Primeiro temos que preencher o grid com os livros do usuário, vindo do banco. Como fazer isso? Criamos um método PreencheGridLivrosUsuario para tal função. E quando ele deve ser chamado? Pela lógica, ele tem que ser chamado assim que o usuário loga, ou seja, no page load da página, e também deve ser chamado toda vez que o grid for alterado (seja por inserção, exclusão, alteração etc...

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

this.PreenchePerfilUsuario();

this.PreencheGridLivrosUsuario();

}

if (HttpContext.Current.Session["selecao\_aba"] != null)

{

this.ajustar\_tabs();

}

}

Esse !IsPostBack é pra que evite de toda hora o método ser chamado. PostBack é quando vc clica em algum botão que dispara um evento no servidor. Toda vez que um PostBack ocorre ele recarrega a página, portando chama o PageLoad de novo. Então esse if basicamente quer dizer “Se for a primeira vez que a página foi carregada”, assim, botões como o de alterar perfil do usuário não vão ficar chamando nosso método de preenchimento do grid toda hora. Ok, o que o método faz?

Primeiro tiramos aquele grid de exemplo, pois vamos gerar um dinamicamente. Para isso, no lugar dele colamos um controle asp chamado Literal. Não existe mistério, o que você colocar como text dele, vai ser renderizado como html.

<asp:Literal runat="server" ID="literal\_grid\_livros"></asp:Literal>

Agora no nosso método de preenchimento,

StringBuilder loHTMLGridLivros = new StringBuilder();

List<Entity.Livro> loListaLivros = new Business.Livro().CarregarLivros(loUsuarioLogado);

StringBuilder é uma classe com mais funções de construção de strings, nela vamos colocar todo o html do nosso grid. Carregamos a lista de livros do usuário.

if (loListaLivros.Count == 0)

{

this.literal\_grid\_livros.Text = "<h2>Você não possui nenhum livro cadastrado</h2>";

}

Veja que se não tiver livros, o nosso literal terá um html h2 com uma mensagem de não ter livros. E pronto, ele vai renderizar aquilo normalmente. Mas se não tiver sozinho? Aew é preencher na mão grande usando Append. Não tem como explicar, só olhando como foi feito mesmo e lembrando que HTML não precisa de quebra de linha.

A única novidade aqui é o pulo do gato pra pegar o id. Se eu renderizar meus livros bonitinhos, blz, mas veja que cada livro tem seus botões (alterar, excluir...) e quando o cara clicar neles eu tenho q saber o id daquele livro em especifico, do contrário o banco não tem como advinhar qual livro vc quer excluir, alterar etc... Então qual é o macete:

loHTMLGridLivros.Append("<td><a title=\"Editar este livro\" onclick=\"selecionar\_livro('");

loHTMLGridLivros.Append(loLivro.LVR\_id.ToString());

loHTMLGridLivros.Append("', 'EDITAR', '");

loHTMLGridLivros.Append(loLivro.LVR\_nome);

loHTMLGridLivros.Append("', '");

loHTMLGridLivros.Append(loLivro.LVR\_disponivel);

loHTMLGridLivros.Append("');\"");

No botão editar esse livro, eu digo que no Onclick dele chama uma função de javascript chamada selecionar livro passando como argumentos o id do livro clicado, ‘EDITAR’, o nome do livro e a quantidade disponível. Vamos pra função do javascript que vc vai entender cada um desses parâmetros:

<script type="text/javascript">

function selecionar\_livro(id, acao, nome, quantidade) {

$.ajax({

type: "POST",

url: window.location.href + "/selecao\_livro",

data: "{codigo: '" + id + "'}",

async: false,

contentType: "application/json; charset=utf-8",

dataType: "json",

success: function () {

if (acao == 'EDITAR') {

$('#<%= txt\_editar\_livro\_nome.ClientID %>').val(nome);

$('#<%= txt\_editar\_livro\_qtd.ClientID %>').val(quantidade);

$('#edit').modal('show');

}

else if (acao == 'EXCLUIR') {

$('#delete').modal('show');

}

else if (acao == 'EMPRESTAR') {

$('#<%= txt\_label\_livro.ClientID %>').val(nome);

$('#modal\_novo\_emprestimo').modal('show');

}

}

});

}

</script>

Primeiro: Esse script salva o id do livro numa sessão. Você deve ver que no code behind existe uma método chamado:

[WebMethod]

public static void selecao\_livro(string codigo)

{

HttpContext.Current.Session["selecao\_livro"] = codigo;

}

Então olha o que acontece. Quando vc clica no botão editar de um livro, ele chama o js com o id dele, o js chama o WebMethod e com isso o id do livro é salvo na sessão HttpContext.Current.Session["selecao\_livro"]. Assim, quando você for chamar o método de editar livro, você sabe onde está o id dele (lembrando que o botão de editar no grid só faz abrir o popup, dentro desse popup, quando vc clicar em Editar mesmo é que ele vai chamar o método de editar livro do code behnd)

Uma vez que ele salva o id do livro, veja o que ele faz

success: function () {

if (acao == 'EDITAR') {

$('#<%= txt\_editar\_livro\_nome.ClientID %>').val(nome);

$('#<%= txt\_editar\_livro\_qtd.ClientID %>').val(quantidade);

$('#edit').modal('show');

}

Se o parâmetro da acao for EDITAR ele seta o txt de nome do livro e quantidade como os passados por parâmetro, pra que quando o popup abrir já vir com esses dois campos preenchidos (imagine se vc tivesse 100 livros, clicasse no 58 e abrisse um popup de edição vazio, vc iria esquecer que livro foi que clicou). E por fim ele abre o modal de edição $('#edit').modal('show');

Onde ‘edit’ é o id do modal de edição de livro.

E no excluir? Não precisa do nome nem da quantidade. Por isso, na hora de renderizar o código

loHTMLGridLivros.Append("<a title=\"Excluir este livro\" onclick=\"selecionar\_livro('");

loHTMLGridLivros.Append(loLivro.LVR\_id.ToString());

loHTMLGridLivros.Append("', 'EXCLUIR', ' ', ' ');\"");

Veja que ele passa vazio nesses parâmetros, afinal não vai ser usado, no caso do excluir o js só abre o popup

else if (acao == 'EXCLUIR') {

$('#delete').modal('show');

}

No emprestar ele só precisa do nome do livro e se tiver livros disponíveis pra ele emprestar

if (loLivro.LVR\_disponivel != loLivro.LVR\_emprestado)

{

loHTMLGridLivros.Append("<span class=\"glyphicon glyphicon-trash\"></span></a>");

loHTMLGridLivros.Append("<a title=\"Empreste este livro\" onclick=\"selecionar\_livro('");

loHTMLGridLivros.Append(loLivro.LVR\_id.ToString());

loHTMLGridLivros.Append("', 'EMPRESTAR', '");

loHTMLGridLivros.Append(loLivro.LVR\_nome);

loHTMLGridLivros.Append("', ' ');\"");

No js

else if (acao == 'EMPRESTAR') {

$('#<%= txt\_label\_livro.ClientID %>').val(nome);

$('#modal\_novo\_emprestimo').modal('show');

}

No final é só converter o stringBuilder pra string e mandar ele pro text do nosso literal

this.literal\_grid\_livros.Text = loHTMLGridLivros.ToString();

Então em todo método que lida com um livro em especifico e precisa do id dele, como Atualizar livro, pra pegar o id é só por

if (HttpContext.Current.Session["selecao\_livro"] != null)

loAlterarLivro.LVR\_id = Convert.ToInt32(HttpContext.Current.Session["selecao\_livro"].ToString());

Pronto, ele já sabe qual livro está tratando.

Agora explicando as mensagens. Toda mensagem de erro, aviso, warning é um elemento do bootstrap, por exemplo

<div runat="server" id="div\_mensagem\_livro" visible="false" class="alert alert-info alert-dismissible" role="alert">

<button type="button" class="close" data-dismiss="alert"><span aria-hidden="true">&times;</span><span class="sr-only">Close</span></button>

<span class="glyphicon glyphicon-exclamation-sign" aria-hidden="true"></span>

<span class="sr-only"></span>

<asp:Label runat="server" ID="lbl\_mensagem\_livro" Text=""> </asp:Label>

</div>

Lembrando que a classe alert-info pode mudar pra alert-warning ou alert-sucess, é só chegar no site do Bootstrap. O que é essa div? Um alert, por padrão de visible=false porque não é pra ela aparecer sempre, um botãozinho de fechar e um label pra vc por sua msg.

Essa div pode ficar em qualquer lugar, então pensem antes de dar ctrl c ctrlv. Por exemplo, quando vc altera um livro, abre o popup de livro. Se der merda, a mensagem tem que estar DENTRO do popup. Mas se foi alterado com sucesso, preciso de OUTRA div de alert que estará FORA do popup. Via código que eu vou decidir qual msg de cada div eu vou exibir ou não

Por exemplo, se eu tenho um alert fora do popup, logo acima do grid por exemplo, de id=’msg\_sucesso’, e outra dentro do popup de alteração com o id=’msg\_erro\_alteracao’. No código de alteração basta seguir a logica:

If(deu merda)

This.msg\_erro\_alteracao.visible = true;

This.label\_msg\_erro\_alteracao.Text = “erro”;

Else

This.msg\_sucesso.visible = true;

This.lbl\_msg\_sucesso.Text = “blab la bla”;

E pelo amor de Deus, corrijam os absurdos de conversão. Outra coisa, não se pode simplesmente dar um convert to int 32 para o que o usuário digitou. Porque se o cara digitar ‘huebr’ no campo quantidade, ele vai ter tentar converter pra int e vai dar merda. Infelizmente esse tipo de verificação terá que ficar no code behind e não no business. Para saber a forma correta de tentar converter, veja o método protected void detalhar\_livro(object sender, EventArgs e)

Morkan escreveu la como fazer essa verificação