05

Evitando importar negociações duplicadas

Transcrição

Vimos que ao implementarmos uma interface que possua métodos, seremos obrigados a implementá-los respeitando sua assinatura. Além disso, podemos utilizar o tipo da interface para referenciar objetos que implemente a interface. Falando em método, isto é, comportamento de objetos, não é raro termos que comparar um objeto com outro. Aliás, a implementação desta comparação pode divergir entre diferentes objetos.

Dessa forma, podemos criar a interface `Igualavel` que possua o método `ehIgual`. O método deve receber como parâmetro um objeto de mesmo tipo do objeto envolvido na comparação, retornando `true` ou `fase` para indicar igualdade ou não.

```
// app/ts/models/Igualavel.ts

export interface Igualavel<T> {

    ehIgual(objeto: T): boolean
}
```

Tivermos que lançar mão do uso de generics, pois o método `ehIgual` deve receber o tipo indicado por T.

Não podemos nos esquecer de exportá-la através de `index.ts`:

```
// app/ts/models/index.ts

export * from './Negociacao';
export * from './Negociacoes';
export * from './NegociacaoParcial';
export * from './Imprimivel';
export * from './Igualavel';
```

Agora, vamos fazer com que a classe `Negociacao` implemente a interface. Diferente da herança com `extends`, podemos implementar quantas interfaces quisermos:

```
import { Imprimivel, Igualavel } from './index';

export class Negociacao implements Imprimivel, Igualavel<Negociacao> {

    constructor(readonly data: Date, readonly quantidade: number, readonly valor: number) {}

    get volume() {

        return this.quantidade * this.valor;
    }

    paraTexto(): void {
        console.log('-- paraTexto --');
        console.log(
```

```
            `Data: ${this.data}
            Quantidade: ${this.quantidade},
            Valor: ${this.valor},
            Volume: ${this.volume}`
        )
    }

    ehIgual(negociacao: Negociacao): boolean {

        return this.data.getDate() == negociacao.data.getDate()
            && this.data.getMonth() == negociacao.data.getMonth()
            && this.data.getFullYear() == negociacao.data.getFullYear();
    }
}
```

Agora, vamos usar nosso método `ehIgual` para evitar a importação de negociações duplicada, usam conhecimento fundamental das funções filter, some e forEach:

```
    @throttle()
    importaDados() {

        this._service
            .obterNegociacoes(res => {

                if(res.ok) {
                    return res;
                } else {
                    throw new Error(res.statusText);
                }
            })
            .then(negociacoesParaImportar => {

                const negociacoesJaImportadas = this._negociacoes.paraArray();

                negociacoesParaImportar
                    .filter(negociacao =>
                        !negociacoesJaImportadas.some(jaImportada =>
                            negociacao.ehIgual(jaImportada)))
                    .forEach(negociacao =>
                    this._negociacoes.adiciona(negociacao));

                this._negociacoesView.update(this._negociacoes);
            });
    }
```

Excelente, mas nosso código pode ficar ainda melhor.