01

Criando mais um modelo

Transcrição

Começando deste ponto? Você pode fazer o download (https://s3.amazonaws.com/caelum-online-public/typescript/03-alurabank.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Vamos continuar com nosso projeto. Já somos capazes de criar uma instância de `Negociacao` a partir dos dados do formulário. Agora, precisamos guardar em uma lista todas as negociações incluídas pelo usuário para mais tarde exibirmos todas elas através de uma tabela.

No entanto, só podemos adicionar negociações em nossa lista, não podemos remover ou apagá-la. Um array do JavaScript permite dezenas de operações, sendo assim, vamos encapsular um array dentro da classe `Negociacoes` e fazer com que todo acesso ao array encapsulado passe pelos métodos da classe.

Vamos criar o arquivo `app/ts/models/Negociacoes.ts`. Ele terá uma única propriedade, um array:

```
class Negociacoes {

// erro de compilação, não podemos deixar o tipo any por padrão

    // Member 'negociacoes' implicitly has an 'any[]' type.
    private _negociacoes = [];

}
```

Parece que desagradamos o TypeScript na declaração do nosso `array`. Lembre-se que agora precisamos identificar os tipos das propriedades da classe. A boa notícia é que o TypeScript já possui o tipo `Array`:

```
class Negociacoes {

   // ainda continua com erro!
    private negociacoes: Array = [];

}
```

Pois é, nosso compilador ainda não satisfeito. Mesmo identificando o tipo `Array` ainda temos a seguinte mensagem de erro:

```
[ts] Generic type 'Array<T>' requires 1 type argument(s).
```

O problema é que Array é um tipo genérico. Mas por que genérico? Podemos guardar strings, números e objetos dentro de um array e utilizarmos apenas `Array` indica que temos um array, mas não o tipo dos dados que contém. Dessa maneira, precisamos especificar o tipo de dados armazenados por um array da seguinte maneira:

```
class Negociacoes {

    private _negociacoes: Array<Negociacao> = [];

}
```

Veja que a declaração do array segue o padrão `Array<TipoDosDadosQueContem>` . Agora, o compilador do TypeScript ficará feliz.

Inclusive, podemos declarar um array do tipo `Negociacao` de uma maneira alternativa, basta usarmos o tipo seguido de `[]` :

```
class Negociacoes {

    private _negociacoes: Negociacao[] = [];
}
```

Agora, vamos para o método `adiciona()` , que deve receber como parâmetro um objeto do tipo `Negociacao`

```
class Negociacoes {

    private _negociacoes: Negociacao[] = [];

    adiciona(negociacao: Negociacao) {

        this._negociacoes.push(negociacao);
    }
}
```

Qualquer outro tipo será vetado pelo compilador do TypeScript. Por fim, vamos criar o método `paraArray` que devolve o array encapsulado por `Negociacoes` :

```
class Negociacoes {

    private _negociacoes: Negociacao[] = [];

    adiciona(negociacao: Negociacao) {

        this._negociacoes.push(negociacao);
    }

    paraArray() {

       return this._negociacoes;
    }
}
```

Nosso código compila, deixamos o TypeScript feliz!

Para não corremos o risco de esquecermos, vamos importar o javascript que será resultante da compilação da classe em `index.html`. Vamos importá-lo antes de `app.ts`. Aliás, deixaremos sempre `app.ts` como último.

```
<!-- app/index.html -->
<!-- código anterior omitido -->
    <script src="js/models/Negociacao.js"></script>
    <script src="js/controllers/NegociacaoController.js"></script>
    <script src="js/models/Negociacoes.js"></script>
    <script src="js/app.js"></script>
```

Agora que já temos nosso novo modelo, já podemos integrá-lo com `NegociacaoController`.