Components I

ARICHITTETURA CON I COMPONENTS

Ogni applicazione Angular ha almeno un componente, il componente radice che collega la gerarchia di componenti al DOM, ovvero l'**app.component**

Ma cos'è un componente?

Come dice la parola stessa, il component, è uno dei tanti elementi che può comporre la nostra pagina web e grazie ad Angular può diventare riutilizzabile.

È composto da tre parti principali, ovvero:

- **IL TEMPLATE**: ovvero l'HTML principale
- **CLASS**: ovvero la classe TypeScript che gestire la logica del componente
- **METADATA**: ovvero i decoratori che utilizziamo per dire al component cosa deve fare. I decoratori sono funzioni speciali che vengono utilizzate per estendere o modificare un comportamento di una classe in Angular (es. *@Component*)

DATABINDING

Cosa vuol dire databinding?

Vuol dire trasmissione dei dati; da CHI a CHI

È un meccanismo che consente di collegare e sincronizzare i dati tra il TEMPLATE HTML e la classe TypeScript.

Ci sono due tipi principali di binding in Angular:

- **ONE WAY DATABINDING**(unidirezionale):
  es.
  Nel template HTML
  `<h1>Benvenuto, {{nome}}</h1>`
  Nella classe TypeScript
  `nome: string = 'Sofia';`

  In questo caso, semplicemente, la classe TypeScript manda il valore della variabile nome al template HTML per la stampa, ONE WAY

- **TWO WAY DATABINDING**(bidirezionale):
  es.
  Nel template HTML

```
<input [(ngModel)]='nome'>
<p>Ciao, {{nome}}!</p>
```

Nella classe TypeScript

```
nome: string = 'Elia';
```

In quest'altro caso, se l'utente all'interno dell'input cambia il valore della variabile nome, il Template manda il dato alla classe TypeScript, lui lo memorizza e lo rimanda indietro ed il template lo stampa, TWO WAY DATABINDING

DIRETTIVE BUILD-IN

Cosa sono le directives?

Le directives sono classi che aggiungono un **comportamento** aggiuntivo agli **elementi** nelle Applicazioni Angular.

Sono direttive predefinite fornite dal framework stesso, pronte all'uso e che ci risparmiano di scrivere righe di codice personalizzato.

Due categorie principali:

1. **DIRETTIVE STRUTTURALI**: iniziano con il prefisso ` * ` e consentono di modificare la struttura del DOM manipolando la presenza o l'assenza di elementi.

- ***ngIf** : per decidere se renderizzare o meno un elemento
- ***ngFor** : per iterare un array e generare dinamicamente elementi HTML

2. **DIRETTIVE DI ATTRIBUTO**: vengono applicate ad un elemento HTML e forniscono comportamenti aggiuntivi

- **ngClass** : per aggiungere o rimuovere classi CSS
- **ngStyle** : per applicare stili CSS dinamici ad un elemento
- **ngModel** : Fornisce il binding bidirezionale per gli elementi di input, consentendo di sincronizzare i dati tra il modello e la vista (*vedi es. two-way databinding sopra*)

NAVIGATION

Angular Routing, a cosa serve?

Il routing è essenziale per far sì che la pagina web possa funzionare come una vera SPA(Single Page Application). In pratica il routing consente di navigare tra le diverse sezioni dell'applicazione.

Con il routing ottengo:

- **Navigazione tra le pagine**: Ci consente di navigare tra le pagine senza dover ricaricare di nuovo l'HTML. (*vedi descrizione SPA negli appunti precedenti*)
- **Preservo l'usabilità, grazie al mantenimento dello stato dell'applicazione**: Quindi il tasto per tornare indietro (back) funziona e consente agli utenti di tornare alla pagina precedentemente visitata.
- **Gestione dei parametri dell'URL**: ad esempio, quando un parametro dell'URL indica l'ID di un oggetto specifico da visualizzare nella pagina di dettaglio.

Il routing in Angular viene gestito attraverso il modulo `RouterModule` e la definizione della route all'interno del file di configurazione. Le route specificano le URL associate a ciascuna visualizzazione, insieme al compotente da caricare e visualizzare per quella specifica route.