# Angular - Services

## 🟦 SERVICES NO ANGULAR

### ✔️ O que é um Service

Um **service** é um arquivo usado para guardar lógica separada dos componentes.

Ele existe para organizar o código e evitar repetição.

---

### ✔️ Por que usar Services

- Para **separar lógica** dos componentes
- Para **reutilizar funções** em vários componentes
- Para **compartilhar dados** entre componentes
- Para manter o componente mais limpo

---

### ✔️ Como criar um Service

Com o comando:

```
ng g s nome-do-service
```

Isso gera:

```
nome-do-service.service.ts
```

---

### ✔️ Estrutura básica de um Service

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

@Injectable({
  providedIn: 'root'
})
```

```
export class MeuService {

  constructor() {}

  mensagem() {
    return "Olá, Ana!";
  }
}
```

---

## ✔️ Como usar um Service no componente

### 1. Importa o service

```
import { MeuService } from '../services/meu.service';
```

### 2. Injeta no construtor

```
constructor(private meuService: MeuService) {}
```

### 3. Usa a função do service

```
msg = this.meuService.mensagem();
```

---

## ✔️ Compartilhar dados entre componentes usando service (simples)

### No service:

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

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

```
  dado = signal("valor inicial");
}
```

### Em qualquer componente que usar:

```
constructor(public compartilhado: CompartilharService) {}
```

### No HTML:

```
<p>{{ compartilhado.dado() }}</p>
```

Qualquer componente que mudar o signal vai atualizar nos outros.

---

Exatamente, Ana — **é isso mesmo**.

E vou te explicar do jeito mais direto possível:

---

# 🟦 MODELO BÁSICO (padrão certinho)

## 1. No service, você coloca a função

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

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

  constructor() {}

  minhaFuncao(a: number, b: number) {
    return a + b;
  }
}
```

---

## 2. No componente, você usa o service

### TS do componente:

```
import { Component } from '@angular/core';
import { UtilService } from '../services/util.service';

@Component({
  selector: 'app-exemplo',
  standalone: true,
  templateUrl: './exemplo.html'
})
export class ExemploComponent {

  resultado: number = 0;

  constructor(private utilService: UtilService) {}

  calcular() {
    this.resultado = this.utilService.minhaFuncao(5, 10);
  }
}
```

---

## 3. No HTML do componente

```
<p>Resultado: {{ resultado }}</p>
<button (click)="calcular()">Calcular</button>
```

---

# 🟦✔️ Situação

Você tem uma classe assim:

calculadora.ts

```
export class Calculadora {
  somar(a: number, b: number) {
    return a + b;
  }

  multiplicar(a: number, b: number) {
    return a * b;
  }
}
```

E quer usar isso dentro de um componente Angular.

---

# 🟦✔️ Passo 1 — Importar essa classe dentro de um Service

calculadora.service.ts

```
import { Injectable } from '@angular/core';
import { Calculadora } from '../utils/calculadora'; // caminho do arquivo

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

  private calc = new Calculadora();

  somar(a: number, b: number) {
    return this.calc.somar(a, b);
  }

  multiplicar(a: number, b: number) {
    return this.calc.multiplicar(a, b);
  }
}
```

✔️ Aqui o service vira uma **fachada** para sua classe.

✔️ Ele cria o objeto da classe e a aplicação inteira usa essa instância.

---

# 🟦✔️ Passo 2 — Usar o service no componente

meu-componente.ts

```
import { Component } from '@angular/core';
import { CalculadoraService } from '../services/calculadora.service';

@Component({
  selector: 'app-meu-componente',
  standalone: true,
  templateUrl: './meu-componente.html'
})
export class MeuComponente {

  resultado = 0;

  constructor(private calcService: CalculadoraService) {}

  testar() {
    this.resultado = this.calcService.somar(5, 10);
  }
}
```

---

# 🟦✔️ Passo 3 — HTML do componente

```
<p>Resultado: {{ resultado }}</p>
<button (click)="testar()">Calcular</button>
```

---