Renderização de página

Abstract

Este documento tem por finalidade esboçar o funcionamento do novo processo de renderização de paginas blazor server em conjunto com a API.

Basicamente, a renderização de uma pagina no blazor server-side possui um ciclo de vida diferente do habitual. Uma vez que uma pagina possui elementos gráficos na tela, e precisa do retorno da API para alimenta-los, é necessário um controlador externo para aguardar esta requisição e forçar uma renderização da pagina. Desta forma, foi implementado um Singleton de renderização que atua em consonância com a API - ERP - e seus components.

Estrutura

A estrutura criada para resolver esta situação é a seguinte:

Classe Singleton : RenderState

C# Copy Capt

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace SupersoftERP.Shared
{
    public class RenderState
    {
        public List<string> ConcurrentRequisition { get; set; } = new List<string>()


        public void SetConcurrentRequisition(string KeyComponent)
        {
            ConcurrentRequisition.Add(KeyComponent);
        }

        public void FinishConcurrentRequisition(string KeyComponent)
        {
            ConcurrentRequisition.Remove(KeyComponent);
            ConcurrentRequisition.Remove(null);
        }
    }
}
```

BaseWithLogin.cs

```
namespace SupersoftERP.Shared
{
    public partial class BaseWithLogin : ComponentBase
    {
        [Inject]
        protected Blazored.LocalStorage.ILocalStorageService BlazoredLocalStorage { get; se

        [Inject]
        protected RenderState _renderState { get; set; }

        public string LoginInfo { get; set; } = null;

        public async Task<string> GetLoginInfo()
        {
            LoginInfo = await BlazoredLocalStorage.GetItemAsync<string>("LoginInfo");
            return LoginInfo;
        }

        protected override async Task OnInitializedAsync()
        {
            await base.OnInitializedAsync();
            _renderState.SetConcurrentRequisition(this.ToString());
        }

        protected override async Task OnAfterRenderAsync(bool firstRender)
        {
            if (firstRender)
            {
                await base.OnAfterRenderAsync(firstRender);
                var resposta = await GetLoginInfo();

                await Task.Yield();
                await FirstRenderTask().ContinueWith(async t =>
                {
                    await Task.Delay(1);
                    _renderState.FinishConcurrentRequisition(this.ToString());
                    await InvokeAsync(StateHasChanged);
                });
            }
        }

        protected virtual async Task<bool> FirstRenderTask()
        {
            return true;
        }
    }
}
```

LayoutWithLogin

```
#region Render Screen

public void StopTimerRender()
{
    TimerRender.Stop();
}

public void StartTimerRender()
{
    TimerRender.Start();
}

public void BeginTimerRender()
{
    TimerRender.Elapsed += CountDownTimerRender;
    TimerRender.Enabled = true;
}

public void CountDownTimerRender(Object source, ElapsedEventArgs e)
{
    StopTimerRender();
    if (_renderState.ConcurrentRequisition.Count == 0)
    {
        FinishRender();
    }
    else
    {
        StartTimerRender();
    }
}

public void FinishRender()
{
    loadScreen = false;
    InvokeAsync(StateHasChanged);
}

public void LocationChanged(object source, LocationChangedEventArgs e)
{
    loadScreen = true;
    StateHasChanged();
    BeginTimerRender();
}

#endregion
```

CONTEUDO

Basicamente, esta estrutura possibilita através da herança a renderização de uma pagina e de seus componentes.

Em resumo, todas as classes que herdam a estrutura disposta acima, no início do carregamento elas adicionam a fila do RenderState a sua inicialização. Após finalizar o carregamento de suas informações, a estrutura avisa o RenderState que finalizou e é removido da fila. Desta forma é criado uma cadeia de controle que verifica todas as paginas e componentes que estão em carregamento, e quando toda a fila é liberada a tela de load é finalizada, e a pagina é completamente renderizada.

Para essa estrutura funcionar, a pagina ou componente que precissa de carregamento de informações e renderização precisa ter a herança da classe BaseWithLogin, Ex:

```
@page "/addProduto"
@inherits BaseWithLogin
```

Após isto, na inicialização desta pagina, ela ira entrar na fila.

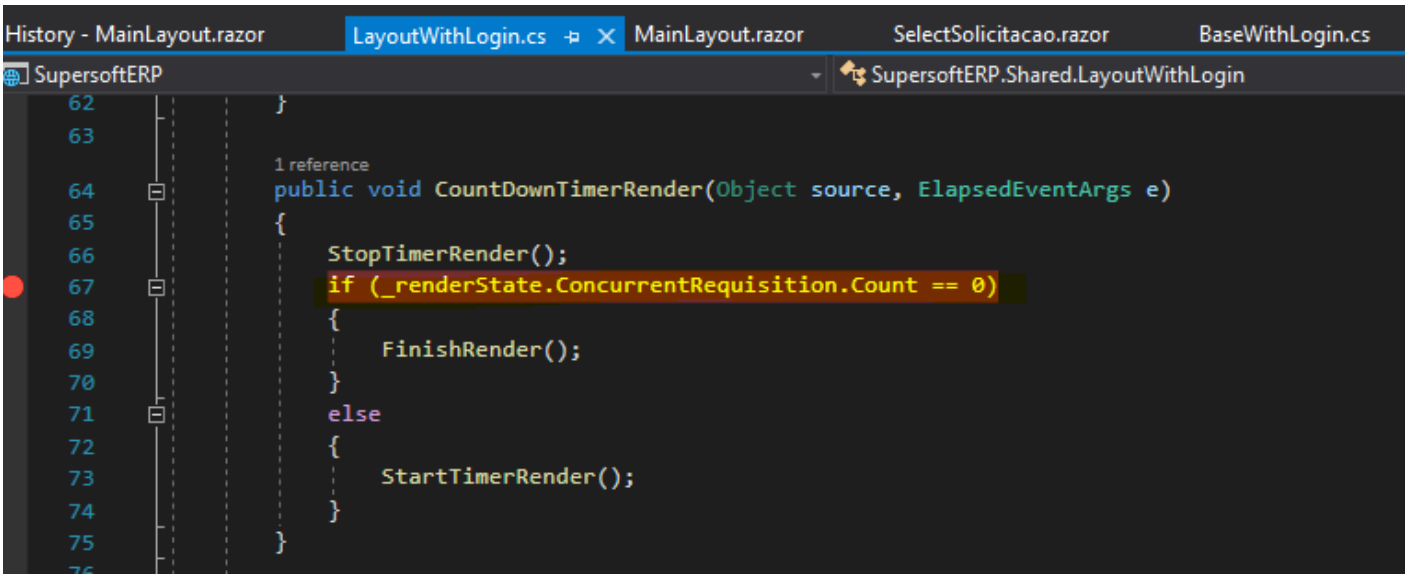
Para que isto funcione corretamente, todas informações de carregamento inicial devem estar concentrados na função override FirstRenderTask, tornando dispensável o uso da função OnAfterRender, Ex:

```
protected override async Task<bool> FirstRenderTask()
{
    produtoObj = await _produtoService.ObterProdutoPorId(Convert.ToInt32(CurrentId), LoginInfo);

    if (produtoObj is not null && produtoObj.FornecedorProdutos is not null && produtoObj.FornecedorProdutos.Count > 0)
        fornobj = produtoObj.FornecedorProdutos[0].Fornecedor;
    return true;
}
```

Desta forma, quando todas operações asyncronas do FirstRenderTask finalizarem, a tela será retirada da fila, e a pagina será carregada.

Existem algumas paginas que podem apresentar erros ainda, não finalizando o load. Uma forma de identifcar este erro é através de um breakpoint neste ponto do código:



Dentro da pagina LayoutWithLogin, o temporizador dele ira disparar este evento a cada 100 milisegundos. Desta forma, quando o breakpoint parar neste ponto, será possível acessar dentro da lista "_renderState.ConcurrentRequisition" qual pagina ou componente não carregou, sendo necessário um tratamento paliativo na pagina.

ERROS

Dentre a lista de erros encontradas, são o seguintes:

Classe principal não estanciada:

É necessário que a classe principal, esteja estanciada como no exemplo abaixo. Caso não esteja, a rotina de renderização ocorrerá com processos inversos.

```
#region Declaration

public bool LimiteImagens = false;
Produto produtoObj = new Produto();
private ProdutoImage ProdutoImage { get; set; } = new ProdutoImage();

#endregion
```

ARQUIVO TUTORIAL

É possível dentro do sistema, na pasta tutorial encontrar na pagina "RenderStatemente" um exemplo de como deve funcionar a criação de uma nova pagina.

Revision #2
Created 17 June 2021 11:20:46
Updated 17 June 2021 14:01:32