

**09**

# **Desenvolupament Web en entorn Client**

**Cicle: Desenvolupament d'aplicacions web**

**Curs 2024-2025**

©José Masip Alonso

## 1. Introducció

En aquest document aprofundirem un poc més en Angular.

## 2. Contingut

### 2.1. Serveis.

Un servei és una classe amb el propòsit de mantenir una lògica (i dades) entre distints components de l'aplicació. És útil per a compartir dades entre components i per a accedir a dades externes (servei web a través d'AJAX).

Al document anterior crearem una estructura que teníem distintes frutes.

```
elmeuprojecte/src/app/frutes/frutes.component.ts
```

```
...  
  
export class FrutesComponent {  
  llistat: Fruta[] = [  
    {identificador: 1, nom:"taronja"},  
    {identificador:2, nom:"llima"},  
    {identificador:3, nom:"melo"}  
  ];  
}
```

Anem a modificar aquest plantejament per a utilitzar un servei.

Primer que res, cal crear el servei amb:

```
ng g service FrutesServei
```

```
pepe@pepe-mint:~$ docker start Angulardwc -i  
Starting virtual X frame buffer: Xvfb.  
Executing command /bin/bash  
root@0b02317ee065:/opt/app# cd elmeuprojecte/  
root@0b02317ee065:/opt/app/elmeuprojecte# ng g service FrutesServei  
CREATE src/app/frutes-servei.service.spec.ts (388 bytes)  
CREATE src/app/frutes-servei.service.ts (141 bytes)
```

Genera l'arxiu del servei amb el següent contingut:

```
elmeuprojecte/src/app/frutes-servei.service.ts
```

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

@Injectable({
  providedIn: 'root'
})
export class FrutesServeiService {

  constructor() { }
}
```

Com ara les frutes les tenim al servei, cal fer que el nostre anterior component FrutesComponent les obtinga d'ahí. Modifiquem aquest cridant al servei.

```
elmeuprojecte/src/app/frutes-servei.service.ts
```

```
import { Injectable } from '@angular/core';
import { Fruta } from './frutes/fruta';

@Injectable({
  providedIn: 'root'
})
export class FrutesServeiService {

  constructor() { }

  getFrutes(): Fruta[] {
    return [
      {identificador: 1, nom:"taronja"},
      {identificador:2, nom:"llima"},
      {identificador:3, nom:"melo"},
      {identificador:4, nom:"kiwi"},
      {identificador:5, nom:"kaki"}
    ];
  }
}
```

Nou codi del component:

```
elmeuprojecte/src/app/frutes/frutes.component.ts

import { Component, OnInit } from '@angular/core';
import { Fruta } from '../fruta';
import { FrutesServeiService } from '../frutes-servei.service';

@Component({
  selector: 'app-frutes',
  templateUrl: './frutes.component.html',
  styleUrls: ['./frutes.component.css']
})
export class FrutesComponent {
  llistat: Fruta[] = [];

  constructor(private FrutesServeiService: FrutesServeiService) { }

  ngOnInit() {
    this.llistat = this.FrutesServeiService.getFrutes();
  }
}
```

Ara explicarem les modificacions a l'arxiu:

- Al constructor de la classe definim un parametre i Typescript el declararà com un atribut de la classe i fa l'assignació automàtica.
- Definim l'atribut que enllaça al servei. Cal importar-lo.
- OnInit – s'executa quan el component és carrega per primera volta. D'aquesta manera obtenem les dades del servei i les assignem a l'atribut llistat. Per a utilitzar-lo cal importar-lo (observa la 1<sup>a</sup> linia).

## 2.2. AJAX.

En una aplicació normal les dades s'obtenen d'un servidor web mitjançant peticions HTTP en segon pla (AJAX). El servei HttpClient d'Angular s'encarrega d'açò (cal importar el mòdul HttpClientModule).

Per a poder utilitzar-lo cal injectar-lo en el constructor de la classe.

Modifiquem el servei anterior per a obtenir les dades per AJAX. Inicialment per a entendre millor els conceptes farem la petició AJAX com hem vist als documents anteriors (sense utilitzar Angular).

Primer crearem un php que fa la consulta a la base de dades i retorna el json. En aquest cas, simplement retorne un json.

<http://localhost:8080/dwc/01angularajax.php>

```
<?php
header('Access-Control-Allow-Origin: *');

echo('[
{"identificador": 1, "nom": "taronja"},
{"identificador": 2, "nom": "llima"},
{"identificador": 3, "nom": "melo"},
{"identificador": 4, "nom": "kiwi"},
{"identificador": 5, "nom": "kaki"}
]');
?>
```

En el meu cas estic utilitzant el docker de xampp que ja utilitzavem als documents anteriors. Per tant, des del docker de angular accedirem al docker de xampp. Com no són el mateix servidor dona un error si no afegim la línia del header.

Solicitud desde otro origen bloqueada: la política de mismo origen impide leer el recurso remoto en http://localhost:8080/dwc/01angularajax.php (razón: falta la cabecera CORS 'Access-Control-Allow-Origin'). Código de estado: 200. 2...

**NOTA:** Recorda arrancar els 2 dockers o el servei del xampp.

Al servei realitzarem les següents modificacions:

```
elmeuprojecte/src/app/frutes-servei.service.ts
```

```
import { Injectable } from '@angular/core';
import { Fruta } from './frutes/fruta';

@Injectable({
  providedIn: 'root'
})
export class FrutesServeiService {
  constructor() { }

  getFrutes() {
    var v:Fruta[] =[]; //inicialitze el vector

    let peticio_http = new XMLHttpRequest();
    // Prepara funcio rep dades del servidor
    peticio_http.onreadystatechange = repDades;
    // Realizar peticio HTTP
    peticio_http.open('POST', 'http://localhost:8080/dwc/01angularajax.php', true);
    //indiquem que les dades estan codificades com un formulari
    peticio_http.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
    //comence el proces. envie les dades
    peticio_http.send(null);

    function repDades() {
      if(peticio_http.readyState == 4) {
        if(peticio_http.status == 200) { //si tot OK
          //alert(peticio_http.responseText);

          let vjson=JSON.parse(peticio_http.responseText);
          for (let e=0;e<vjson.length; e++){
            v.push({"identificador":vjson[e].identificador,"nom":vjson[e].nom})
          }
        }
        else{ //mostra el estat en format de text
          alert("error: "+peticio_http.statusText);
        }
      }// readyState=4
    }
    //console.log(v);
    return v;
  }
}
```

La funció que rep les dades les transforma a json i les transforma al tipus Fruta. Angular s'encarrega de mostrar-les.

### 2.3. Ajax amb Angular

Per a explicar aquesta opció crearem un nou servei amb les modificacions necessaries.

```
ng g service FrutesServei2
```

Importarem el servei HttpClient del mòdul necessari modificant l'arxiu:

```
elmeuprojecte/src/app/app.module.ts

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';
import { FrutesComponent } from './frutes/frutes.component';

import { HttpClientModule } from '@angular/common/http'; //nou

@NgModule({
  declarations: [
    AppComponent,
    FrutesComponent
  ],
  imports: [
    HttpClientModule, //nou
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

A continuació modificarem el nou servei creat:

```
elmeuprojecte/src/app/frutes-servei2.service.ts
```

```
import { Injectable } from '@angular/core';
import { Fruta } from './frutes/fruta';

import { HttpClient } from '@angular/common/http'; //nou

@Injectable({
  providedIn: 'root'
})
export class FrutesServei2Service {
  constructor(private http: HttpClient) { } //nou

  getFrutes() {
    var v:Fruta[] =[];

    this.http.get("http://localhost:8080/dwc/01angularajax.php").subscribe((data:
    any) => {
      console.log(data+"data:"+JSON.stringify(data));
      /*
      data ----- [object Object],[object Object],[object Object],[object Object],
      [object Object]
      JSON.stringify(data) ----- [{"identificador":1,"nom":"taronja"}, {"i...,
      {"identificador":5,"nom":"kaki"}]
      */
      let vjson=JSON.parse(JSON.stringify(data));
      for (let e=0;e<vjson.length; e++){
        v.push({"identificador":vjson[e].identificador,"nom":vjson[e].nom})
      }
    }); ;
    return v;
  }
}
```

#### Modificacions:

- Importem HttpClient.
- Al constructor `constructor(private http: HttpClient)` enllacem aquest amb un atribut del servei.



- A través de http realitzem la petició ajax `this.http.get("http://localhost:8080/dwc/01angularajax.php")`
- Aquesta petició retorna un objecte Observable. Aquest objecte permet que tots els mòduls que es subscriuen a ell puguin obtenir les modificacions fetes sobre aquest. `.subscribe()`
- Com són objectes les dades rebudes les pasem a string per a poder fer el `JSON.Parse()`.

A continuació modificarem el component per a mostrar les dades dels dos serveis.

```
elmeuprojecte/src/app/frutes.component.html
```

```
<h2>ajax normal</h2>
<p *ngFor="let f of llistat">{{f.identificador + " - " + f.nom}}</p>

<h2>ajax angular</h2>
<p *ngFor="let f of llistat2">{{f.identificador + " - " + f.nom}}</p>
```

## mostrar les frutes

### ajax normal

1 - taronja  
2 - llima  
3 - melo  
4 - kiwi  
5 - kaki

### ajax angular

1 - taronja  
2 - llima  
3 - melo  
4 - kiwi  
5 - kaki

```
elmeuprojecte/src/app/frutes.component.ts
```

```
import { Component, OnInit } from '@angular/core';
import { Fruta } from './fruta';
import { FrutesServeiService } from '../frutes-servei.service';
import { FrutesServei2Service } from '../frutes-servei2.service';

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

export class FrutesComponent {
  llistat: Fruta[] = [];
  llistat2: Fruta[] = [];

  constructor(private FrutesServeiService: FrutesServeiService, private
FrutesServei2Service: FrutesServei2Service) { }

  ngOnInit() {
    this.llistat = this.FrutesServeiService.getFrutes();
    this.llistat2 = this.FrutesServei2Service.getFrutes();
  }
}
```

Si necessitem passar dades al servidor web (pàgina php que genera el json/XML) simplement afegir aquest valor a la url.

```
this.http.get("http://localhost:8080/dwc/01angularajax.php?var=1")
```

#### 2.4. Mètodes de HTTP



Hi ha distints tipus de peticions HTTP que podem utilitzar ([https://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol#Request\\_methods](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods)).

Al nostre exemple hem utilitzat una petició GET. Aquesta petició sols rep dades dels servidor web. Equivaldria a un SELECT en SQL.

Tambè existeix la petició POST per a insertar noves dades en el servidor (Equivaldria a un INSERT de MYSQL) o PUT que actualitza les dades del servidor (Equivaldria a un UPDATE de MYSQL) o DELETE que elimina unes dades del servidor (Equivaldria a un DELETE de MYSQL) .

## 2.5. Vinculació d'un camp de formulari.

Angular permet la vinculació entre un camp i una variable d'un atribut d'un component o classe.

elmeuprojecte/src/app/app.component.html
<pre>&lt;input type="text" [(ngModel)]="this.variableCamp"/&gt; {{this.variableCamp}}</pre>
elmeuprojecte/src/app/app.component.ts
<pre>import { Component } from '@angular/core'; import { NgForm } from '@angular/forms';  @Component({   selector: 'app-root',   templateUrl: './app.component.html',   styleUrls: ['./app.component.css'] }) export class AppComponent {   title = 'hola mon';   vector = ['aaaaa', 'aaaaa', 'aaaaa', 'aaaaa', 'aaaaa'];   a = 33;   variableCamp:string="aa"; }</pre>



En el app.module.ts afegim `import {FormsModule} from '@angular/forms';`  
i en el import

```
@NgModule({  
  declarations: [  
    AppComponent,  
    ...  
    FrutesComponent  
  ],  
  imports: [  
    HttpClientModule,  
    BrowserModule,  
    FormsModule  
  ],  
  ...  
})
```

A cada pulsació en el camp es modificarà el valor del costat del camp. Açò ocorreix perquè la vinculació és bidireccional.

## 2.6. Vinculació d'un event.

Els events els definim en la vista i criden a funcions del component. Per tant, es considera que hi ha un fluxe d'informació i cal definir-lo entre ().

elmeuprojecte/src/app/app.component.html

```
<button name="boto" (click)="fafegir()">boto afegir</button>
```

elmeuprojecte/src/app/app.component.ts

```
...  
export class AppComponent {  
  title = 'hola mon';  
  vector = ['aaaa', 'eeeeee', 'ieeeeeee', 'oooooooo', 'ueeeeeee'];  
  a = 33;  
  variableCamp:string="aa";  
  
  fafegir(){  
    this.vector.push(this.variableCamp);  
    console.log(this.vector);  
  }  
}
```

A l'exemple cadascuna de les voltes que polsem el boto afegim al vector el valor del camp.

### 2.7. @input – comunicació entre pare i fill.

En aquest punt vorem com comunicar un component pare amb un fill. A la nostra aplicació el component pare es la app-component i un fill seria el component frutes ja que el importem dins de l'altre.

Cal modificar tant el component pare com el fill. Al fill cal importar input i definir un atribut com a que rebrà les dades del pare. I a la vista simplement mostrarem aquest valor al nostre exemple.

Modificacions al component:

```
elmeuprojecte/src/app/frutes.component.html
<h2>ajax normal</h2>
<p *ngFor="let f of llistat">{{f.identificador + " - " + f.nom}}</p>

<h2>ajax angular</h2>
<p *ngFor="let f of llistat2">{{f.identificador + " - " + f.nom}}</p>

valor camp {{valorcamp}}

elmeuprojecte/src/app/frutes.component.ts
import { Component, OnInit, Input } from '@angular/core'; //importar input
import { Fruta } from './fruta';
...
...
export class FrutesComponent {
  llistat: Fruta[] = [];
  llistat2: Fruta[] = [];
  @Input() valorcamp = "";
  ...
}
```

Al component pare cal afegir on utilitzem el selector del component la vinculació d'aquesta comunicació. Enllaçarem l'atribut del fill amb l'atribut del component pare.

```
elmeuprojecte/src/app/app.component.html  
  
<button name="boto" (click)="fafegir()">boto afegir</button>  
  
<app-frutes [valorcamp]="this.variableCamp"></app-frutes>
```

## 2.8. @output – comunicació entre fill i el pare.

Ara vorem la comunicació inversa. El fill llença un event per a comunicar-se amb el pare.

Per a tal fi, crear un component nou totfrutes. A aquest cal importar output i definir un event com es mostra a continuació.

```
elmeuprojecte/src/app/topfrutes.component.html  
  
<p>totfrutes works!</p>  
<input type="button" value="boto" (click)="topactualitzar('poiuy')">  
  
elmeuprojecte/src/app/topfrutes.component.ts  
  
import { Output, EventEmitter, Component } from '@angular/core';  
  
@Component({  
  selector: 'app-totfrutes',  
  templateUrl: './totfrutes.component.html',  
  styleUrls: ['./totfrutes.component.css']  
})  
export class TotfrutesComponent {  
  constructor() { }  
  
  @Output() topfrutesEvent = new EventEmitter<string>();  
  //creem un nou emisor d'events que enviara un string  
  
  topactualitzar(a:string="qwerty") {  
    this.topfrutesEvent.emit(a); //enviem el valor per l'event  
  }  
}
```

Al component pare cal afegir on utilitzem el selector del component la vinculació d'aquesta comunicació . Enllaçarem event del fill amb una funció del component pare que rep l'event.

D'aquesta manera hem vinculat el valor que passem des del nou component amb el camp que teniem al component pare.

```
elmeuprojecte/src/app/app.component.html
```

```
<button name="boto" (click)="fafegir()">boto afegir</button>

<app-frutes [valorcamp]="this.variableCamp"> </app-frutes>

<app-totfrutes (topfrutesEvent)="PareRepFill($event)"></app-totfrutes>
```

```
elmeuprojecte/src/app/app.component.html
```

```
export class AppComponent {
  title = 'hola mon';
  vector = ['aaaaa','aaaaaa','iaaaaaaa','oaaaaaa','uaaaaaa'];
  a = 33;
  variableCamp:string="aa";
  ...
  PareRepFill(aux:string){
    this.variableCamp=aux;
    console.log(aux);
  }
}
```

### 3. Activitats.

**NOTA:**Crea un projecte amb el teu nom i dins realitza els exercicis següents.

- E01. Volem saber els horàries disponibles de tren. Per a tal fi realitza les següents tasques:
  1. Defineix una interface amb les següents propietats: ident, estacioorige, estaciodesti, horaeixida i horaarribada.
  2. Defineix un vector d'aquesta interface i **ompli aquest a través d'una consulta ajax**.
  3. Crea un select amb els orogens distints. Al seleccionar un valor ompli el següent select amb els distints destins des d'aquest origen.

## Origen - Desti estacio de tren



### 4. Activitats entregables.

No hi han.

### 5. Bibliografia.

- <https://docs.angular.lat/tutorial/>
- <https://docs.angular.lat/tutorial/toh-pt6>
- <https://www.typescriptlang.org/es/>
- <https://angular.io/>
- <https://angular.io/guide/inputs-outputs>



## 6. Aclariments

No intentes copiar exercicis d'altres companys o internet. ni tans sols pegar una miradeta. Si que pots consultar la sintaxis d'una funció, etc. Però no et serveix per a res intentar buscar la solució.

És el major error que pot realitzar qui comença a programar. Sols et deixarà una falsa sensació d'aprenentatge i després no sabràs solucionar els problemes per tu mateix.

Cal intentar que el resultat que genere el vostre codi siga el mes paregut possible a l'exercici (si hi ha imatge d'exemple).

Com els Navegadors no son compatibles entre sí al 100% jo corregire els examens amb FIREFOX. Per tant, es recomanable utilitzar aquest per als nostres exercicis.

Crea uns arxius per a cada exercici o apartat. Així un error a un no afectarà a la resta.

Per a entregar els exercicis crear una carpeta amb el vostre nom i comprimiu aquesta (el nom de l'arxiu comprimit serà elteunom.zip o elteunom.tar.gz). Si entregueu arxius solts o arxius genèrics sense identificar el vostre nom puc confondre'm i no puntuar-vos algun exercici al no saber de qui és.

Cal entregar tots els arxius necessaris per a l'exercici (fotos, llibreries, etc).