

# 5 Directivas de atributo

Modifican el elemento host al que hacen referencia

Las directivas suelen llevar un prefijo distingivo, en este caso he puesto Mlao

**RAMA: directivas-atributo**

p.ej -> mlao-attr

## Usando @Attribute

Para pasar datos fijos

Attr.directive.ts

import { Attribute, Directive, ElementRef } from '@angular/core';

@Directive({

  selector: '[mlao-attr]',

})

export *class* MlaoAttrDirective {

*constructor*(*element*: ElementRef, @Attribute('mlao-attr') *bgClass*: *string*) {

    element.nativeElement.classList.add(bgClass || 'bg-success', 'fw-bold');

  }

}

Producto.component.html

<td mlao-attr>{{item.name}}</td>

<td mlao-attr="bg-warning">{{item.category}}</td>

<td mlao-attr="bg-info">{{item.price}}</td>

## Dinámicas y con envío de eventos

Attr-dynamic.directive.ts

import {

  Directive,

  ElementRef,

  EventEmitter,

  HostBinding,

  HostListener,

  Input,

  Output,

  SimpleChanges,

} from '@angular/core';

import { Product } from '../model/product.model';

@Directive({

  selector: '[mlao-attr-dyn]',

})

export *class* MlaoAttrDynamicDirective {

  // bindings on the host element are defined using 2 decorators: @HostBinding y @HostListener

  @Input('mlao-attr-dyn') @HostBinding('class') bgClass: *string* | *null* = '';

  //esta directiva también lanza eventos

  @Input('mlao-product-dyn') product: Product = new Product();

  @Output('mlao-category-dyn') click = new EventEmitter<*string*>();

  @HostListener('click') triggerCustomEvent() {

    if (this.product != null) {

      this.click.emit(this.product.category);

    }

  }

*constructor*(private *element*: ElementRef) {

    // podemos usar esto para capturar eventos, pero esto sólo funciona en aplicaciones

    // que corran en un web browser, porque juega con el DOM

    // para evitar esto usaremos el @HostListener-@HostBinding

    /\*

    this.element.nativeElement.addEventListener('click', () => {

      if (this.product != null) {

        this.click.emit(this.product.category);

      }

    });

    \*/

  }

  /\*

  // no usamos esto porque esto no controla los cambios

  ngOnInit() {

    this.element.nativeElement.classList.add(

      this.bgClass || 'bg-success',

      'fw-bold'

    );

  }

  \*/

  ngOnChanges(*changes*: SimpleChanges) {

*let* change = changes['bgClass'];

*let* classList = this.element.nativeElement.classList;

    if (!change.isFirstChange() && classList.contains(change.previousValue)) {

      classList.remove(change.previousValue);

    }

    if (!classList.contains(change.currentValue)) {

      classList.add(change.currentValue);

    }

  }

}

Product.component.html

<tr \*ngFor="let item of getProducts();let i=index"

          [mlao-attr-dyn]="getProducts().length<6 ? 'bg-success' : 'bg-warning'"

          [mlao-product-dyn]="item"

          (mlao-category-dyn)="newProduct.category=$event">

        <td [mlao-attr-dyn]="item.category=='Soccer' ? 'bg-info':null">{{i+1}}</td>

## 2-way data binding

Twoway-directive.ts

import {

  Directive,

  EventEmitter,

  HostBinding,

  HostListener,

  Input,

  Output,

  SimpleChange,

} from '@angular/core';

@Directive({

  selector: '[mlao-twoway-attr]',

})

export *class* TwowayDirective {

  @Input('mlao-twoway-attr') modelProperty: *string* | *undefined* = '';

  @HostBinding('value') fieldValue: *string* = '';

  @Output('mlao-twoway-change') update = new EventEmitter<*string*>();

  @HostListener('input', ['$event.target.value']) updateValue(

*newValue*: *string*

  ) {

    this.fieldValue = newValue;

    this.update.emit(newValue);

  }

*constructor*() {}

  ngOnChanges(*changes*: { [*property*: *string*]: SimpleChange }) {

*let* change = changes['modelProperty'];

    if (change.currentValue != this.fieldValue) {

      this.fieldValue = changes['modelProperty'].currentValue || '';

    }

  }

}

Product.component.html

    <div class="form-group bg-info text-white p-2">

      <label>Name:</label>

      <input class="bg-primary text-white form-control"

              [mlao-twoway-attr]="newProduct.name"

              (mlao-twoway-change)="newProduct.name=$event"/>

    </div>

# 6 Directivas estructurales

Cambian el layout del documento html, añadiendo o quitando elementos.

Sus nombres va precedidos por \*, por ejemplo \*ngIf y \*ngFor son directivas de atributo

**RAMA: directivas-estructurales**

[mlao-if] \*mlao-ig

## Podemos usar el <ng-template>

 <div class="form-check m-2">

      <input type="checkbox" class="form-check-input" [(ngModel)]="showTable"/>

      <label class="form-check-label">Show Table</label>

    </div>

    <ng-template [mlao-if]="showTable">

…. Cosas a mostrar

</ng-template>

Structure.directive.s

import {

  Directive,

  Input,

  SimpleChanges,

  TemplateRef,

  ViewContainerRef,

} from '@angular/core';

@Directive({

  selector: '[mlao-if]',

})

export *class* StructureDirective {

  @Input('mlao-if') expressionResult: *boolean* | *undefined*;

*constructor*(

    private *container*: ViewContainerRef,

    private *template*: TemplateRef<Object>

  ) {}

  ngOnChanges(*changes*: SimpleChanges) {

*let* change = changes['expressionResult'];

    if (!change.isFirstChange() && !change.currentValue) {

      this.container.clear();

    } else if (change.currentValue) {

      this.container.createEmbeddedView(this.template);

    }

  }

}

## Usando la sintaxis Concise Structural Directive 🡪 \*^

<div \*mlao-if="showTable">

      usando \*mlao-if

    </div>

## Iterating structural directives

[mlao-for]

Podemos usar sintaxis con []

<tbody>

<ng-template [mlao-for]="getProducts()" let-item let-i="index" let-odd="odd" let-even="even">

          <tr [class.table-info]="odd" [class.table-warning]="even">

            <td>{{i+1}}</td>

            <td>{{item.name}}</td>

            <td>{{item.category}}</td>

            <td>{{item.price}}</td>

          </tr>

</ng-template>

</tbody>

O sintaxis con \*

 <tr \*mlao-for="getProducts(); let item=item; let i=index;let odd=odd;let even=even"

            [class.table-info]="odd" [class.table-warning]="even">

                  <td>{{i+1}}</td>

                  <td>{{item.name}}</td>

                  <td>{{item.category}}</td>

                  <td>{{item.price}}</td>

        </tr>

Iterator.directive.ts

import { Directive, ViewContainerRef, TemplateRef, Input } from '@angular/core';

@Directive({

  selector: '[mlao-for]',

})

export *class* IteratorDirective {

  @Input('mlao-for') dataSource: *any*;

*constructor*(

    private *container*: ViewContainerRef,

    private *template*: TemplateRef<Object>

  ) {}

  ngOnInit() {

    console.log('ngOnInit', this.dataSource);

    this.container.clear();

    for (*let* i = 0; i < this.dataSource.length; i++) {

      console.log(i);

      this.container.createEmbeddedView(

        this.template,

        new IteratorContext(this.dataSource[i], i, this.dataSource.length)

      );

    }

  }

}

*class* IteratorContext {

  odd: *boolean*;

  even: *boolean*;

  first: *boolean*;

  last: *boolean*;

  item: *any*;

*constructor*(public *$implicit*: *any*, public *index*: *number*, *total*: *number*) {

    this.odd = index % 2 == 1;

    this.even = !this.odd;

    this.first = index == 0;

    this.last = index == total - 1;

    this.item = $implicit;

  }

}

# 7 usando componentes, pipes, servicios, y service providers

**RAMA: componentes-basico**

## Uso de <ng-content>

<app-toggle-view>

<app-product-table [model]="model"></app-product-table>

</app-toggle-view>

Toggleview.component.html

<div class="form-check">

  <label class="form-check-label">Show Content</label>

  <input class="form-check-input" type="checkbox" [(ngModel)]="showContent"/>

</div>

<ng-content \*ngIf="showContent"></ng-content>

Toggleview.component.ts

export *class* ToggleViewComponent {

  showContent: *boolean* = true;

}

## @ViewChildren

## Pipes y custom pipes @Pipe

{{ data | pipe\_name:”argumeno”:”argunmento”:”argumento”:….

## Impure pipes 🡪 @pure

El valor por defecto de la propiedad pure es true -> le dice a angular que el método transform del pipe genere un nuevo valor sólo si la input data value cambia o cuando uno o más de sus argumentos es modificado.

Usar un pipe **impuro** le dice a Angular que el pipe tiene su propio “state data”. En estos casos se invoca al método transform incluso cuando no ha habido cambios en la data o en los argumentos.

El ejemplo lo vemos con **mlaoCategoryFilter** pipe

@Pipe({

  name: 'mlaoCategoryFilter',

  pure: false,

})

  <tr \*ngFor="let item of getProducts() | mlaoCategoryFilter:categoryFilter; let i=index; let odd=odd; let even=even"

        [class.bg-info]="odd" [class.bg-warning]="even">

Si añadimos algun elemento más a la tabla, si el pipe mlaoCategoryFilter es puro, no cambia la visualización porque aunque el array de elementos de la tablaha cambiado no se entera porque ha cambiado un objeto dentro del array retornado por getProducts, por eso ponemos el pure:false

## Servicios -> @Injectable

Los servicios son objetos que definen la funcionalidad requerida por otros bloques de código como componentes o directivas..

Si los hacemos a pelo, sin usar ng g s hay que declarar los servicios en los **providers**

**En el ejemplo usamos el servicio en una pipe 🡪 discount.pipe.ts**

# 8 Angular Avanzado

Rutas, forms reacgtivos, etc

**RAMA: avanzado**

**RAMA: avanzado-templateform**

Ejemplo básico usando formsmodule

Se ve la comunicación usando los servicios

**RAMA: avanzado-reactiveform**

Siguiente paso pero usando formularios reactivos

# 9 Formularios reactivos

**RAMA: avanzado-reactiveform**