A diretiva `ngTemplateOutlet` é usada no Angular para **renderizar dinamicamente** um `ng-template` em tempo de execução. Ela é extremamente útil quando você quer **reutilizar blocos de HTML ou conteúdo dinâmico** em diferentes partes da aplicação.

✅ Forma geral de uso

```
<ng-container *ngTemplateOutlet="meuTemplate"></ng-container>

<ng-template #meuTemplate>
  <p>Conteúdo do Template</p>
</ng-template>
```

📌 O que pode ser feito com `ngTemplateOutlet`?

1. Renderização básica de um template

```
<ng-container *ngTemplateOutlet="templateSimples"></ng-container>

<ng-template #templateSimples>
  <p>Renderizado dinamicamente!</p>
</ng-template>
```

---

2. Renderizar múltiplos templates com base em condição

```
<ng-container *ngTemplateOutlet="condicao ? templateA :
templateB"></ng-container>

<ng-template #templateA>
  <p>Este é o Template A</p>
</ng-template>

<ng-template #templateB>
```

```
  <p>Este é o Template B</p>
</ng-template>

condicao = true; // ou false
```

---

3. Passando contexto para o template

Você pode passar variáveis com o contexto para usar dentro do template com `let-`.

HTML:

```
<ng-container *ngTemplateOutlet="templateComContexto; context: { nome: 'Jessica', idade: 30 }"></ng-container>

<ng-template #templateComContexto let-nome let-idade="idade">
  <p>Nome: {{ nome }}</p>
  <p>Idade: {{ idade }}</p>
</ng-template>
```

---

4. Reutilização em componentes filhos

Você pode passar templates para um componente filho e renderizar dentro dele com `ngTemplateOutlet`.

Componente pai:

```
<app-card [template]="meuTemplate"></app-card>

<ng-template #meuTemplate>
  <p>Conteúdo do template passado como input!</p>
</ng-template>
```

Componente filho:

```
<ng-container *ngTemplateOutlet="template"></ng-container>

@Input() template!: TemplateRef<any>;
```

---

5. Usando ngTemplateOutlet com variáveis locais (`let-`) para conteúdo dinâmico

<ng-container *ngTemplateOutlet="templateComVariaveis; context: { $implicit: 'Ana', cargo: 'Engenheira' }"></ng-container>

<ng-template #templateComVariaveis let-nome let-cargo="cargo">
  Nome: {{ nome }}
  Cargo: {{ cargo }}
</ng-template>

- `$implicit`: valor padrão passado para `let-<nome>`.
- `let-cargo="cargo"`: associa a variável `cargo` do contexto ao nome local `cargo`.

---

6. Renderizar template dentro de `ngIf`, `ngFor`, etc.

Com `ngIf`:

<ng-container *ngIf="mostrarTemplate">
  <ng-container *ngTemplateOutlet="templateCondicional"></ng-container>
</ng-container>

<ng-template #templateCondicional>
  Somente visível quando mostrarTemplate for true
</ng-template>

TS:

mostrarTemplate = true;

⚠️ Importante

Se estiver usando `*ngTemplateOutlet` e aparecer erro como:

> NG0303: Can't bind to 'ngTemplateOutlet' since it isn't a known property of 'ng-container'

✅ **Solução:**

- Importe o `CommonModule` no `@Component({ imports: [...] })` se estiver usando **Standalone Components**.

- Ou, no `@NgModule`, importe `CommonModule`.
