webAula

# Framework para criação de páginas ASP

# Revisões

| Autor | Data |
|---|---|
| Matheus Villela Neder Issa | 21/03/2014 |



WWW.WEBAULA.COM.BR

CNPJ: 06.954.022/0001-77

© copyright - webAula

# Sumário






CNPJ: 06.954.022/0001-77
© copyright - webAula

# Introdução

Tendo em vista dificuldades encontradas em relação à reutilização de fragmentos de páginas ASP, os includes, a proposta deste FRAMEWORK é o de contornar essas dificuldades por meio de um conjunto de ferramentas que serão apresentadas neste documento.

Um das dificuldades encontradas para reutilização dos fragmentos de códigos contidos nos includes diz respeito à dependência entre o arquivo de include, que contém um fragmento de código e o arquivo que o inclui. As dependências tratadas são:

1. Variáveis e constantes globais
2. Folhas de estilo CSS
3. Arquivos de JavaScript

O problema das variáveis e constantes globais consiste em que uma vez que um fragmento de código depende de uma dada variável que tenha sido declarada e inicializada em outro arquivo, para reutilizar o fragmento de código contido no arquivo de include, será necessário declarar e inicializar e os elementos no novo arquivo, o que de certo modo inviabiliza a reutilização, uma vez que um determinado nome de variável possa estar sendo utilizado para outro propósito no arquivo onde será necessário incluir a funcionalidade proporcionada pelo trecho de código contido no arquivo de include. Outro ponto em questão é que se um elemento diz respeito a um determinado fragmento de código, o ideal é que ele seja colocado junto a este fragmento, para facilitar tanto o aspecto da reutilização quando de sua manutenção.

Extinguir a utilização das variáveis globais é simples: Tirar do escopo global todo o código referente a um include e leva-lo para uma classe que represente aquela funcionalidade. Esta classe por sua vez, deverá implementar uma interface que será conhecida pelo FRAMEWORK.

O FRAMEWORK oferece também uma ferramenta para declarar as dependências de folhas de estilo CSS e JavaScript. Dessa forma, um código que irá incorporar um include, conhecerá também quais são os arquivos CSS e JavaScript necessários para o correto funcionamento do recurso fornecido pelo include e poderá fazer as declarações referentes ao carregamento desses elementos no espaço adequado do documento HTML.

A seguir segue o manual de como utilizar os recursos do FRAMEWORK. A sessão Includes mostra, em forma um exemplo prático, como criar o arquivo de include e a sessão Utilizando os Includes mostra como deverá ser a estrutura do arquivo ASP principal.



CNPJ: 06.954.022/0001-77

© copyright - webAula

# Includes

Tomando como exemplo um arquivo de include de nome incExample.asp, o mesmo deverá conter uma classe de nome `clsIncExample`. O nome da classe deverá ser sempre constituído pelo nome do arquivo, desconsiderando sua extensão, precedido por `cls`. Esta classe deverá obrigatoriamente implementar os métodos `Init` e `Html.` Exemplo da classe `clsIncExample`:

```
Class clsIncExample

    Public Sub Init(dependencies)
        dependencies _
            .SetJSXParameter("param1", True) _
            .SetJSXParameter("param2", False)
            .SetJSXExpression("expr1", objIdioma("E000001")) _
            .SetJSXExpression("expr2", objIdioma("E000002")) _
            .UseCss("/Portal/Modules/Example/Css/incExample.css") _
            .UseJs("/Portal/Modules/Example/Js/incExample.js") _
            .EndChain()
        ...
    End Sub

    Public Sub Html
       ...
    End Sub

End Class
```

## O método Init

O método `Init` será chamado internamente pelo objeto `IncludeManager` (Tabela 1 - Class clsIncludeManager). O `IncludeManager` fornecerá ao método `Init` um parâmetro com uma referência ao objeto `dependencies`, do tipo `clsIncludeManagerDependencies` (Tabela 2 - Class clsIncludeManagerDependencies). O objeto `dependencies` fornecerá os métodos necessários para se registrar as dependências do include.

## O método Html

O método `Html` deverá ser explicitamente chamado pela página ASP principal ou no escopo do método `Html` de outro include. Ele será responsável por produzir o código html a ser incorporado à página. A chamada ao método `Html` deverá ocorrer no ponto do código onde o código html será entregue ao navegador.

## Instanciando e registrando o include junto ao IncludeManager

Para completar a tarefa de criação do include, considerando ainda o arquivo de nome incExample.asp, uma instância da classe `clsIncExample` deverá ser criada e fornecida como parâmetro para o método `Add` do `IncludeManager`, conforme o fragmento de código a seguir:

```
IncludeManager.Add(new clsIncExample)
```



CNPJ: 06.954.022/0001-77

© copyright - webAula

## Tabela 1 - Class clsIncludeManager

| Método | Descrição | Parâmetros |
|---|---|---|
| GetInclude | Obtem a instancia de uma classe referente a um include. | 1. String **className**: Nome da classe. |
| Css | Produz o código html referente ao carregamento das folhas de estilo CSS | |
| Js | Produz o código html referente ao carregamento dos arquivos JavaScript além de produzir o código JavaScript referente às expressões e parâmetros JSX | |
| Add | Adiciona um include ao IncludeManager e retorna a instancia da classe clsIncludeManagerDependencies que diz respeito às dependências do include em questão. | 1. <clsInc> **objInstance**: Instancia da classe referente ao include. |
| MainPageDependencies | Retorna a instancia do objeto do tipo Tabela 2 - Class clsIncludeManagerDependencies que diz respeito à página ASP principal. | |

**Observação**: A classe clsIncludeManager não deve ser instanciada diretamente, o acesso deve ser sempre realizado por meio de uma chamada à função IncludeManager. A grosso modo, a função IncludeManager pode ser entendida como se fosse um objeto do tipo clsIncludeManager, e assim está sendo considerado ao longo deste documento.



CNPJ: 06.954.022/0001-77

© copyright - webAula

## Exemplo de um código completo para um arquivo de include

Considerando um arquivo de include de nome incListFiles.asp, o seu papel será o de produzir um código html que contenha a lista de arquivos de um determinado diretório no servidor. Ele possui uma folha de estilos CSS e um arquivo JavaScript que fornece algum tipo de recurso interativo com o usuário. A folha de estilo e o JavaScript não serão discutidos aqui. Hipoteticamente, o JavaScript associado depende das expressões `expr1`, `expr2` e dos parâmetros `param1`, `param2`, `param3`.

O conteúdo do arquivo de include ficaria da seguinte forma:

```
<%
IncludeManager.Add(new clsIncListFiles)

Class clsIncListFiles

    Private objFSO

    Public Sub Init(dependencies)
        dependencies _
            .SetJSXParameter("param1", True) _
            .SetJSXParameter("param2", False) _
            .SetJSXParameter("param3", True) _
            .SetJSXExpression("expr1", objIdioma("E000001")) _
            .SetJSXExpression("expr2", objIdioma("E000002")) _
            .UseCss("/Portal/Modules/Example/Css/incListFiles.css") _
            .UseJs("/Portal/Modules/Example/Js/incListFiles.js") _
            .EndChain()

        Set objFSO = GetStaticObject(Null, "Scripting.FileSystemObject")
    End Sub

    Private Sub Class_Terminate
        Set objFSO = Nothing
    End Sub

    Public Sub Html(dir)
        Dim files
        Set files = objFSO.GetFolder(dir).Files
        %>
        <div class="incExample">
            <ul>
            <%
            For Each filename In files
                Response.Write("<li>" & filename & "</li>")
            Next
            %>
            </ul>
        </div>
        <%
        Set files = Nothing
    End Sub

End Class
%>
```



CNPJ: 06.954.022/0001-77

© copyright - webAula

## A função global GetStaticObject

Para permitir que includes diferentes possam compartilhar uma instancia de uma determinada classe, sem que seja necessário utilizar uma variável global, o FRAMEWORK fornece a função global `GetStaticObject`. Esta função implementa um comportamento similar ao padrão de projeto *singleton* (GOF-1995), ou seja, quando chamado pela primeira vez, uma instancia da classe é criada, armazenada (para que possa ser reutilizada) e retornada. Se ocorrer uma nova chamada a esta função, pedindo por um objeto do mesmo tipo, o objeto previamente armazenado será retornado. A `GetStaticObject` requer dois parâmetros:

1. String **namespace**: Em alguns casos será necessário que uma instancia de uma classe seja compartilhada apenas por um grupo de elementos. Neste caso, o parâmetro namespace deve ser usado como identificador para o objeto que será criado e armazenado.

2. String **classPath**: Este parâmetro indica o caminho da classe que será instanciada. Ele será internamente fornecido como parâmetro para `Server.CreateObject`.

Exemplo:

```
Dim objFSO
Set objFSO = GetStaticObject("myNameSpace", "Scripting.FileSystemObject")
```

## A função global GetWAStaticObject

Esta função retorna o resultado de uma chamada à GetStaticObject, com a diferença que o parâmetro `classPath` será precedido da expressão `Application("DataBaseType")`.

Exemplo:

```
Dim objNoticias
Set objNoticias = GetWAStaticObject("myNameSpace", "WAInterno.clsNoticias")
```



CNPJ: 06.954.022/0001-77

© copyright - webAula

## Tabela 2 - Class clsIncludeManagerDependencies

| Método | Descrição | Parâmetros |
|---|---|---|
| UseCss | Registra a dependência de um arquivo de folha de estilos CSS. | 1. String **path**: Caminho para o arquivo CSS. |
| UseCssForIElte7 | Registra a dependência de um arquivo CSS para ser carregado apenas por navegadores Internet Explorer versão 7 ou inferior. | 1. String **path**: Caminho para o arquivo CSS. |
| UseCssForIElte8 | Registra a dependência de um arquivo CSS para Internet Explorer 8 ou inferior. | 1. String **path**: Caminho para o arquivo CSS. |
| UseCssForIElte9 | Registra a dependência de um arquivo de folha de estilos CSS para ser carregado apenas por navegadores Internet Explorer versão 9 ou inferior. | 1. String **path**: Caminho para o arquivo CSS. |
| UseJs | Registra a dependência de um arquivo JavaScript. | 1. String **path**: Caminho para o arquivo JavaScript. |
| SetJSXExpression | Define uma expressão que será utilizada por um script (JavaScript) baseado em JSX. | 1. String **term**: Identificador do termo no JavaScript.<br>2. String **translatedTerm**: Expressão traduzida. |
| SetJSXParameter | Define um parâmetro que será utilizado por um script (JavaScript) baseado em JSX. | 1. String **param**: Nome do parâmetro.<br>2. Any **value**: Valor do parâmetro. |
| EndChain | Finaliza um conjunto de chamadas encadeadas, tornando o código livre de erros de sintaxe. Não realiza operações. | |
| **Retorno**: Os métodos acima, exceto o EndChain, retornam uma referência para o objeto a partir do qual estão sendo chamados, permitindo realizar as chamadas de forma encadeada. | | |



CNPJ: 06.954.022/0001-77

© copyright - webAula

# Utilizando os Includes

Tendo em vista que algumas declarações e diretivas são necessárias por todas as páginas ASP do sistema e que na maioria delas, o código referente a estas declarações e diretivas se repete, criamos o arquivo /Portal/Includes/incRegularPage.asp, para que o mesmo seja incluído nas páginas ASP principais, poupando algumas linhas de código.

Os itens fornecidos pelo incRegularPage.asp são:

a. Constantes globais do sistema (que deverão ser definidas em /Portal/Lib/Constants.asp)

b. A biblioteca `IncludeManager` e suas dependências (/Portal/Lib/Portal.vbs)

c. A declaração `Option Explicit`

d. Declaração de `charset`, `codepage` e `lcid`

e. Biblioteca compartilhada do LMS (/ArquivosCompartilhados/libCompartilhada.vbs), carregada por meio do arquivo de include /Portal/Lib/External/incShared.asp

f. Regras de particularidades do LMS (/VBScript/incParticularidades.vbs), carregada por meio do arquivo de include /Portal/Lib/External/incParticularities.asp

g. Habilitação do buffer de saída (`Response.Buffer = True`)

h. Envio dos cabeçalhos para desabilitar o cache pelo navegador

Segue abaixo um exemplo para uma página ASP que utiliza o include incExample.asp, onde a primeira linha diz respeito à inclusão do arquivo incRegularPage.asp e a segunda linha diz respeito à inclusão do arquivo incExample.asp.

```
<!--#include virtual="/Portal/Includes/incRegularPage.asp"-->
<!--#include virtual="/Portal/Modules/Example/Includes/incExample.asp"-->
<!DOCTYPE html>
<html>
<head>
    <title>Página de Exemplo</title>
    <% IncludeManager.Css() %>
</head>
<body>
    <div class="container">
        <h1>Página de Exemplo</h1>
        <% IncludeManager.GetInclude("clsIncExample").Html() %>
    </div>
    <% IncludeManager.Js() %>
</body>
</html>
```



CNPJ: 06.954.022/0001-77

© copyright - webAula

Note que a inclusão do arquivo incExample.asp não produz nenhuma saída html. O que torna seguro que a inclusão seja feita logo nas primeiras linhas do arquivo ASP principal, depois da inclusão do arquivo incRegularPage.asp.

O código html referente ao carregamento das folhas de estilo CSS será gerado a partir da chamada ao método `Css` do `IncludeManager`, que deverá ser realizada no interior da tag `<head>` do documento html. A chamada a `IncludeManager.Css()` irá produzir o código para carregamento dos CSSs referentes a todos os includes que estarão sendo utilizados pela página principal em questão, onde a ordem em que eles serão produzidos estará de acordo com a ordem em que os includes foram declarados.

O código html para o incExample.asp será produzido por meio de uma chamado ao método `Html` do objeto de tipo `clsIncExample`, que no exemplo em questão encontra-se logo abaixo do fragmento de código html `<h1>Página de Exemplo</h1>`. Note que a instancia da classe `clsIncExample` está sendo obtida através da chamada `IncludeManager.GetInclude("clsIncExample").`

Por último, o código html referente ao carregamento dos JavaScripts serão gerados a partir da chamada ao método `Js` do `IncludeManager`, que no exemplo em questão está situado imediatamente anterior ao fechamento da tag `body` do documento html. A chamada `IncludeManager.Js()` produz também o código JavaScript referente às expressões e parâmetros JSX.

## Especificando dependências, parâmetros e expressões na página ASP principal

O IncludeManager fornece, por meio do método `MainPageDependencies`, um mecanismo para especificação de dependências, parâmetros e expressões que dizem respeito exclusivamente à página ASP principal.

Este método retorna uma instancia de um objeto do tipo Tabela 2 - Class clsIncludeManagerDependencies e pode ser utilizado conforme o exemplo:

```
IncludeManager.MainPageDependencies _
    .SetJSXParameter("param1", True) _
    .SetJSXExpression("expr1", objIdioma("E000001")) _
    .UseCss("/Portal/Css/Css.css") _
    .UseJs("/Portal/Js/Js.js") _
    .EndChain()
```



CNPJ: 06.954.022/0001-77

© copyright - webAula

# Parâmetros e expressões JSX no código JavaScript

Considere as declarações de parâmetros e expressões JSX no arquivo de include ASP:

```
Public Sub Init(dependencies)
    dependencies.SetJSXParameter "showUnreadMessagesAlert", True
    Dim unreadMessageExpression
    unreadMessageExpression = "Existem {0} mensagens não lidas de {1} no total"
    dependencies.SetJSXExpression "messages.unread", unreadMessageExpression
End Sub
```

O parâmetro `showUnreadMessagesAlert` é do tipo booleano e o seu papel é o de controlar se será ou não exibido para o usuário um alerta com uma mensagem sobre mensagens não lidas. A expressão JSX referente à mensagem em questão foi definida sob o identificador `messages.unread`. Note que a mensagem está parametrizada, onde os termos `{0}`, `{1}` serão substituídos pelos valores correspondentes antes que o alerta seja gerado para o usuário.

O trecho de código a seguir diz respeito à utilização do parâmetro `showUnreadMessagesAlert` e da expressão `messages.unread` no JavaScript:

```
var unreadMessages, totalMessages;
...
if (JSX.getParameter("showUnreadMessagesAlert")) {
    if(unreadMessages > 0) {
        var message = JSX.i18n.expr("pagination.displayingItems")
            .format(unreadMessages, totalMessages);
        alert(message);
    }
}
```



CNPJ: 06.954.022/0001-77

© copyright - webAula