

Criando Pipes Customizados

Vamos supor que, em sua aplicação, você precisa mostrar ao usuário o resultado da soma de vários números que estão em um array. Nós podemos criar um Pipe para fazer essa soma.


Para criar um Pipe, use o comando

```
ng generate pipe array-sum
```

ou

```
ng g p array-sum
```

array-sum é o nome do Pipe. Caso queira dar outro nome, basta alterar este valor. Quando você gerar o Pipe, você terá um arquivo parecido com esse:



```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'arraySum'
})
export class ArraySumPipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }
}
```

Essa é a base de um Pipe. Um Pipe implementa a interface *PipeTransform*, interface que te obriga a implementar o método *transform()*, em que o parâmetro **value** é o valor que vai ser transformado e o parâmetro **args** é um array que recebe qualquer outro valor opcional. Esse parâmetro não é obrigatório e você pode retirá-lo.

Esse método deve retornar algum valor. Por padrão, a implementação coloca o tipo de retorno, tanto do **value** quanto do retorno, como *unknown*, mas nós podemos alterar. Vamos alterar as coisas necessárias nesse método.

```

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'arraySum'
})
export class ArraySumPipe implements PipeTransform {

  transform(value: number[]): number {
    let sum = 0

    for (let number of value) {
      sum += number
    }

    return sum
  }
}

```

Agora, vamos usar nosso Pipe. Veja abaixo um exemplo de como poderíamos usar:

```

import { Component } from '@angular/core';

@Component({
  selector: 'main',
  template: `
    <div>
      <p>{{ numeros | arraySum }}</p>
    </div>
  `
})
export class MainComponent {
  numeros: number[] = [5, 10, 50, 60, 70, 90]
}

```