02

Nosso proxy ainda não está 100%!

Transcrição

O que está acontecendo? O `ProxyFactory` está preparado para interceptar métodos, mas não está preparado para interceptar propriedades, como em `Mensagem.js`:

```
class Mensagem {

    constructor(texto='') {

        this._texto = texto;
    }

    get texto() {

      return this._texto;
    }

    set texto(texto) {

        this._texto = texto;
    }
}
```

O `get texto()` é um getter. O `ProxyFactory` não entende que precisa interceptar com o `get`. Já que os getters e setters são acessados como propriedades, temos também que colocar no `ProxyFactory.js`, um código para lidarmos com as propriedades. Para isto, adicionaremos um `set`, logo abaixo do último `return`:

```
set(target, prop, value, receiver) {

    Reflect.set(target, prop, value, receiver);
    acao(target);
}
```

Depois, usaremos o `Reflect.set()`, recebendo os quatro parâmetros. Logo após, chamaremos o `acao(target)`. Assim, garantiremos que quando for executada a propriedade, depois, será a vez do interceptador.

Mas se preenchermos o formulário no navegador, veremos que ele não ficará limpo, e aparecerá uma mensagem de erro que dará `'set' on proxy`. Isto ocorreu porque quando chamamos um `Reflect.set()` temos que chamar também um `return`. Também será necessário adicionar o `if`:

```
set(target, prop, value, receiver) {
    if(props.includes(prop)) {
        target[prop] = value;
        acao(target);
    }
```

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

Aplicamos os filtros nas propriedades que queremos. Podemos melhorar o nosso código... Começando pelo fato que não queremos retornar fixo o `ListaNegociacoes()` - nós queremos levar em consideração o objeto que estamos usando como parâmetro. Faremos um segundo ajuste: o código do `typeof()` quer identificar se estamos trabalhando com uma função. Para ficar mais específico, criaremos o método `static _ehFuncao()`, em seguida, substituiremos o `typeof` por ele:

```
    static _ehFuncao(func) {

        return typeof(func) == typeof(Function);

    }
```

Depois, substituiremos o `typeof` localizado no `if` do `get`, por `_ehFuncao`.

```
    class ProxyFactory{

        static create(objeto, props, acao) {

            return new Proxy(objeto, {

                get(target, prop, receiver) {

                    if(props.includes(prop) && ProxyFactory._ehFuncao(target[prop])) {

                        return function() {

                            console.log(`a propriedade "${prop}" foi interceptada`);
                            Reflect.apply(target[prop], target, arguments);
                            return acao(target);
                        };
                    }
                    return Reflect.get(target, prop, receiver);
                },

                set(target, prop, value, receiver) {
                    if(props.includes(prop)) {
                        target[prop] = value;
                        acao(target);
                    }
                    return Reflect.set(target, prop, value, receiver);
                }
          });
        }
    }
```

Atenção: como estamos trabalhando com um método estático, não se esqueça de adicionar o nome da classe `ProxyFactory` antes do `_ehFuncao()`, no `if`.

Vamos recarregar a página no navegador e preencher os campos do formulário. Tudo estará funcionando no nosso sistema.