04

Organizando Código JavaScript com Classe

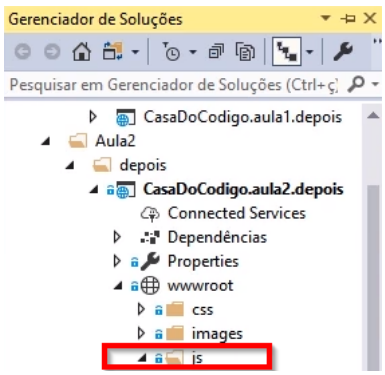
Transcrição

Conseguimos atualizar as quantidades no banco de dados. Voltaremos ao código JavaScript para organiza-lo e precisaremos criar uma nova função, para que possamos tornar o botão "-" também funcional na página do carrinho de compras.

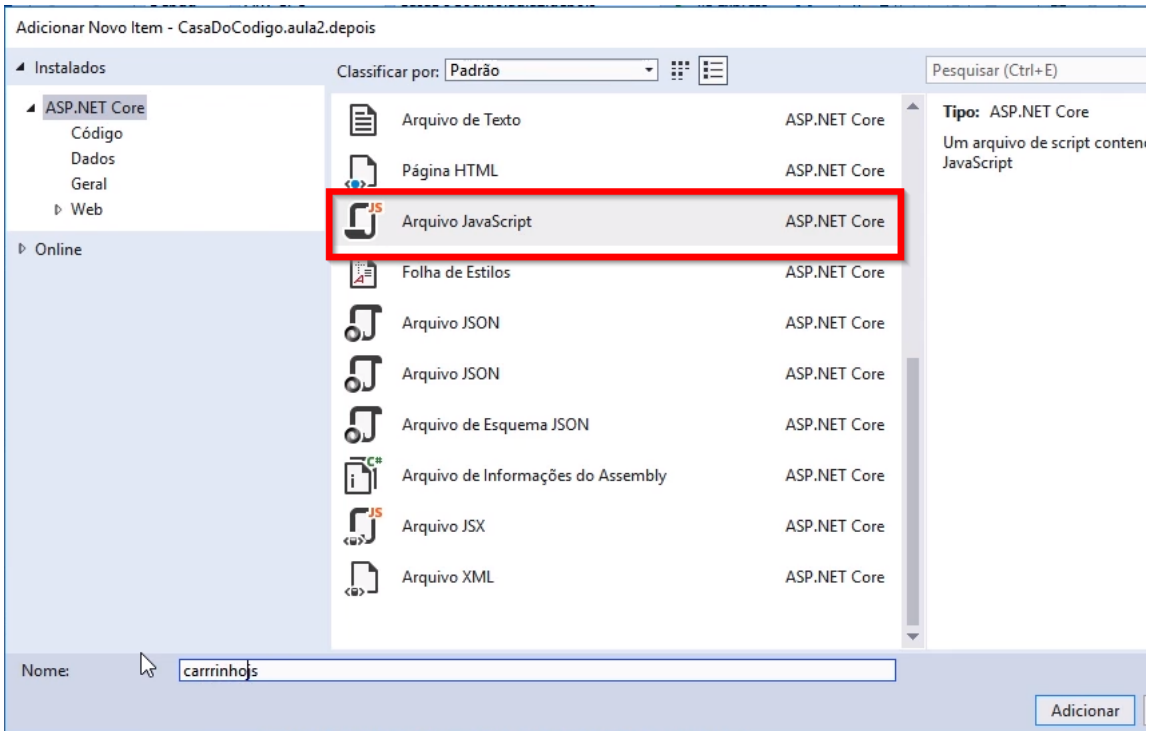
O nosso código JavaScript está se tornando muito extenso, o que prejudica sua organização e manutenção.

```
@{
    ViewData["Title"] = "Carrinho";
}


@section Scripts
{
    <script type="text/javascript">
        function clickIncremento(btn) {
            var linhaDoItem = $(btn).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            var data = {
                Id: itemId,
                Quantidade: novaQtde
            };

            $.ajax({
                url: '/pedido/updatequantidade',
                type: 'POST',
                contentType: 'application/json',
                data: JSON.stringify(data)
            });

            debugger;
        }
    </script>
}
```

Criaremos um novo arquivo que armazenará essas informações. Na área "Gerenciador de Soluções" do Visual Studio, abriremos a pasta "wwwroot > js".



Com `Js` selecionado, clicaremos com o botão direito e escolheremos as opções "Adicionar > Novo Item". Na nova caixa de diálogo, configuraremos para que este novo arquivo seja JavaScript, e seu nome será `carrinho.js` .



Moveremos para dentro do novo arquivo o código JavaScript, que contém somente a função `clickIncremento()` . Como nós movemos esse código, precisamos avisar a view onde está localizado o arquivo js, portanto devemos passar a fonte do arquivo (`src`). Teremos a subpasta `js` e o arquivo `carrinho.js` .

```
@section Scripts
{
    <script src="~/js/carrinho.js">

    </script>
}
```

Voltando para `carrinho.js` , teremos uma função `clickIncremento()` que chamos de **escopo global**, o que indica que esta função está visível para todo o código JavaScript que referencia esse arquivo. O que desejamos fazer é organizar esse código, deixando a função dentro de um outro escopo, logo, ciaremos uma nova classe para encapsular a função.

```
function clickIncremento(btn) {
            var linhaDoItem = $(btn).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            var data = {
                Id: itemId,
                Quantidade: novaQtde
            };

            $.ajax({
                url: '/pedido/updatequantidade',
                type: 'POST',
                contentType: 'application/json',
                data: JSON.stringify(data)
```

```
            });

            debugger;
```

A classe é um conceito que entrou no JavaScript a partir do ECMAScript versão 6, e ele irá permitir que organizemos melhor nosso código. o Nome dessa classe será `Carrinho` .Para utilizar um função dentro de uma classe, não podemos ter a palavra `function` dentro dela, portanto a deletaremos.

```
class Carrinho {
    clickIncremento(btn) {
            var linhaDoItem = $(btn).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            var data = {
                Id: itemId,
                Quantidade: novaQtde
            };

            $.ajax({
                url: '/pedido/updatequantidade',
                type: 'POST',
                contentType: 'application/json',
                data: JSON.stringify(data)
            });

            debugger;

        }

}
```

Para acessarmos a função `clickIncremento()` que está na classe precisamos criar uma instância, que ficará ao final do código e se chamará `carrinho` . `carrinho` será uma nova instância da classe `Carrinho()` .

```
class Carrinho {
    clickIncremento(btn) {
            var linhaDoItem = $(btn).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            var data = {
                Id: itemId,
                Quantidade: novaQtde
            };

            $.ajax({
                url: '/pedido/updatequantidade',
                type: 'POST',
                contentType: 'application/json',
                data: JSON.stringify(data)
            });

            debugger;
```

```
        }

}

var carrinho = new Carrinho();
```

Criaremos uma nova função para decrementar a quantidade de itens no carrinho. Essa nova função se chamará `clickDecremento()`, e receberá o botão como parâmetro (`btn`).

```
class Carrinho {
    clickIncremento(btn) {
            var linhaDoItem = $(btn).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            var data = {
                Id: itemId,
                Quantidade: novaQtde
            };

            $.ajax({
                url: '/pedido/updatequantidade',
                type: 'POST',
                contentType: 'application/json',
                data: JSON.stringify(data)
            });

            debugger;

        }

        clickDecremento(btn) {

        }
}

var carrinho = new Carrinho();
```

O código de `clickDecremento()` será praticamente o mesmo de `clickIncremento()`, portanto iremos realizar um reaproveitamento do código atribuindo-lhe novas funções. Observem este trecho:

```
clickIncremento(btn) {
            var linhaDoItem = $(btn).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            var data = {
                Id: itemId,
                Quantidade: novaQtde
            };
```

Sabemos que esta parte de código é responsável pela obtenção do objeto `data`. Podemos criar uma nova função denominada `getData()` ao final da classe `Carrinho`. Essa nova função deverá receber um elemento html, portanto escreveremos `elemento` como parâmetro. Feito isso, inseriremos o trecho de código que havíamos destacado para dentro de `getData()`.

```
class Carrinho {
    clickIncremento(btn) {
            var linhaDoItem = $(btn).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            var data = {
                Id: itemId,
                Quantidade: novaQtde
            };

            $.ajax({
                url: '/pedido/updatequantidade',
                type: 'POST',
                contentType: 'application/json',
                data: JSON.stringify(data)
            });

            debugger;

        }

        clickDecremento(btn) {

        }
        getData(elemento) {
        var linhaDoItem = $(btn).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            var data = {
                Id: itemId,
                Quantidade: novaQtde
            };
        }
}

var carrinho = new Carrinho();
```

No método `getData()` substituiremos o parâmetro `btn` em `linhaDoItem()` para `elemento`.

```
getData(elemento) {
        var linhaDoItem = $(elemento).parents('[item-id]');
            var itemId = $(linhaDoItem).attr('item-id');
            var novaQtde = $(linhaDoItem).find('input').val();

            var data = {
                Id: itemId,
                Quantidade: novaQtde
            };
```

```
        }
}
```

Iremos consumir a função `GetData()` em `clickIncremento()`, para isso acessaremos a instância `this.getData()` e passaremos o parâmetro `btn`. Depois, atribuiremos esse conteúdo a uma variável local denominada `let data`.

```
class Carrinho {
    clickIncremento(btn) {
        let data = this.getData(btn);

        $.ajax({
            url: '/pedido/updatequantidade',
            type: 'POST',
            contentType: 'application/json',
            data: JSON.stringify(data)
        });
```

Vejamos o próximo trecho de código restante:

```
    $.ajax({
      url: '/pedido/updatequantidade',
      type: 'POST',
      contentType: 'application/json',
      data: JSON.stringify(data)
    });
```

Sabemos que sua função é postar a quantidade do pedido, logo, criaremos ao final da classe `Carrinho`, uma nova função denominada `postQuantidade()` que receberá o objeto `data`. Dentro desta função alocaremos o trecho do código destacado.

```
postQuantidade(data) {
        $.ajax({
      url: '/pedido/updatequantidade',
      type: 'POST',
      contentType: 'application/json',
      data: JSON.stringify(data)
    });
```

Assim feito, chamaremos esse código por meio do `this.postQuantidade()`, passando como parâmetro a variável `data`

```
class Carrinho {
    clickIncremento(btn) {
        let data = this.getData(btn);

        this.postQuantidade(data);

        debugger;
    }
```

Ao observarmos atentamente a função `getData()`, veremos que ela não retorna nada. Substituiremos `var data` por `return`

```
    getData(elemento) {
            var linhaDoItem = $(elemento).parents('[item-id]');
                var itemId = $(linhaDoItem).attr('item-id');
                var novaQtde = $(linhaDoItem).find('input').val();

                return  {
                    Id: itemId,
                    Quantidade: novaQtde
                };
            }
    }
```

Nosso código está bem mais organizado e funcional. Copiaremos a código de `clickIncremento()` para `clickDecremento()` e removeremos o `debugger`.

```
class Carrinho {
    clickIncremento(btn) {
        let data = this.getData(btn);
        this.postQuantidade(data);

    }

    clickDecremento(btn) {
        let data = this.getData(btn);
        this.postQuantidade(data);

}
```

Precisamos diferenciar os dois: no caso de `clickIncremento()` o valor de `Quantidade` deve aumentar, já em `clickDecremento()` deve diminuir. Faremos isso adicionando, respectivamente `data.Quantidade++` e `data.Quantidade--`.

```
class Carrinho {
    clickIncremento(btn) {
        let data = this.getData(btn);
        data.Quantidade++;
        this.postQuantidade(data);

    }

    clickDecremento(btn) {
        let data = this.getData(btn);
        data.Quantidade--;
        this.postQuantidade(data);

}
```

Para conseguirmos utilizar a instância da classe `Carrinho`, que está na variável `carrinho` devemos fazer algumas modificações no código html. Iremos até `Carrinho.cshtml` e modificaremos o evento `onclick`, referenciando a instância `carrinho`.

```
<input type="text" value="@(item.Quantidade)"
                                class="form-control text-center" />
```

```
                        <span class="input-group-btn">
                            <button class="btn btn-default"
                                    onclick="carrinho.clickIncremento(this)">
                                <span class="glyphicon-plus"></span>
```

Copiaremos a linha `onclick="carrinho.clickIncremento(this)"` e a colocaremos mais acima do código onde se localiza o botão "-" e copiaremos esse tracho. Depois, alteraremos o nome da função `clickIncremento()` para `clickDecremento()`.

```
<span class="input-group-btn">
                        <button class="btn btn-default">
                                                onclick="carrinho.clickDecremento(this)'
                            <span class="glyphicon-minus"></span>
                        </button>
                    </span>
```

Com isso, temos o código para aumentar e diminuir a quantidade de itens no carrinho de compras.