01

Repetindo código...oh my!

Transcrição

Começando deste ponto? Você pode fazer o download (https://s3.amazonaws.com/caelum-online-public/typescript/04-alurabank.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Nossa aplicação consegue incluir e exibir negociações. Mas é bem comum uma aplicação informar o usuário sobre o sucesso das operações realizadas. Para isso, vamos criar o arquivo `app/ts/views/MensagemView.ts`. Ele terá uma estrutura praticamente idêntica ao arquivo de view que criamos antes, a diferença estará apenas no seu template, que recebe um parâmetro diferente e retorna um template específico:

```
class MensagemView {

    private _elemento: Element;

    constructor(seletor: string) {

        this._elemento = document.querySelector(seletor);
    }

    update(model: string) {

        this._elemento.innerHTML = this.template(model);
    }

    template(model: string): string {

        return `<p class="alert alert-info">${model}</p>`;
    }
}
```

Como todos os arquivo que criamos, vamos importar o resultado da sua compilação em `app/index.html`:

```
<!-- 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/views/NegociacoesView.js"></script>
<script src="js/views/MensagemView.js"></script>
<script src="js/app.js"></script>
<!-- código posterior omitido -->
```

Agora, vamos seguir um processo parecido como os que fizemos com a `NegociacoesView`, a diferença é que não chamaremos a função `update()` na inicialização, apenas no método `adiciona()`:

```
class NegociacaoController {

    private _inputData: HTMLInputElement;
    private _inputQuantidade: HTMLInputElement;
    private _inputValor: HTMLInputElement;
    private _negociacoes = new Negociacoes();
    private _negociacoesView = new NegociacoesView('#negociacoesView');
    private _mensagemView = new MensagemView('#mensagemView');

    constructor() {
        this._inputData = <HTMLInputElement>document.querySelector('#data');
        this._inputQuantidade = <HTMLInputElement>document.querySelector('#quantidade');
        this._inputValor = <HTMLInputElement>document.querySelector('#valor');
        this._negociacoesView.update(this._negociacoes);
    }

    adiciona(event: Event) {

        event.preventDefault();

        const negociacao = new Negociacao(
            new Date(this._inputData.value.replace(/-/g, ',')),
            parseInt(this._inputQuantidade.value),
            parseFloat(this._inputValor.value)
        );

        this._negociacoes.adiciona(negociacao);
        this._negociacoesView.update(this._negociacoes);
        this._mensagemView.update('Negociação adicionada com sucesso');
    }
}
```

A cada negociação adicionada, a mensagem é exibida. Tudo funciona, mais há algum "cheiro" de código que paira no ar. Assunto do próximo vídeo.