09

Aplicando a solução no NegociacaoController

Transcrição

Você pode estar pensando "será que existe uma maneira mais fácil de fazer o que queremos?". A resposta é sim. Conheceremos a forma mais simples, depois. Primeiramente, vamos levar a estratégia criada no `index.html`, para o `NegociacoesController`.

Abaixo dos `input`s do `constructor()`, adicionaremos o seguinte trecho do código:

```
this._listaNegociacoes = new Proxy(new ListaNegociacoes(), {

    get(target, prop, receiver) {

        if(['adiciona', 'esvazia'].includes(prop) && typeof(target[prop]) == typeof(Function)) {

            return function(){

              console.log(`método '${prop}' interceptado`);

             Reflect.apply(target[prop], target, arguments);

              this._negociacoesView.update(target);

            }
     }

     return Reflect.get(target, prop, receiver);
  }
});
```

Passamos o `model` como parâmetro do `update()`, ao adicionarmos o `target`. No entanto, o nosso código terá problema porque o `this` não será o `controller`, nem se usássemos uma *arrow function*. Por isso, vamos inserir uma variável chamada `self`. que receberá `this`.

```
let self = this;

this._listaNegociacoes = new Proxy(new ListaNegociacoes(), {

    get(target, prop, receiver) {

        if(['adiciona', 'esvazia'].includes(prop) && typeof(target[prop]) == typeof(Function)) {

            return function(){

              console.log(`método '${prop}' interceptado`);

             Reflect.apply(target[prop], target, arguments);

              self._negociacoesView.update(target);
```

```
                }
            }

            return Reflect.get(target, prop, receiver);
        }
    });
```

Faremos um teste a seguir, mas faltou um detalhe: só podemos chamar a atualização da View, depois de aplicar o valor depois do `target`. Vamos testar no navegador, e veremos que o método foi interceptado.

Então, o código está funcionando. Nós tivemos que aprender a lidar com as particularidades do Proxy, e assim, não "sujamos" o nosso modelo. Mais adiante veremos como esconder a complexidade de criação do Proxy. O nosso código ficou muito verboso. Inclusive, não vamos inserir um Proxy para `MensagemView`, mas seria um erro repetir todo este código novamente.

Por enquanto, só brincaremos com o `listaNegociacoes`. A seguir, vamos aplicar um outro padrão de projeto que nos ajudará bastante na criação de um Proxy. Mas a forma de se trabalhar internamente já foi apresentada.