WebWap – Biblioteca para Desenvolvimento Web em xHarbour

Guia de Referência

Índice

Introdução

A **WEBWAP** é uma biblioteca que possibilita a criação de arquivos executáveis que rodam como ***CGI*** em servidores WEB, buscando informações em bases relacionais ou mesmo em bases DBF com índices nativos (CDX/NTX).

Os dados gerados pelo sistema podem ser acessados com qualquer sistema operacional e também via **Wap** em dispositivos móveis como celulares, palms ou qualquer plataforma que tenha suporte a **Wap**.

Toda a lógica de programação da **WEBWAP** é a mesma dos já habituados "clippeiros". A Saída deve ser uma página HTML devidamente formatada.

Algumas funcionalidades da WebWap:

- Método GET e POST;
- Permite trabalhar com Cookies;
- Permite importação parcial ou total de HTML pré-formatados;
- Criação de Charts (gráficos) com a API do Google.

A WebWap foi desenvolvida com o intuito de permitir uma programação CGI fácil e clara para os desenvolvedores xBase.

O que é CGI

CGI é abreviatura de Common Gateway Interface, uma tecnologia utilizada para fazer a "ponte" entre o navegador e as aplicações alojadas num servidor Web permitindo aumentar a dinâmica do site.

Embora a linguagem tipicamente associada ao CGI seja o PERL, o CGI foi concebido de forma a ser independente da linguagem utilizada.

O que é WAP

WAP é abreviatura de Wireless Application Protocol ou Protocolo de Aplicação sem Fio. É uma tecnologia que permite o acesso à Internet através de aparelhos portáteis como celulares e handhelds, os quais devem estar capacitados a utilizar WAP. Os aparelhos preparados para receber mensagens WAP contêm uma navegador de Internet semelhante aos navegadores utilizados para acesso convencional à Internet via computadores, permitindo o envio de texto e a visualização de imagens.

O que são os métodos GET e POST e qual é a diferença

O método GET é usado quando queremos pesquisar ou passar dados para uma outra página usando a URL da página. Veja um exemplo:

http://www.vagucs.com.br/busca.exe?produto=123

Tudo que é inserido depois do sinal de interrogação (?) é chamado de Query String e pode ser acessado na página atual usando a combinação nome=valor, onde nome é "produto" e valor é "123". Esta forma de passar informações de uma página a outra é a preferida em sites de busca. Se mais de um par nome=valor precisar ser fornecido, o simbolo "&" é usado na separação. Veja:

http://www.vagucs.com.br/busca.exe?produto=123&tipo=1

O método POST é usado quando queremos enviar dados a serem gravados em um banco de dados ou uma pesquisa cujos dados sejam grandes o suficiente para não caber na URL da página. Veja um formulário HTML que usa o método POST para enviar dados a uma página:

```
<form name="teste" method="post" action="pesquisar.exe">
   <input type="text" name="produto">
</form>
```

Aqui a página pesquisar.exe receberá um par composto pelo nome do campo "produto" e o valor informado pelo usuário.

O que são Cookies

Os Cookies são arquivos criados no HD através de sites para que se possa fazer uma comunicação entre o site e o usuário. Como por exemplo, quando logamos em um site/fórum e a opção “Lembrar informações” ou algo do tipo é ativada, quando retornamos ao mesmo link, não precisamos digitar as informações novamente, pois elas ficaram gravadas no Cookie e o site lê esta informação. Em resumo, os sites geralmente utilizam os cookies para distinguir usuários, memorizar preferências ou contar o número de visitantes.

Inicializa_Html()

Inicializa o tipo de saída da CGI.

Sintaxe:

Inicializa_Html([conteudo]) → Nil

Argumentos:

[conteudo]: Parâmetro opcional que pode ser especificado se a saída da biblioteca não for uma saída HTML comum. Pode ser imagem, arquivo binário, ZIP, Wap, etc.

Retorno:

Esta função não possui retorno.

Descrição:

A função *Inicializa_Html()* inicializa o tipo de saída da CGI, permitindo informarmos se não for uma saída HTML comum. Se nada for repassado como parâmetro na função, é entendido que o conteúdo é um HTML padrão.

Veja também:

Inicializa_CGI()

Exemplo:

```
#include "webwap.ch"
Procedure Main
Inicializa_CGI()    // Inicializa as variáveis públicas
Inicializa_Html()   // Inicializa a biblioteca com Html padrão
text
<html>
     <head>
          <title>Exemplo função Inicializa_Html()</title>
     </head>
     <body>
          Teste da função Inicializa_Html()<br>
     </body>
</html>
Endtext
```

Inicializa_CGI()

Inicializa variáveis públicas.

Sintaxe:

Inicializa_CGI() → Nil

Argumentos:

Esta função não possui parâmetros.

Retorno:

Esta função não possui retorno.

Descrição:

A função *Inicializa_CGI()* inicializa todas as variáveis públicas para utilização da biblioteca.

Veja também:

Inicializa_Html()

Exemplo:

```
#include “webwap.ch”
Procedure Main
Inicializa_CGI()    // Inicializa as variáveis públicas
Inicializa_Html()   // Inicializa a biblioteca com Html padrão
text
<html>
     <head>
          <title>Exemplo função Inicializa_CGI()</title>
     </head>
     <body>
          Teste da função Inicializa_CGI()<br>
     </body>
</html>
endtext
```

Get_Value()

Retorna valor de variável.

Sintaxe:

Get_Value(<obj>) → ValorVariavel

Argumentos:

<obj>: Nome do objeto do site.

Retorno:

Esta função retorna o conteúdo da variável do site.

Descrição:

A função *Get_Value()* retorna o valor de alguma variável transmitida pelo Browser. Pode ser campo de algum formulário ou alguma variável passada como parâmetro para a CGI após o "&" no link do mesmo.

Veja também:

Get_Extra_Info(), Get_Cookie()

Exemplo:

Valor.html

```
<html>
<head>
<title>Exemplo Get_Value()</title>
</head>
<body>
<form name="form1" method="post" action="/cgi-bin/valor.exe">
  <input name="campo_texto" type="text" id="campo_texto">
  <input  name="btn_verificar"  type="submit"  id="btn_verificar"
value="Verificar valor inserido">
  <br>%%MSG%%
</form>
</body>
</html>
```

Valor.prg

```
#include "webwap.ch"
Procedure Main
Inicializa_CGI()
Inicializa_Html()
Valor=Get_Value("campo_texto")
if !empty(Valor)
   MSG="O último valor digitado foi "+Valor
end if
import_html("c:\apache\htdocs\valor.html")
```

Get_Extra_Info()

Extrai conteúdo de arquivo.

Sintaxe:

Get_Extra_Info(<obj>) → InformaçãoArquivo

Argumentos:

<obj>: Nome do objeto do site.

Retorno:

Esta função retorna informações sobre o objeto “Arquivo”.

Descrição:

A função *Get_Extra_Info()* é utilizada no caso de campos para envio de arquivos. Para extrair o conteúdo do arquivo deve-se usar esta rotina passando o nome do campo onde o arquivo foi carregado.

Veja também:

Get_Value(), Get_Cookie()

Exemplo:

Getextra.html

```
<html>
<head>
<title>Documento sem t&iacute;tulo</title>
</head>
<body>
<form   action=""   method="post"   enctype="multipart/form-data"
name="form1">
  <input name="arquivo" type="file" id="arquivo">
</form>
%%ERRO_MSG%%
</body>
</html>
```

Getextra.prg

```
#include "webwap.ch"
Procedure Main
Inicializa_CGI()
Inicializa_Html()
nome_arquivo=Lower(Get_Value("arquivo"))
If nome_arquivo==""
   Mensagem()
   Quit
Else
   Bin=Get_Extra_Info("arquivo")
```

```
   If Empty("arquivo")
      Mensagem("Erro no Upload do arquivo!")
      Quit
   End If
End If

Procedure Mensagem(Erro)
If Erro=Nil
   Erro=""
Else
   Erro="<br>"+Erro+"<br><br>"
End If
ERRO_MSG=Erro
Import_Html("getextra.html")
```

Get_Cookie()

Retorna o valor de um cookie.

Sintaxe:

Get_Cookie(<cookie>) → ValorCookie

Argumentos:

<cookie>: Nome do cookie.

Retorno:

Esta função retorna os valores do cookie.

Descrição:

A função *Get_Cookie()* retorna os valores específicos de um cookie informado existente para o site..

Veja também:

Get_Value(), Get_Extra_Info()

Exemplo:

```
#include "webwap.ch"
Procedure Main
Inicializa_CGI()
SET LF TO CHR(13)+CHR(10)
If Get_Cookie("teste")==""
   Set_Cookie("teste",DtoC(Date())+" "+Time())
   Msg="Cookie setado:"+DtoC(Date())+" "+Time()+"<br>A cada vez
que executar este CGI, <br>"+;
       "ele mostrará esta Data e Hora que está armazenada no
Cookie local."
Else
   Msg="Cookie    local:"+Get_Cookie("teste")+"<br>Limpe    seus
cookies para refazer o teste."
End If
Inicializa_Html()
Text
   <html>
      <head>
         <title>Exemplo do uso de cookies</title>
      </head>
      <body>
EndText
?? "Olá mundo!<br>"
?? Msg+"<br>"
Text
      <body>
   </html>
EndText
```

Set_Cookie()

Define o valor de um cookie.

Sintaxe:

Set_Cookie(<nome>,<valor>,[data],[hora],[caminho],[dominio],[secure]) → lOk

Argumentos:

<nome>: Nome do cookie que será definido.

<valor>: Valor do cookie. Esta informação é armazenada no computador do cliente, então evite guardar informações sensíveis e/ou sigilosas.

[data]: Parâmetro no formato data, opcional, porém quando repassado para a função define qual a data de expiração do cookie.

[hora]: Parâmetro caractere no formato de hora (time()) [99:99:99], opcional, porém quando repassado para a função define qual a hora de expiração do cookie. Exemplo: time()+3600 define que o cookie será expirado em 1 hora.

[caminho]: Parâmetro caractere, opcional, porém quando repassado para a função permite definir o grupo de páginas que o cookie se aplica. Como exemplo: "/meuscookies". Se definido como "/" todo o site poderá acessar o cookie. O valor padrão é o diretório atual que o cookie está sendo definido.

[dominio]: Parâmetro caractere, opcional, porém quando repassado permite indicar de quem é o domínio do cookie, como por exemplo seu próprio site: www.seusite.com.br.

[secure]: Parâmetro lógico, opcional. Quando repassado como verdadeiro (.T.), indica que o cookie só deve ser transmitido através de conexão segura HTTPS por parte do cliente. O padrão para este parâmetro é falso (.F.).

Retorno:

Se Set_Cookie for executado com sucesso, ele irá retornar VERDADEIRO (.T.). Isto não indica se o usuário aceitou o cookie.

Descrição:

A função *Set_Cookie()* define um cookie para ser enviado juntamente com o resto dos cabeçalhos HTTP. É criado um cookie na máquina local que sempre será repassado para o script se ele for acionado posteriormente.

Veja também:

Get_Cookie()

Exemplo:

```
#include "webwap.ch"
Procedure Main
Inicializa_CGI()
SET LF TO CHR(13)+CHR(10)
If Get_Cookie("teste")==""
   Set_Cookie("teste",DtoC(Date())+" "+Time())
   Msg="Cookie setado:"+DtoC(Date())+" "+Time()+"<br>A cada vez
que executar este CGI, <br>"+;
       "ele mostrará esta Data e Hora que está armazenada no
Cookie local."
Else
   Msg="Cookie    local:"+Get_Cookie("teste")+"<br>Limpe    seus
cookies para refazer o teste."
End If
Inicializa_Html()
Text
   <html>
      <head>
         <title>Exemplo do uso de cookies</title>
      </head>
      <body>
EndText
?? "Olá mundo!<br>"
?? Msg+"<br>"
Text
      <body>
   </html>
EndText
```

_URL_Encode()

Codifica uma string para uma URL.

Sintaxe:

_URL_Encode(<string>) → cURL

Argumentos:

<string>: Parâmetro caractere que deve ser convertido para URL.

Retorno:

Esta função retorna a string repassada como parâmetro no formato de URL.

Descrição:

A função *_URL_Encode()* codifica uma string para uma URL. Como por exemplo a string ***Grupo Vagucs*** em URL ficaria ***Grupo%20Vagucs***.

Veja também:

_URL_Decode()

_URL_Decode()

Decodifica uma URL para uma string.

Sintaxe:

_URL_Decode(<URL>) → cString

Argumentos:

<URL>: Parâmetro caractere URL que deve ser convertido para string.

Retorno:

Esta função retorna a URL repassada como parâmetro no formato de string.

Descrição:

A função *_URL_Decode()* decodifica uma URL para uma string. Como por exemplo a URL ***Grupo%20Vagucs*** em string ficaria ***Grupo Vagucs***.

Veja também:

_URL_Encode()

Set_Conversion()

Conversão automática de texto.

Sintaxe:

Set_Conversion(<valor>) → Nil

Argumentos:

<valor>: Parâmetro numérico que define a conversão automática dos textos. Veja a tabela abaixo:

0	-	Desabilita conversão.
1	-	Converte as strings para formato HTML, passando de OEM para ANSI.
2	-	Converte as strings para formato HTML, passando de ANSI para OEM.

Retorno:

Esta função retorna os textos convertidos para o padrão HTML.

Descrição:

A função *Set_Conversion()* modifica o tipo de conversão automática de textos.

Veja também:

ENABLE CONVERSION, DISABLE CONVERSION

StrToHtml()

Converte caracteres para o padrão Web.

Sintaxe:

StrToHtml(<Texto>) → cStringConvertido

Argumentos:

<Texto>: Parâmetro caractere para conversão no padrão Web.

Retorno:

Esta função retorna uma string convertida no padrão Web.

Descrição:

A função *StrToHtml()* converte uma seqüência de caracteres para o padrão Web. Exemplos:

Espaço	-	
®	-	&reg
¦	-	&brvbar
Ø	-	&oslash

Veja também:

HtmlToStr()

HtmlToStr()

Converte caracteres do padrão Web convertidos com StrToHtml() para texto livre.

Sintaxe:

HtmlToStr(<Texto>) → cStringLivre

Argumentos:

<Texto>: Parâmetro caractere para conversão do padrão Web para string livre.

Retorno:

Esta função retorna um texto livre a partir de uma string no padrão Web convertida pela função StrToHtml().

Descrição:

A função *HtmlToStr()* converte uma seqüência de caracteres do padrão Web para texto livre. Exemplos:

	-	Espaço
&reg	-	®
&brvbar	-	¦
&oslash	-	Ø

Veja também:

StrToHtml()

Import_Html()

Importação de Html pré-formatado.

Sintaxe:

Import_Html(<nome>,[inicio],[fim]) → lOk

Argumentos:

<nome>: Parâmetro caractere contendo o nome do arquivo Html pré-formatado para importação.

[inicio]: Parâmetro opcional que define a linha inicial do arquivo Html onde irá começar a importação. Se nada for repassado no parâmetro [inicio] é assumida a primeira linha do arquivo.

[fim]: Parâmetro opcional que define a linha final do arquivo Html onde irá finalizar a importação. Se nada for repassado na parâmetro [fim] é assumida a última linha do arquivo.

Retorno:

Esta função retorna VERDADEIRO (.T.) quando o arquivo é encontrado e importado.

Descrição:

A função *Import_Html()* importa um arquivo Html pré-formatado parcialmente ou na íntegra para a saída da CGI. Função muito útil, podendo-se criar arquivos pré-formatados com editores HTML poderosos existentes hoje no mercado como o DreamWeaver e/ou FrontPage e depois somente importando os scripts parciais ou em suas totalidades, facilitando muito a criação de CGI´s.

Veja também:

TEXT, ENDTEXT, ?, ??

Import.html

```
<html>
<head>
<title>Exemplo Import_HTML()</title>
</head>
<body>
  <br>Este é um exemplo ensinando a utilização da função
Import_Html() da biblioteca WebWap.<br>
</body>
</html>
```

Import.prg

```
#include "webwap.ch"
Procedure Main
Inicializa_CGI()
Inicializa_Html()
import_html("Import.html")
```

Date_To_GMT()

Converte data e hora para o formato GMT.

Sintaxe:

Date_To_GMT(<data>,<hora>) → cDataHoraGMT

Argumentos:

<data>: Parâmetro data para conversão no padrão GMT.

<hora>: Parâmetro hora para conversão no padrão GMT.

Retorno:

Esta função retorna uma string contendo hora e data no formato GMT.
<Dia_da_semana>, <Dia>-<Mês>-<Ano> <Hora>:<Minuto>:<Segundo> GMT
Xxx, 99-Xxx-9999 99:99:99 GMT

Descrição:

A função *Date_To_GMT()* converte data e hora para o formato GMT. Exemplo: Quarta-feira, dia 31 de Dezembro de 1998 às 16:15 horas. Ficaria no padrão GMT: “Thu, 31-Dec-1998 16:15:00 GMT”

SET LF TO

Altera finalização da linha (line feed e/ou carriage return).

Sintaxe:

SET LF TO <quebra>

Argumentos:

<quebra>: caracteres ASCII que representam o ***line feed*** e ***carriage return*** das linhas do arquivo.

Retorno:

Este comando não possui retorno.

Descrição:

O comando *SET LF TO* permite alterar o ***line feed*** e o ***carriage return*** dos arquivos permitindo adequar a quebra de linha de acordo com cada sistema operacional.

Os CR/LF (“carriage return” e “line feed”) dos arquivos texto devem ser modificados de acordo com o sistema operacional utilizado. Por exemplo no DOS, cada linha termina com CR/LF enquanto no Linux termina com LF. Se você tentar editar um arquivo texto do DOS no Linux, cada linha provavelmente terminará com um estranho caractere ‘M’ enquanto um texto do Linux sob o DOS aparecerá como uma única e quilométrica linha, sem parágrafos. Daí então a importância do uso do CR/LF.

Na biblioteca WebWap foi definido a inclusão de tags Html para quebra de linha “
” após cada linha de comando Html digitada entre os comandos TEXT e ENDTEXT ou após os comandos ? e ??. Informando o comando SET LF TO no seu programa, as tags Html “
” não serão informadas, passando tudo para o modo manual.

Veja também:

TEXT, ENDTEXT, ?, ??

?

Comando para escrita de textos Html.

Sintaxe:

```
? “<comando_html>”
```

Argumentos:

<comando_html>: Linha de comando Html entre aspas (“).

Descrição:

O comando *?* foi modificado na biblioteca WebWap para permitir a escrita de comandos Html. O que for encrito a partir do comando ? é entendido como script Html pela CGI.

Veja também:

??, TEXT, ENDTEXT

Exemplo:

```
#include “webwap.ch”
Procedure Main
Inicializa_CGI()
Inicializa_Html()
? “<html>”
? “  <head>”
? “       <title>Exemplo função Inicializa_Html()</title>”
? “  </head>”
? “  <body>”
? “       Teste da função Inicializa_Html()<br>”
? “  </body>”
? “</html>”
```

??

Comando para escrita de scpripts Html.

Sintaxe:

?? “<comando_html>”

Argumentos:

<comando_html>: Linha de comando Html entre aspas (“).

Descrição:

O comando *??* foi modificado na biblioteca WebWap para permitir a escrita de comandos Html. O que for encrito a partir do comando ?? é entendido como script Html pela CGI.

Veja também:

?, TEXT, ENDTEXT

Exemplo:

```
#include “webwap.ch”
Procedure Main
Inicializa_CGI()
Inicializa_Html()
?? “<html>”
?? “  <head>”
?? “       <title>Exemplo função Inicializa_Html()</title>”
?? “  </head>”
?? “  <body>”
?? “       Teste da função Inicializa_Html()<br>”
?? “  </body>”
?? “</html>”
```

ENABLE CONVERSION

Ativa a conversão de caracteres Html.

Sintaxe:

ENABLE CONVERSION <TipoConversao>

Argumentos:

<TipoConversao>: Define se o tipo de conversão de caracteres dos comandos Html será de OEM para ANSI ou o inverso, de ANSI para OEM.
OEM para ANSI: ENABLE CONVERSION OEMTOANSI
ANSI para OEM: ENABLE CONVERSION ANSITOOEM

Descrição:

O comando *ENABLE CONVERSION* ativa a conversão de caracteres Html, podendo ser definido para converter OEM para ANSI ou o inverso, ANSI para OEM. Para desabilitar a conversão Html, utilize o comando DISABLE CONVERSION.

Veja também:

DISABLE CONVERSION

DISABLE CONVERSION

Desabilita a conversão Html.

Sintaxe:

```
DISABLE CONVERSION
```

Argumentos:

Este comando não possui argumentos.

Descrição:

O comando *DISABLE CONVERSION* desabilita a conversão Html

Veja também:

ENABLE CONVERSION

TEXT

Define a abertura de seqüência de comandos Html.

Sintaxe:

```
TEXT
```

Argumentos:

Este comando não possui argumentos.

Descrição:

O comando *TEXT* foi modificado na biblioteca WebWap para permitir a digitação de blocos de códigos Html. Tudo que for digitado entre os comandos TEXT e ENDTEXT serão entendidos pela CGI como scripts Html.

Veja também:

```
ENDTEXT
```

Exemplo:

```
#include “webwap.ch”
Procedure Main
Inicializa_CGI()    // Inicializa as variáveis públicas
SET LF TO chr(13)+chr(10)
Inicializa_Html()   // Inicializa a biblioteca com Html padrão
Text                // Inicializa a seqüência de comandos Html
<html>
     <head>
          <title>Exemplo do comando TEXT</title>
     </head>
     <body>
          Teste do comando TEXT da WebWap<br>
     </body>
</html>
EndText             // Finaliza a seqüência de comandos Html
```

ENDTEXT

Define o fechamento da seqüência de comandos Html.

Sintaxe:

ENDTEXT

Argumentos:

Este comando não possui argumentos.

Descrição:

O comando *ENDTEXT* foi modificado na biblioteca WebWap para permitir a digitação de blocos de códigos Html. Tudo que for digitado entre os comandos TEXT e ENDTEXT serão entendidos pela CGI como scripts Html.

Veja também:

TEXT

Exemplo:

```
#include “webwap.ch”
Procedure Main
Inicializa_CGI()    // Inicializa as variáveis públicas
SET LF TO chr(13)+chr(10)
Inicializa_Html()   // Inicializa a biblioteca com Html padrão
Text                // Inicializa a seqüência de comandos Html
<html>
     <head>
          <title>Exemplo do comando TEXT</title>
     </head>
     <body>
          Teste do comando TEXT da WebWap<br>
     </body>
</html>
EndText             // Finaliza a seqüência de comandos Html
```