01

Capturando os dados do formulário

Transcrição

Começando deste ponto? Você pode fazer o download (https://s3.amazonaws.com/caelum-online-public/typescript/02-alurabank.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

De nada nos adianta a classe `Negociacao` se o usuário não pode interagir com ela. Vamos criar uma classe que lidará com as ações do usuário, a classe `NegociacaoController`. Ela ficará no namespace `controllers`.

```
// app/ts/controllers/NegociacaoController.ts

    class NegociacaoController {

        private _inputData;
        private _inputQuantidade;
        private _inputValor;

        constructor() {

            this._inputData = document.querySelector('#data');
            this._inputQuantidade = document.querySelector('#quantidade');
            this._inputValor = document.querySelector('#valor');
        }

        adiciona(event) {

            event.preventDefault();

            const negociacao = new Negociacao(
                this._inputData.value,
                this._inputQuantidade.value,
                this._inputValor.value);

            console.log(negociacao);
        }
    }
```

Nossa classe declara três propriedades que guardarão os elementos do DOM de entrada, no caso, os de data, quantidade e valor. No `constructor()` usamos `document.querySelector` para buscar os elementos. Por fim, no método adiciona, criamos uma nova instância de `Negociacao` com base nos dados do formulário.

Não podemos nos esquecer de importar o script gerado pelo processo de compilação:

```
    <script src="js/models/Negociacao.js"></script>
    <script src="js/app.js"></script>
    <script src="js/controllers/NegociacaoController.js"></script>
```

Agora, vamos alterar `app.ts`, para que crie uma instância de `NegociacaoController` para em seguida associar ao evento `submit` do formulário de `index.html` a chamada do método adiciona.

```
// app/ts/app.ts

const controller = new NegociacaoController();
document
    .querySelector('.form')
    .addEventListener('submit',controller.adiciona.bind(controller));
```

Nosso código compila, mas a ordem de carregamento de scripts nos causou problemas. O arquivo `NegociacaoController.js` deve ser carregado antes de `app.ts` e depois `NegociacaoController`, pois este depende de `Negociacao`. O sistema de namespace não ataca o problema da ordem de dependência entre scripts, algo que ainda resolveremos neste treinamento.

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

Isso é suficiente para fazer funcionar nossa aplicação.