

Todos os direitos autorais reservados pela **TOTVS S.A.**

Proibida a reprodução total ou parcial, bem como a armazenagem em sistema de recuperação e a transmissão, de qualquer modo ou por qualquer outro meio, seja este eletrônico, mecânico, de fotocópia, de gravação, ou outros, sem prévia autorização por escrito da proprietária.

O desrespeito a essa proibição configura em apropriação indevida dos direitos autorais e patrimoniais da TOTVS.

Conforme artigos 122 e 130 da LEI no. 5.988 de 14 de Dezembro de 1973.

Programação ADVPL Web

Protheus – Versão 12



Sumário

1. Objetivo.....	4
MÓDULO 08: Introdução ao desenvolvimento WEB	4
2. Introdução ao ADVPL Asp	4
3. O Servidor Protheus como um servidor HTTP.....	6
4. Configurando servidor de WEB-Protheus	7
5. Modulos Web.....	11
6. Características do ADVPL ASP - Arquivos .APH.....	21
7. Metodo de Envio e recebimento de dados via WEB	23
7.1. Os métodos de envio GET.....	23
7.2. Os métodos de envio POST	25
7.3. Os métodos de envio HTTPSESSION.....	27
7.4. Os métodos de envio COOKIE	31
7.5. Os métodos de envio HTTPHEADIN	34
7.6. Retorna um valor Advpl numérico , informando a porta utilizada para realizar a requisição.....	35
8. Criação do Potal	36
8.1. Processamento Para varias Empresas/Filiais.....	36
RpcSetEnv.....	37
<FORM name="login" id="login" action="u_NOMEDOPRW.apw" METHOD="POST" >	40
Podemos utilizar funções de JavaScript.....	40
8.2. Processo de Gravação de dados Via Paginas da Web	44
Para isso demonstraremos o código HTML encapsulado no ADVPL como String, para ser executado posteriormente na Pagina	52
9. Pagina Modelo 2/ modelo 3	54
10. Desenvolvimento e impressão de Relatorio na WEB.....	72
Vamos ver os exemplos.....	72
11. UPLOAD / DONWLOAD	77
Para simular os processos da Web demonstraremos os dados de Upload e download utilizando as funções protheus ADVPL juntamente com Javascript.....	77
APÊNDICES	86
12. Funções e Comandos ADVPL	86
APWEXADDERR.....	107
CAPITALACE	108
CTON	109
ESCAPE	109
EXECINPAGE	110
EXISTPAGE	111
EXISTUSRPAGE.....	112

GETJOBPROFSTRING	112
GETWEXVERSION	113
HEXSTRDUMP	114
HTMLNOTAGS	115
HTTPISWEBEX	115
ISEMAIL	116
LOWERACE	116
NTOC	117
REDIRPAGE	118
RETSQLACE	118
RETSQLCOND	119
RETSQLDEL	120
RETSQLFIL	121
RETSQLTAB	121
SEPARA	122
UNESCAPE	123
UPPERACE	124
UPSTRTRAN	124
VALTOSQL	125
VARINFO	126
WEBINFO	126
STARTWEBEX	128
CONNECTWEBEX	129
RESETWEBEX	130
FINISHWEBEX	130
ENDSESSION	131
WEBEXERROR	131

1. Objetivo

Ao final do curso o treinando deverá ter desenvolvido os seguintes conceitos, habilidades e atitudes:

a) Conceitos a serem aprendidos

- estruturas para implementação aplicações ADVPL ASP
- introdução as técnicas de programação de desenvolvimento de telas, Formulários, métodos de transporte de dados
- introdução aos conceitos de inserir, alterar, excluir e visualizar na Web

b) Habilidades e técnicas a serem aprendidas

- desenvolvimento de aplicações voltadas ao ERP/WEB Protheus
- análise de fontes de média complexidade
- desenvolvimento de com linguagens ASP, Java Script, HTML, Advpl Asp

c) Atitudes a serem desenvolvidas

- adquirir conhecimentos através da análise dos funcionalidades disponíveis no ERP Protheus;
- estudar a implementação de fontes com estruturas orientadas a objetos em ADVPL WEB;
- embasar a realização de outros cursos relativos a linguagem ADVPL ASP e WebService

MÓDULO 08: Introdução ao desenvolvimento WEB

2. Introdução ao ADVPL Asp

HTML

HTML, abreviação para 'Hypertext Markup Language', que significa 'Linguagem de Formatação de HyperTexto'

HTML é a formatação padrão adotada para a publicação de HyperTexto na Internet (World Wide Web). O HTML consiste em uma formatação não proprietária, baseada no SGML (Standard Generalized Markup Language), e pode ser criada e processada por um grande número de ferramentas, desde editores de texto-plano (como o NotePad, por exemplo), até sofisticados 'softwares de autoria' WYSIWYG (What You See Is What You Get) .

Basicamente, o HTML utiliza-se dos marcadores < e > para estruturar e formatar texto. Por exemplo:

Letra normal, negrito e <u>sublinhado </u>

A linha de texto acima, representada em um Web Browser, seria mostrada assim :

Letra normal, **negrito** e sublinhado.

HTTP

HTTP é a abreviação de 'Hyper Text Transfer Protocol', que significa 'Protocolo de Transferência de Hyper-Texto'.

O HTTP é um protocolo em nível de aplicação para distribuição de informações. Trata-se de um protocolo genérico, que pode ser utilizado para muitas outras aplicações além de transferência de hipertexto, como nomear servidores e trocas de informações entre sistemas integrados, utilizando-se suas extensões, códigos de erro, métodos de requisição e cabeçalhos (Headers). Uma característica importante do HTTP é a tipagem e normalização da representação da informação, permitindo a construção de sistemas independente do modo pelo qual os dados estão sendo transferidos.

O que é uma Thread?

A maioria dos programadores está familiarizada com programas de execução sequencial. Você provavelmente já deve ter escrito um programa que mostra 'Olá Mundo' ou ordena uma lista de nomes, ou calcula uma lista de números primos. Estes programas são sequenciais, e cada um deles têm um começo, uma sequência de execução e um final. Em qualquer momento da execução do programa, temos apenas um ponto de execução.

Uma Thread é semelhante ao programa descrito acima, tendo um começo, meio e fim. Porém, uma Thread em si não é um programa, pois ela não se executa : ela roda (com) o programa !

Por definição : **Uma Thread é o fluxo sequencial de controle único dentro de um programa.**

Não há nada de novo ou especial sobre programas sequenciais executados em uma simples thread. A grande sacada consiste no uso de Múltiplas Threads dentro de uma aplicação, todas em execução simultânea, porém cada uma realizando tarefas diferentes.

De tal modo que, por estarem realizando tarefas independentes, cada thread possui o seu próprio contexto de execução, alocação de memória e controle de pilha de execução (Stack). O código em execução em uma Thread trabalha dentro de seu contexto específico, de modo que várias outras documentações referem-se ao 'contexto de execução' como sendo um sinônimo de Thread.

Working threads

Damos o nome de 'working thread' quando são iniciadas mais de uma thread independente na aplicação, e todas as threads iniciadas são colocadas em 'modo de espera' (idle), aguardando uma solicitação de processamento da aplicação. Mais de uma solicitação pode ser realizada, e o núcleo da aplicação encarrega-se de distribuir os processamentos entre as threads disponíveis. Terminado o processamento de uma thread, a mesma retorna ao 'modo de espera', estando pronta novamente para atender à uma nova solicitação. Este recurso possibilita um ganho significativo de performance, por não haver a necessidade de criar e finalizar uma nova thread para cada solicitação de processamento.

3. O Servidor Protheus como um servidor HTTP

O servidor Protheus pode ser configurado para trabalhar como um servidor WEB. Isso significa trabalhar como um servidor de requisições dos protocolos HTTP e/ou FTP, do mesmo modo que outros servidores conhecidos no mercado (por exemplo, o IIS - Internet Information Server, da Microsoft (R) ou o Apache para Linux). Assim, basta ter o Protheus para poder criar sua própria Intranet num ambiente de rede local, ou publicar o endereço IP da máquina com o servidor Protheus na Internet e executar funções através de RPC ou simplesmente criar o seu próprio Web Site com páginas HTML estáticas ou dinâmicas.

Serviço de HTTP

O protocolo HTTP (Hyper Text Transfer Protocol) é o protocolo utilizado na comunicação entre um servidor e um Web Browser. É o protocolo utilizado para o envio e recebimento de páginas formatadas em padrões SGML(HTML,XML,etc). Este protocolo se baseia principalmente em dois comandos: GET e POST. O comando GET é utilizado para obter alguma informação do servidor HTTP e o POST para postar informações para o servidor. Mas adiante, será mais fácil compreender onde tais comandos são utilizados no servidor Protheus.

Utilizando o servidor Protheus como um servidor HTTP, o mesmo poderá ser acessado através de um Web Browser como o Internet Explorer por exemplo, que receberá as páginas HTML enviadas de um diretório configurado no servidor. Adicionalmente ao envio e recebimento de páginas estáticas formatadas, pode-se utilizar a linguagem Advpl do Protheus para processar páginas mistas, que contém código Advpl e comandos HTML de formatação. Tais páginas serão processadas no servidor Protheus, e então enviadas para o Web Browser, que irá formatá-las de acordo com os comandos HTML contidos. Também é possível executar diretamente funções compiladas no repositório do Protheus, através de um request HTTP (por exemplo, através de um POST em um formulário em HTML, ou de um link, ou mesmo diretamente na linha de URL do Web Browser. O mesmo vale para qualquer outra aplicação que seja capaz de efetuar comandos GET ou POST utilizando o protocolo HTTP).

Páginas Dinâmicas e Advpl ASP

Quando é utilizado o servidor Protheus para desenvolvimento de aplicações Web, é possível lançar mão do recurso de criação de páginas dinâmicas, isto é, uma requisição HTTP realizada ao Server Protheus, devidamente configurado para atendê-la, dispara o processamento de uma função no Servidor, e esta função encarrega-se de devolver ao usuário uma página HTML com o resultado do processamento.

Para viabilizar o desenvolvimento deste tipo de função, foi criado um tipo de arquivo especial no Protheus IDE, com a extensão .APH, onde é inserido um conteúdo Html a ser enviado ao Web Browser, e instruções Advpl que serão processadas no momento em que a página for solicitada ao servidor Protheus, sendo possível de forma prática 'mesclar' um conteúdo gerado por este processamento à uma página Html para ser retornado ao Web Browser.

Nos tópicos abaixo relacionados, será visto em mais detalhes as configurações necessárias para atender à estas requisições, as características particulares de cada uma delas, e as funções de infra-estrutura criadas para auxiliar o desenvolvimento de aplicações Web.

Além da criação de arquivos, foi disponibilizado no repositório padrão do Protheus as funções de infra-estrutura ApWebEx, desenvolvidas para permitir um melhor aproveitamento dos recursos disponibilizados pela ferramenta Protheus para o desenvolvimento de soluções Web. Estas funcionalidades são exploradas no tópico Infra-Estrutura ApWebEx.

4. Configurando servidor de WEB-Protheus

Nos serviços HTTP e HTTPS, é possível especificar as configurações padrões deste protocolo e propriedades gerais aplicadas a todos os hosts e URLs de acesso utilizadas pelos projetos Web.

O protocolo FTP (File Transfer Protocol) permite a transferência de arquivos entre um servidor e uma aplicação client de FTP (com um Web Browser como o Internet Explorer, por exemplo). Utilizando o Protheus Server como um servidor FTP, os usuários poderão, remotamente, "baixar" arquivos disponibilizados em um diretório no servidor.

A habilitação do serviço de HTTP é necessária para a instalação dos módulos Web. Durante a instalação de um módulo Web, caso o serviço de HTTP não esteja habilitado, esta operação será executada automaticamente.

Ao expandir o tópico "Servidor HTTP", são mostrados os itens HTTP, HTTPS e FTP, que permitem a edição independente das configurações de cada um destes protocolos. Para cada um deles, são permitidas as operações de edição e exclusão da respectiva configuração.

Neste tópico iremos abordar somente a criação do Serviço HTTP

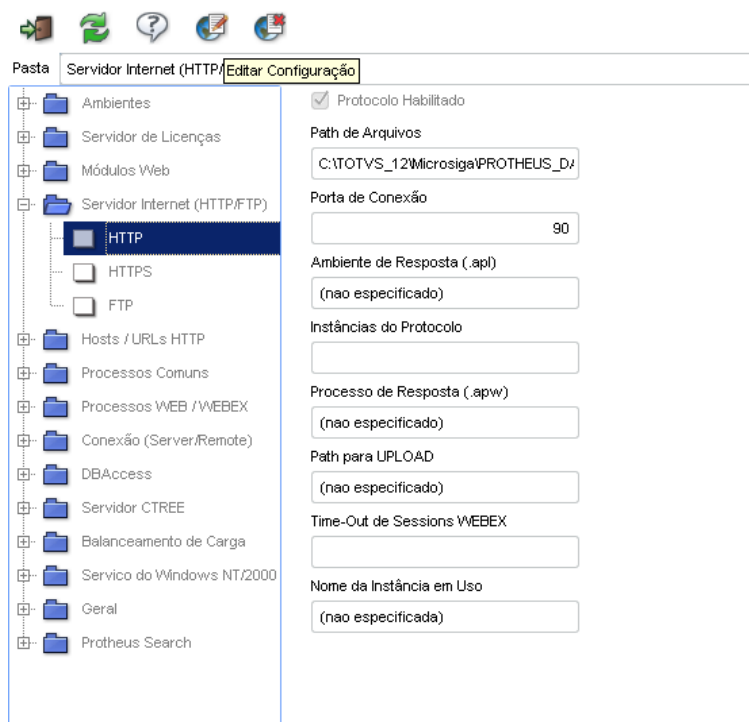
Editando a Configuração HTTP

Para inserir / editar uma configuração HTTP:

1. Abra o Wizard pelo programa inicial do smarclient




2. Clique no botão "OK"
3. No tópico "Servidor HTTP", posicione o cursor sobre o item "HTTP" e clique em - "Editar Configuração", nas Ações Relacionadas.



4. Clique em editar para abrir a tela

Assistente de Configuração HTTP


Configurações da Seção

☒ Protocolo Habilitado

Nome da Instância

Path de Arquivos

Porta de Conexão

Ambiente

Processo de Resposta

Instâncias do Protocolo (mínimo)

Instâncias do Protocolo (máximo)

Path para Upload de Arquivos

Time-Out de Sessions WEBEX (em segundos)

Será apresentada uma única janela, contendo as configurações padrões atuais para o serviço de http.

Protocolo Habilitado

Através deste campo é possível desabilitar a utilização do protocolo http, sem deletar as configurações atuais desta seção.

Nome da Instância

Este campo não está disponível para edição. Caso esteja preenchido, informa que um módulo Web foi instalado no host "HTTP [default]"; neste caso, não é possível alterar as informações de path, porta de conexão, ambiente e processo de resposta.

Se necessário alterar as informações de configuração padrão do protocolo, deve-se utilizar o assistente de edição de Módulos Web, editando a instância que está utilizando a seção http [default].

Path de Arquivos

Especifica o diretório raiz a ser utilizado pelo protocolo "HTTP" para o acesso a arquivos estáticos e imagens.

Deve ser informado com unidade de disco e caminho completo.

Porta de Conexão

Informa a porta de conexão utilizada. Para HTTP, a porta padrão é a 80.

Ambiente

Permite selecionar um ambiente (Environment) neste ByYou Application Server para atender às solicitações de processamento de links ".apl".

Processo de Resposta

Permite selecionar um processo WEB/WEBEX configurado neste ByYou Application Server para atender às solicitações de processamento de links ".apw".

Instâncias de Protocolo (mínimo e máximo)

Nestas configurações, é possível especificar um número mínimo e máximo de processos internos referentes ao serviço de HTTP. Estes processos internos são utilizados para o atendimento simultâneo das requisições de conteúdo estático, arquivos, imagens, e demais arquivos disponíveis a partir da pasta definida em "Path de Arquivos", através deste protocolo(*).

Path para Upload de Arquivos

Caso o host HTTP [default] esteja sendo utilizado com um processo de resposta de um projeto Web que suporte a funcionalidade de Upload de arquivos via HTTP,

através desta chave, é possível configurar a partir de qual diretório serão gravados os arquivos enviados via http (relativo ao diretório raiz do ambiente utilizado pelo processo de resposta).

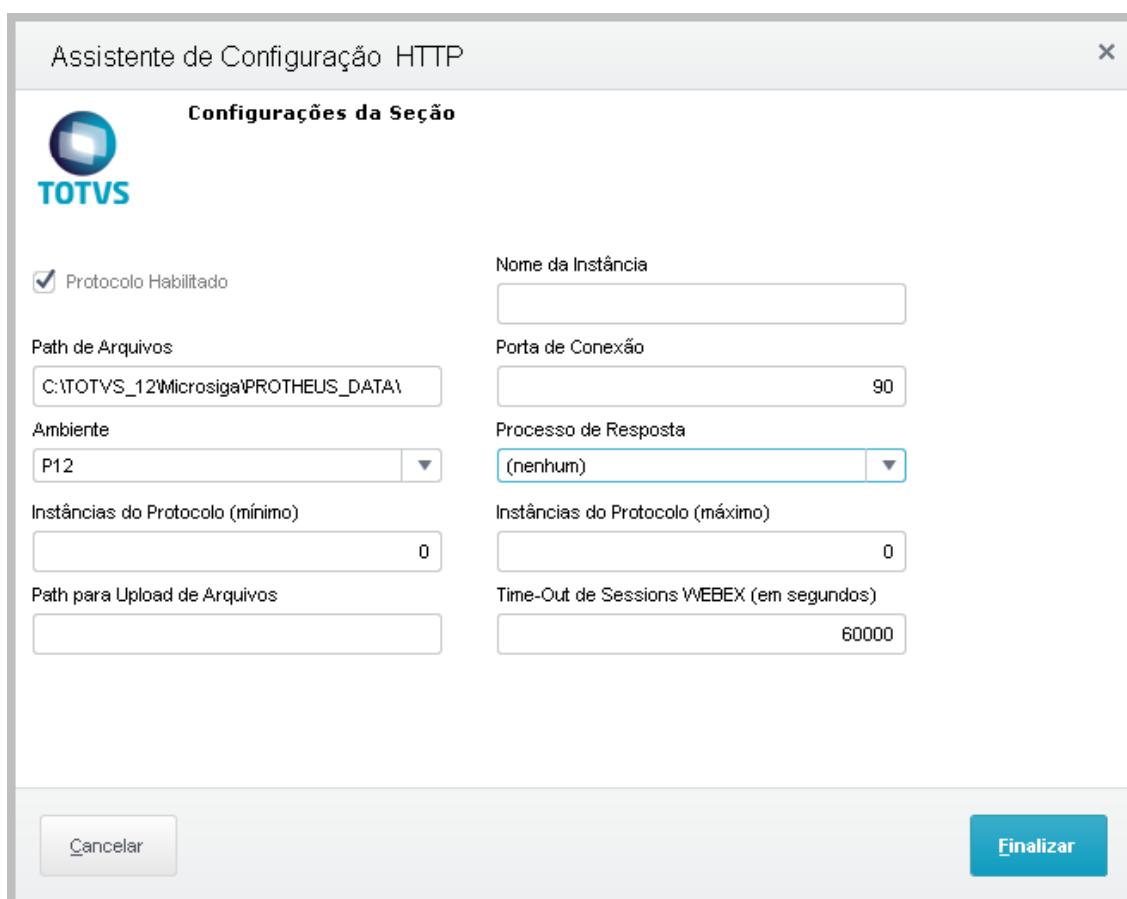
Esta configuração é atualizada, automaticamente, conforme o módulo web instalado.

Time-out de Sessões WEBEX (em segundos)

Ao configurar um ou mais módulos Web que utilizem sessões de usuário através de um Processo WEBEX, é possível definir qual será o tempo de permanência em inatividade em memória das variáveis de sessões utilizadas pelos usuários do módulo web.


Caso seja não especificado, o valor padrão é equivalente a 3600 segundos (uma hora).

(*) Vale ressaltar que uma thread HTTP não possui, necessariamente, ligação implícita com uma Thread AdvPL. Um Web Browser, quando solicita um arquivo HTML ou uma imagem, estabelece uma conexão HTTP com o ByYou Application Server, para receber o dado solicitado. Quando o browse recebe a informação desejada, fecha esta conexão, mantendo a Thread HTTP do Protheus disponível para atender a outras requisições HTTP, oriundas deste ou de outro Web Browser.



Assistente de Configuração HTTP

Configurações da Seção



☒ Protocolo Habilitado

Nome da Instância

Path de Arquivos

C:\TOTVS_12\Microsiga\PROTHEUS_DATA\

Porta de Conexão

90

Ambiente

P12

Processo de Resposta

(nenhum)

Instâncias do Protocolo (mínimo)

0

Instâncias do Protocolo (máximo)

0

Path para Upload de Arquivos

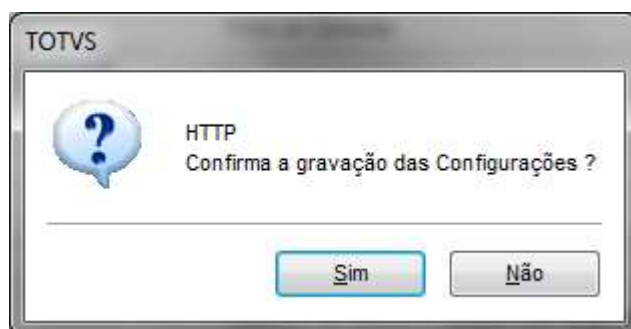
Time-Out de Sessions WEBEX (em segundos)

60000

Cancelar Finalizar

Para gravar as configurações atuais, deve-se clicar em “Finalizar”.

Ao confirmar a gravação, o arquivo de configurações do ByYou Application Server (appserver.ini) será atualizado e o Assistente será reiniciado, apresentando a tela principal do Wizard.



5. Modulos Web

Neste tópico, é possível instalar, configurar e excluir as configurações e arquivos adicionais pertinentes aos módulos Web disponibilizados pelo Sistema.

Os módulos Web disponibilizados são:

- DW - Data Warehouse
- BSC - Balanced Scorecard
- GE - Gestão Educacional
- TCF - Terminal do Funcionário (RH on-line)
- PP - Portal Protheus
- WS - Web Services
- WPS - WebPrint/WebSpool
- MAK - Módulo Webex Makira (ambientes customizados)
- GPR - Gestão de Pesquisas e Resultados
- GAC - Gestão de Acervos

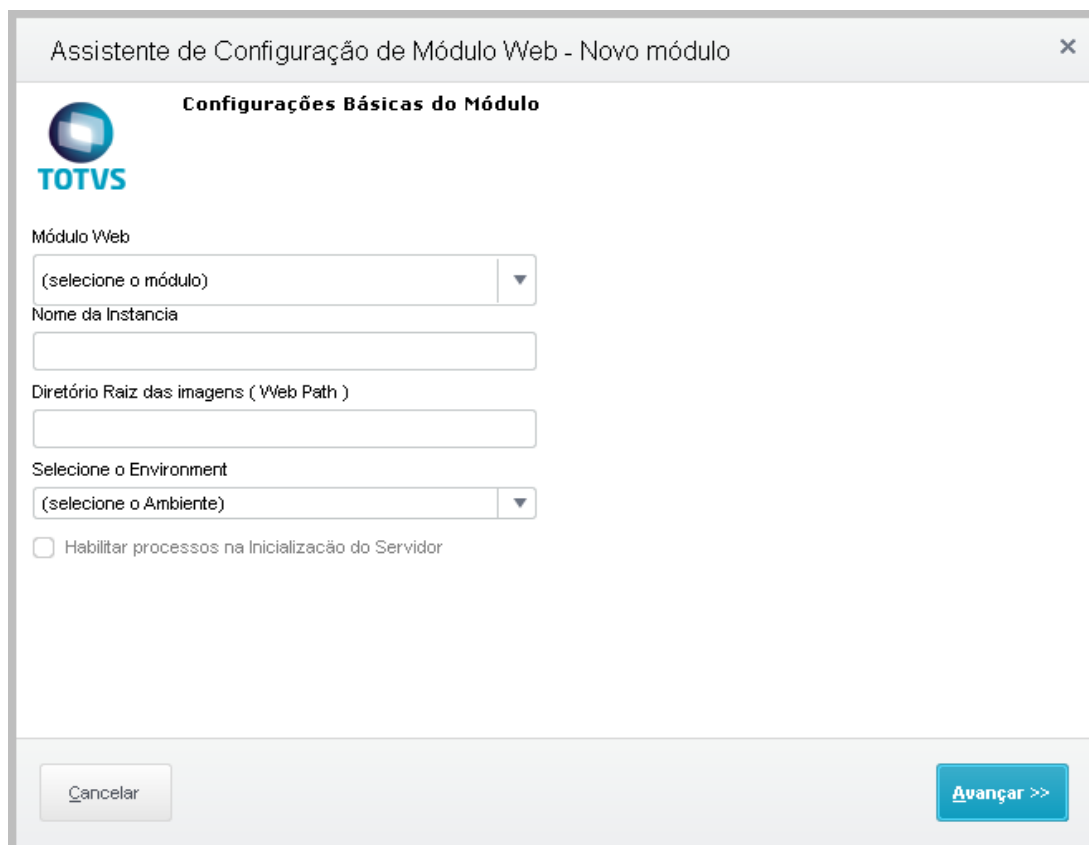
Instalando um Módulo Web

Para instalar um módulo Web:

1. Abra o Wizard através da tela do smartclient.



2. Clique no ícone Wizard
3. Posicione com o mouse sobre o tópico “Modulos Web” na árvore de tópicos, e clique em “Novo Módulo”, na barra de botões.



Módulo Web

Selecione o módulo Web que deve ser instalado. Para instalação do módulo PP - Portal Protheus, GPR - Gestão de Pesquisa e Resultado e GAC - Gestão de Acervos, é necessária a instalação prévia do módulo Web Services.

Nome da Instância

Informe o nome para identificação desta configuração do módulo Web; não utilize caracteres acentuados ou espaços.

Este nome será utilizado para individualizar as configurações das instalações do módulo Web, assim, se a empresa necessita aplicar diferentes configurações para um mesmo módulo Web, é possível instalá-lo sob uma nova instância.

Exemplo: Na instalação do módulo GE - Educacional, cada unidade educacional pode utilizar um conjunto diferente de imagens para apresentação do seu site ou, ainda, um environment diferente no Server Protheus da versão correspondente; para isto, será necessário criar diferentes instâncias.

Diretório Raiz de Imagens (Web Path)

Informe o diretório para instalação das imagens e dos demais arquivos (.css,.jar,.htm, etc) deste módulo, que serão utilizados para apresentação no browser.

Este diretório será criado abaixo do diretório raiz (RootPath) do Ambiente (environment) selecionado para a instalação.

Para cada instalação de módulo Web, deverá ser especificado um diretório diferente, iniciando com o sinal de "\" (barra inversa).

Environment

Selecione o environment (ambiente) que será utilizado para execução do módulo. São relacionados todos os ambientes disponíveis no Server ativo.

Habilitar processos na inicialização do Servidor


Caso esta configuração seja selecionada, os processos WEB / WEBEX criados para esta configuração de módulo serão automaticamente inseridos na configuração de "OnStart" do ByYou Application Server.

URL do Protheus Web Services

Este campo somente é exibido na instalação do módulo PP - Portal Protheus; neste caso, deve ser preenchido com a URL utilizada na instalação do módulo Web Services, precedido por "HTTP://".

Assistente de Configuração de Módulo Web - Editando Instância : [PP]

Configurações Básicas do Módulo



Módulo Web
PP - Portal Protheus

Nome da Instância
PP

Diretório Raiz das imagens (Web Path)
C:\TOTVS_12\Microsigla\protheus_data\web\PP

Selecione o Environment
P12

☒ Habilitar processos na Inicialização do Servidor

URL do Protheus Web Services
http://localhos:90/ws

Cancelar Avançar >>

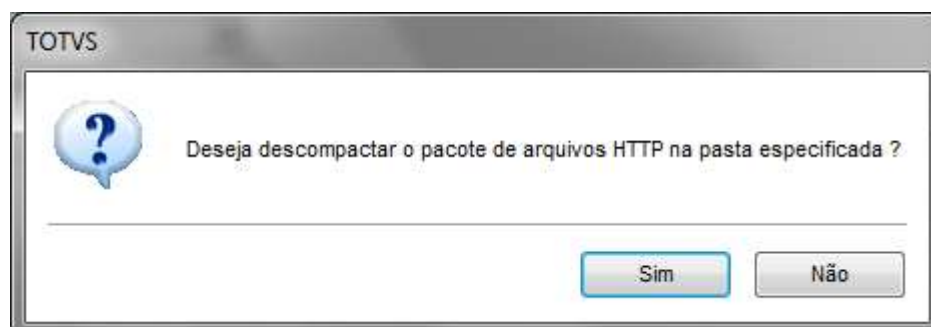
4. Para prosseguir para a segunda tela, clique em "Avançar".

O Assistente irá consistir as informações fornecidas e determinar se o ambiente está apto para instalação do módulo.

Deve-se observar que para instalação dos módulos Web, é necessário que os pacotes de instalação dos arquivos Web (.MZIP) estejam disponíveis na pasta

"SYSTEMLOAD" localizada abaixo do diretório raiz (RootPath) do Ambiente (environment). Caso os pacotes não sejam localizados, será apresentada uma janela de advertência.

A instalação poderá prosseguir; no entanto, os arquivos Web não serão descompactados, sendo apenas atualizada a configuração do servidor. Em seguida, será apresentada a janela "Configuração de Host x Empresas/Filiais".



5. Informe os dados conforme a orientação a seguir:

Assistente de Configuração de Módulo Web - Editando Instância : [PP]

Configuração de Hosts X Empresas/Filiais
Definindo o host e as empresas

Host (Pode incluir o diretório virtual)

localhost:90/portal

[HTTP] DEFAULT HOST
localhost:90/pp

Selecione a Empresa/Filial

***** - Todas as Empresas e Filiais

Relacionar Excluir

Relacionamentos

Host	Empresa/Filial
localhost:90/pp	***** - Todas as Empresas e Filiais

Cancelar << Voltar Avançar >>

Host

Informe o endereço Web a partir do qual o módulo será acessado, por meio de um browser.

Exemplos:

"www.nomedosite.com.br" (para um ambiente Internet)

"nomedosservidor" (para um ambiente Intranet).

Pode-se, adicionalmente, informar um diretório virtual após o Host, separando-os por uma barra "/". Isto permite que seja instalado, no mesmo host, mais de um módulo Web.

Exemplos:

“nomedoservidor/ws” (para webservice)

“nomedoservidor/pp” (para o Portal)

Não se deve especificar o protocolo utilizado (como "HTTP://" ou "HTTPS://").

Vale ressaltar que é possível especificar um nome de host, não sendo obrigatoriamente o nome da estação servidor, desde que o nome especificado esteja registrado em um servidor de DNS (que relaciona um nome de host ao IP do equipamento servidor) e visível no âmbito do parque de máquinas-cliente da aplicação Web.

Selecione as Empresa/Filiais

Na área "Seleção Empresas/Filiais", selecione a empresa/filial para a qual está sendo configurado este host.

Se a instalação for referente aos módulos BSC, PP e WPS, estará disponível apenas a opção "Todas as Empresas".

6. Após informar o Host e selecionar um item da área de “Seleção Empresa/Filiais”, clique em "Relacionar".

A amarração do host informado com este item será apresentada na janela "Relacionamentos".

É possível criar diversos relacionamentos, o Assistente, automaticamente, irá criticar as amarrações, de acordo com as características operacionais do módulo em instalação.

Assistente de Configuração de Módulo Web - Editando Instância : [PP]

Configuração de Hosts X Empresas/Filiais
Definindo o host e as empresas

Host (Pode incluir o diretório virtual)

***** - Todas as Empresas e Filiais

[HTTP] DEFAULT HOST
localhost:90/pp
localhost:90/portal

Relacionar Excluir

Relacionamentos

Host	Empresa/Filial
localhost:90/pp	***** - Todas as Empresas e Filiais
localhost:90/portal	***** - Todas as Empresas e Filiais

Cancelar << Voltar Avançar >>

Exemplo: O módulo de WebServices não permite amarrar um mesmo host a mais de uma empresa/filial; já para o módulo TCF, esta amarração é possível.

7. Se necessário excluir um relacionamento, posicione o cursor sobre este e clique em "Excluir".
8. Clique em "Avançar" para prosseguir.

Não é possível prosseguir para a próxima tela sem que seja informada, no mínimo, uma amarração entre um host e uma Empresa/Filial.

9. Informe os dados conforme a orientação a seguir:

Host Virtual

Apresenta o host configurado.

Empresa/Filial

Apresenta a empresa/filial relacionada.

Mínimo Usuários

Informe a expectativa mínima de usuários que irão acessar o site.

Máximo Usuários

Informe a expectativa máxima de usuários que irão acessar o site.

Com base nos campos "Mínimo Usuários" e "Máximo Usuários", o Assistente irá determinar a quantidade média de processos (working threads) que o site irá utilizar.

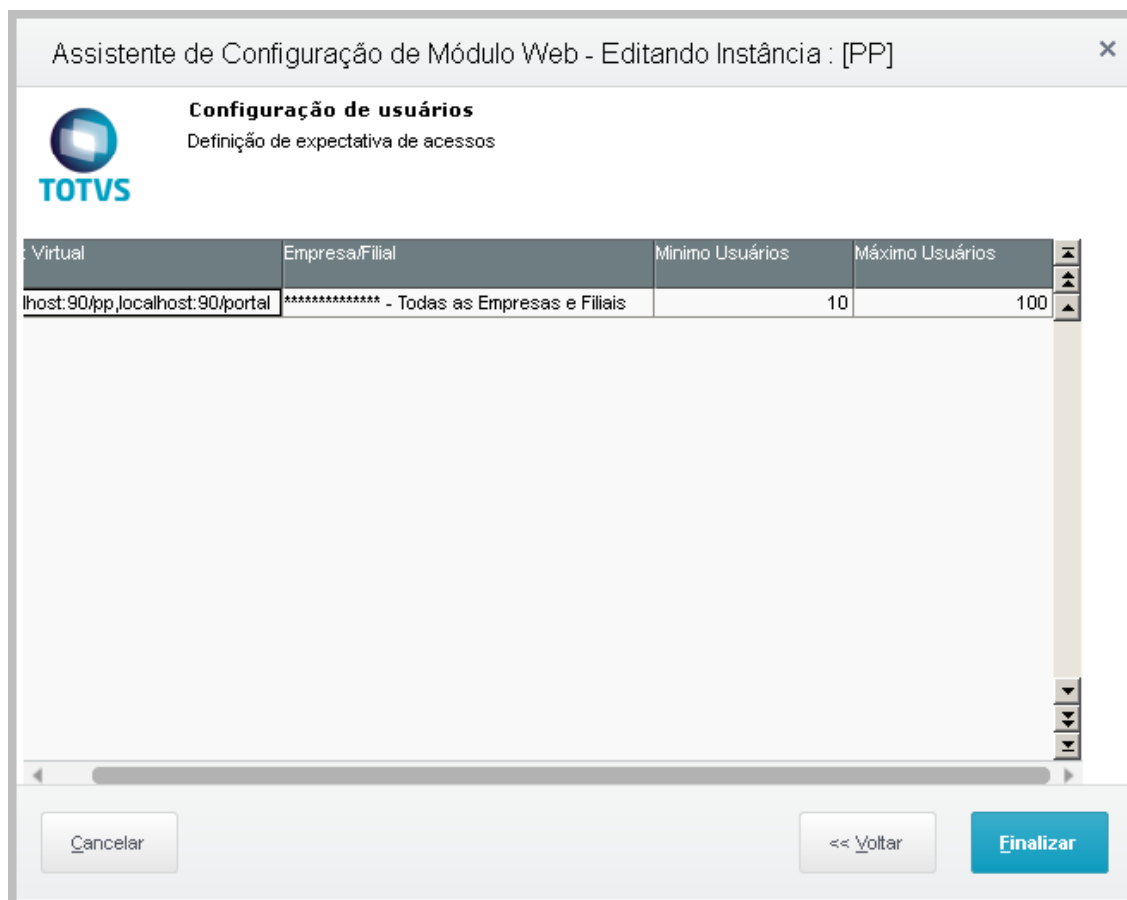
Exemplo:

Mínimo: 5

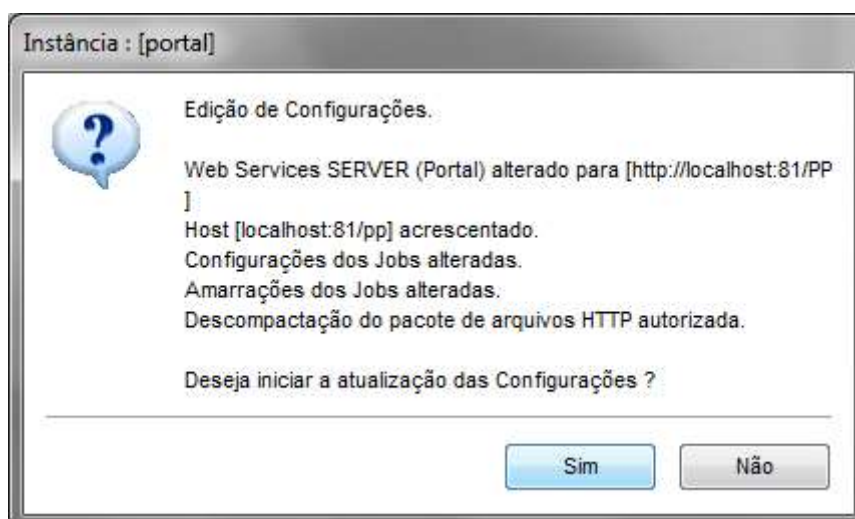
Máximo: 10

Com base nesta configuração, o Protheus irá determinar um número de processos para atender a essa demanda. Considerando que a média de usuários por processo é de 10 para 1, neste caso, o Protheus irá configurar o número mínimo de 1 processo e o máximo de 3.

A informação do número mínimo não é necessária; caso omitida, será considerado 1 processo. A informação do número máximo é obrigatória.



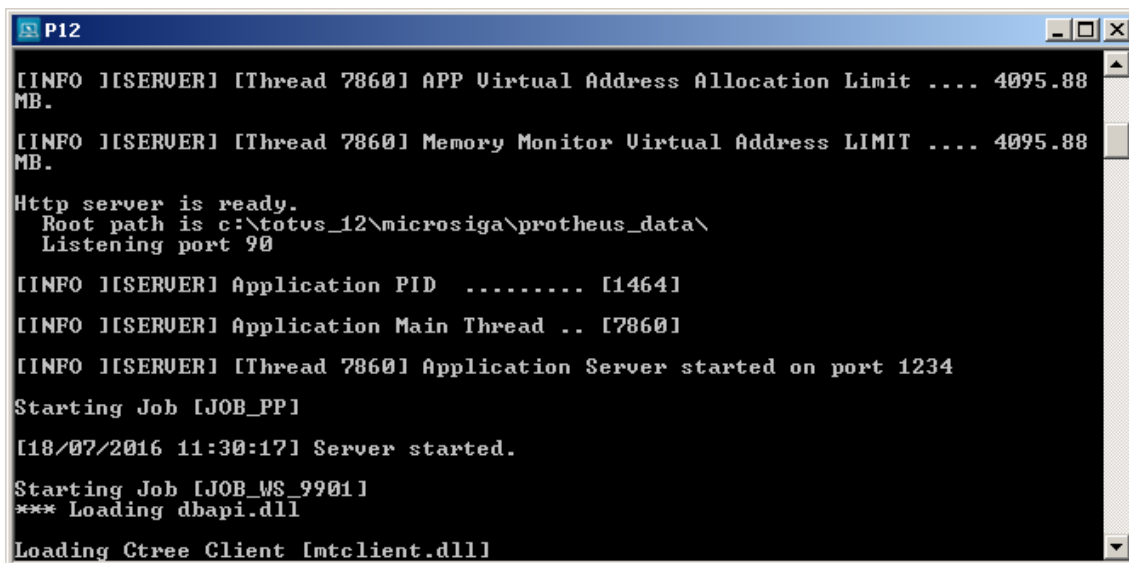
10. Para finalizar a instalação do módulo, clique em “Finalizar”.



Ao confirmar a instalação, o pacote de arquivos do módulo Web será descompactado no diretório raiz de imagens informado, e o arquivo de configurações do ByYou Application Server (appserver.ini) será atualizado com as definições pertinentes ao módulo (Host, Processos WEB/WEBEX).

Ao final do processo de atualização, o Assistente será reiniciado para apresentação da tela inicial do Wizard.

Feche o assistente e Levante o serviço em modo console para verificar o seu funcionamento.



```
P12
[INFO ][SERVER] [Thread 7860] APP Virtual Address Allocation Limit .... 4095.88 MB.
[INFO ][SERVER] [Thread 7860] Memory Monitor Virtual Address LIMIT .... 4095.88 MB.
Http server is ready.
  Root path is c:\totvs_12\microsig\protheus_data\
  Listening port 90
[INFO ][SERVER] Application PID ..... [1464]
[INFO ][SERVER] Application Main Thread .. [7860]
[INFO ][SERVER] [Thread 7860] Application Server started on port 1234
Starting Job [JOB_PP]
[18/07/2016 11:30:17] Server started.
Starting Job [JOB_WS_9901]
*** Loading dbapi.dll
Loading Ctree Client [mtclient.dll]
```

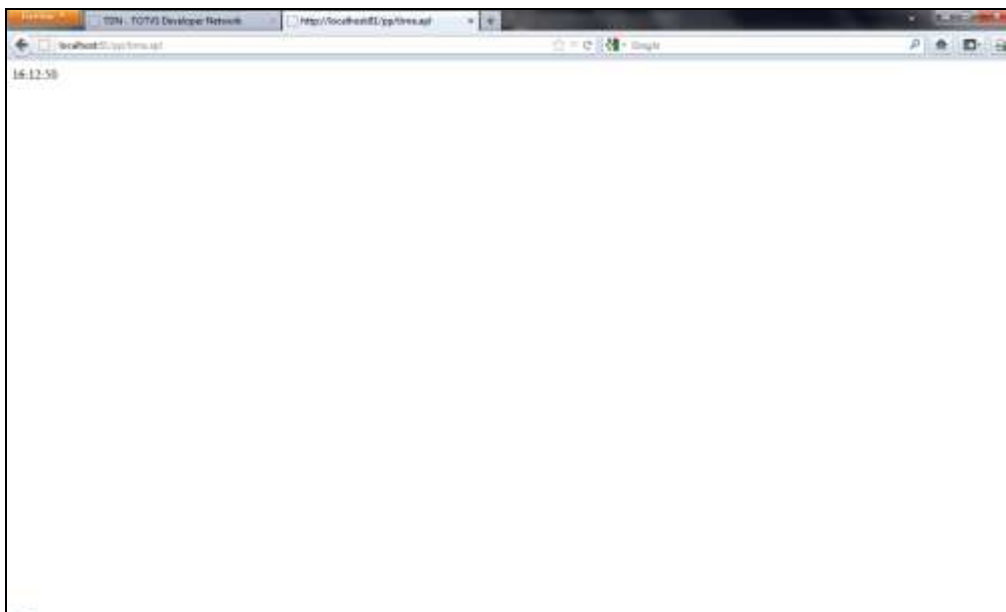
Princípio de Funcionamento do HTTP

A utilização do protocolo HTTP não envolve o uso de uma conexão persistente entre o Web Browser e o Servidor HTTP: Isto significa que, ao ser solicitada uma página, imagem ou até o processamento de uma função, o Web Browser abre uma conexão com o Server HTTP, realiza a solicitação e fica com a conexão aberta, aguardando pelo retorno do Server. Quando o server já houver enviado os dados solicitados, a conexão é fechada .

Processamento de Funções

Para o retorno de páginas estáticas, imagens e demais arquivos via HTTP, o servidor identifica o que está sendo solicitado pelo Web Browser através da extensão do Link. Por exemplo, ao receber uma solicitação da URL <http://localhost:90/pp/htmls/imagem.gif>, o Server HTTP sabe que deve procurar um arquivo chamado imagem.gif, dentro da pasta htmls a partir da pasta local no servidor configurada para armazenar os arquivos para o acesso HTTP.

No servidor Protheus, foram implementadas duas extensões de Link para permitir a execução de funções Advpl através de uma requisição HTTP : A extensão .APL e a extensão .APW . Quando o servidor recebe uma requisição via HTTP, por exemplo da url <http://localhost:90/pp/time.apl>, e está corretamente configurado para atender à esta requisição, o Protheus Server realiza o processamento, e informa para o Web Browser solicitante que a string que será retornada deve ser interpretada pelo Web Browser como sendo um Script HTML .



Características comuns do processamento de funções Advpl via HTTP

Independente da extensão de link utilizada, existem características e pré-requisitos comuns ao funcionamento de ambas as requisições, que seguem abaixo :

A função, seja Advpl ou uma função compilada no Repositório, deve obrigatoriamente ter um retorno do tipo String, pois o Web Browser solicitante será informado pelo Server Protheus que a string retornada deverá ser interpretada e mostrada pelo Browser como um HTML.

Devido à origem da requisição não ter relação alguma com a interface Remote do Protheus, não é possível utilizar determinadas funções Advpl que foram escritas exclusivamente para esta interface, como por exemplo as funções MsgStop, Alert, e demais funções e objetos de Interface de Janelas.

O que irá diferenciar uma função executada via link .apl e .apw será as etapas de execução das mesmas. Esta característica de funcionamento será melhor esclarecida após a leitura dos tópicos sobre desenvolvimento de funções .apl e desenvolvimento de funções .apw

Uma página ASP (Active Server Pages) é uma página HTML contendo código interpretável em uma linguagem compreensível ao servidor HTTP em uso. Por exemplo, o IIS da Microsoft utiliza o VBScript para criar suas páginas ASP, do mesmo modo que o Protheus utiliza o ADVPL. Uma página ASP é uma combinação de script HTML e código interpretável. No ADVPL ASP esse código é padrão xBase, portanto a preocupação maior daqueles que já conhecem e trabalham com o Protheus e desejam desenvolver páginas ativas para aplicações Web utilizando essa facilidade é conhecer HTML.

6. Características do ADVPL ASP - Arquivos .APH

Os arquivos ADVPL ASP têm a extensão padrão .APH. São arquivos texto e devem ser adicionados a um projeto no Protheus IDE e compilados da mesma maneira que os programas tradicionais. A diferença é que o Protheus Server identificará que se trata de um ADVPL ASP e executará uma espécie de tradutor (parser) antes que a compilação seja executada. Este parser irá transformar todo o arquivo em uma função única, que receberá os mesmos parâmetros das funções APL simples, como explicado anteriormente no Item 'Desenvolvendo Funções .APL', e retornará uma string.

O desenvolvedor não precisa se preocupar em retornar HTML algum, pois o APH também é um arquivo HTML. A função que foi gerada na compilação irá se encarregar de retornar o HTML contigo no arquivo, depois que o código foi processado. Um Arquivo APH gera no repositório de Objetos do Protheus uma função com o mesmo nome do arquivo, porém prefixada com:

H_ + nome_do_arquivo_apw

Por se tornar uma função no momento da compilação, não é possível utilizar a cláusula FUNCTION para criar outras funções dentro de um arquivo APH. Caso exista essa necessidade, tais funções devem ser criadas em um arquivo PRW tradicional e chamadas de dentro do APH.

A extensão APH para o nome dos arquivos é obrigatória. Qualquer outra extensão usada no nome do arquivo não será reconhecida e o parser do IDE não será executado durante a compilação. Foi criada também a extensão de arquivos .AHU (Aph de Usuário), que possui o mesmo tratamento de Parser do arquivo APH, gerando uma função prefixada com L_. A diferença é que a função gerada pelo AHU pode ser gerada sem a necessidade de autorização de compilação com permissão para substituir fontes microsigla, por tratar-se de um Aph de Usuário, equivalente a uma User Function.

Assim como outros ASP's, a diferenciação entre script HTML e código é efetuada através dos caracteres <% para indicação de abertura de código e %> para indicação do encerramento de código. Por exemplo, o HTML abaixo contém um pedaço de código ADVPL separado pelos delimitadores:

Quando este arquivo for requisitado ao Protheus Server (através de uma chamada em URL por exemplo) o código entre os delimitadores será executado, porém o script colocado ao redor do código será mantido exatamente como se encontra. Por exemplo :

OBS: A padronização do Protheus refere-se a o nome do arquivo fonte ANTECEDENDO A LETRA H-> HTML + NOME DO ARQUIVO DO CODIGO FONTE + COM A EXTENSÃO APL

A grande vantagem de se utilizar dos arquivos ADVPL ASP em relação a criar funções APL simples, decorre do fato de que nas funções APL simples o desenvolvedor deve se preocupar em retornar todo o HTML necessário para a correta exibição no Web Browser.

E também, como o ADVPL ASP mistura o script HTML com o código interpretável, pode-se criar um arquivo APH utilizando o editor desejado (como o Microsoft FrontPage, por exemplo) e inserir nele os códigos Advpl necessários entre as Tags. Outro detalhe importante é que pode-se utilizar as estruturas de fluxo da linguagem ADVPL para repetir comandos do próprio script HTML (por exemplo, colocar um comando de script HTML dentro de um comando While em ADVPL):

```
<% While !EOF() %>
<B> Esta linha será repetida no HTML até ocorrer o fim de arquivo </B>
<%
dbSkip()
```

```
EndDo  
%>
```

Note que também pode existir diferentes blocos de código interpretável separados pelos delimitadores, dentro de um mesmo arquivo.

Tão importante quanto mesclar código interpretável com script de formatação HTML, é utilizar os comandos ADVPL para alterar o script de formatação. Ou seja, colocar o conteúdo de variáveis, campos, etc, diretamente no HTML que será enviado à aplicação client (ao Browser por exemplo). Isso pode ser realizado através dos delimitadores de avaliação. Os delimitadores de avaliação são `<%=` para abertura e `%>` para encerramento. Diferentemente dos delimitadores de código interpretável, estes devem sempre estar na mesma linha. Com eles pode-se criar uma linha de script HTML, cujo conteúdo contém uma expressão que será avaliada em tempo de execução, e seu resultado inserido como parte do HTML retornado ao Browse:

```
<b>Esta linha é HTML, mas o horário exibido aqui: <%= Time() %> foi obtido em tempo  
de execução no Servidor.</b>
```

No exemplo acima, a linha HTML será retornada para o Browser com o resultado da função time (ou seja, a hora atual no servidor) inserido no texto.

Utilizando todos esses conceitos, pode-se criar toda uma aplicação Web baseada no Protheus. Ou seja, o processamento e acesso aos dados fica por conta do Protheus Server, e a interface fica por conta do Browser (utilizando o HTML) .

Importante

Ao codificar um arquivo .APH e/ou .AHU , devemos estar atentos às regras de utilização dos delimitadores de execução e avaliação Advpl:

A Abertura e fechamento dos delimitadores de execução `<% ... %>` devem estar isoladas em suas respectivas linhas, não devendo ter qualquer conteúdo adicional, nem duas aberturas e fechamentos na mesma linha. Partindo dessa premissa, veja abaixo alguns exemplos de como inserir o código Advpl abaixo dentro de um APH

Exemplo:

```
IF !IOk  
nErro++  
Endif
```

Certo

```
<%  
IF !IOk
```

```
nErro++  
Endif  
%>
```

Certo

```
<%  
IF !IOk  
nErro++  
Endif  
%>
```

Certo

```
<% IF !IOk %>  
<% nErro++ %>  
<% Endif %>
```

Certo

```
<% IF !IOk ; nErro++ ; Endif %>
```

Errado

```
<% IF !IOk %><% nErro++ %>-- 2 aberturas e fechamentos na mesma linha  
<% Endif %>
```

Quanto aos delimitadores de avaliação `<%= ... %>` , podemos ter várias aberturas e fechamentos na mesma linha , porém não podemos ter uma abertura e seu respectivo fechamento em linhas diferentes.

Uma linha qualquer em um arquivo .APH nao deve conter mais do que 150 Caracteres, pois o Parser insere caracteres de controle em cada linha do mesmo durante a pré-compilação . e a linha final resultante não pode ultrapassar 254 caracteres, pois neste caso isto impossibilita a compilação do APH.

7. Metodo de Envio e recebimento de dados via WEB

7.1. Os métodos de envio GET

O método GET é o método que o usuário fica sabendo qual ou quais foram os dados de comunicação entre uma determinada página e outra. O método GET não necessariamente precisa vir de um formulário tendo em vista que os formulários envolvem botões e troca de informações de uma página para outra. Este método não é recomendado para inserção de conteúdo entre informações de um formulário e um banco de dados, tendo em vista que eles não devem ser mostrados ao usuário.

Através do alias virtual HttpGet, podemos consultar se uma determinada propriedade nos foi enviada através da URL
(método GET).

Trata-se portanto de uma propriedade de leitura (read-only), disponível apenas quando a função Advpl é executada através de uma requisição http via link .apw utilizando a configuração de Working Threads WEBEX.

Consultando um Parâmetro

O retorno da consulta de um parâmetro pode ter dois tipos : NIL , caso o parâmetro não tenha sido enviado , ou String , contendo o conteúdo do parâmetro.

Por exemplo, vamos realizar uma requisição a um link .apw , passando pela URL o parâmetro IMAGEM , com o conteúdo TESTE :

`http://localhost/u_TesteGet.apw?imagem=teste&cor=azul`

Para recuperarmos em Advpl o conteúdo dos parâmetros imagem e cor , utilizamos:

```
climagem: = HttpGet->imagem  
cCor: = HttpGet->cor
```

Podemos inserir também um tratamento default : Caso algum parâmetro não seja enviado (resulte NIL) , assumimos um valor para o mesmo

```
DEFAULT climagem: = 'LogoAp8'  
DEFAULT cCor : = 'amarelo'
```

Existe também uma propriedade do alias virtual HttpGet chamada aGets , onde podemos recuperar um array de strings, contendo a lista com os nomes dos parâmetros enviados pelo browser solicitante.

Por exemplo :

```
alInfo:=HttpGet->aGets  
For nl:=1 to len(alInfo)  
  conout('GET'+str(nl,3)+' = '+alInfo[nl])  
Next
```

Exemplo de criação APW

```
#INCLUDE "APWEBEX.CH  
  
User Function EXMETODOGET  
Local cHtml := ""  
  
WEB EXTENDED INIT cHtml  
  
//Exemplo do metodo post.
```



```
If ! Empty(HttpGET->Var1)

// Exibe na Console do Servidor.
  ConOut(HttpGET -> Var1)
  ConOut(HttpGET -> Var2)
Endif
```

```
//Nome do arquivo APH

cHtml := ExecInPage("METODOGET")

WEB EXTENDED END

Return(cHtml)
```

Exemplo de criação APH

```
<html>
<head>
<title>Metodo Get</title>
<body>
<p>Exemplo Metodo GET</p>
<p><a href="u_ EXMETODOGET.apw?Var1=Teste_do_metodo_GET& Var2=ExemploGet">Teste metodo
GET</a></p>

</body>
</html>
```

7.2. Os métodos de envio POST

O método POST é o método que irá transferir suas informações escondidas do link. Este método tem algumas vantagens a mais do que a GET.

1. O número de caracteres enviados chega a ser quase que infinito, em torno de alguns megas de informações.
2. O usuário não saberá qual ou quais foram os dados enviados para a outra página.
3. Este é um método dinâmico, geralmente você, programador, não saberá qual será o valor que irá ser transferido, como por exemplo email, senha e outras informações.

Através do alias virtual HttpPost, podemos consultar os campos submetidos ao servidor através do método POST.

Trata-se portanto de uma propriedade de leitura (read-only), disponível apenas quando a função Advpl é executada através de uma requisição http via link .apw utilizando a configuração de Working Threads WEBEX

Consultando um Parâmetro

O retorno da consulta de um parâmetro pode ter dois tipos : NIL , caso o parâmetro não tenha sido enviado , ou String , contendo o conteúdo do parâmetro. Por exemplo, vamos realizar uma requisição a um link .apw , através de um formulário html com método POST , partindo do Html abaixo :

```
<html><body>
<form method='POST' action='http://localhost/u_TstPost.apw'>
  Teste : <input type='text' name='CODIGO' size='10'>
<hr>
<input type='submit'>
</form>
</body></html>
```

Para recuperarmos em Advpl o conteúdo do campo CODIGO, utilizamos:

```
Codigo := HttpPOST->Codigo
```

cExiste também uma propriedade do alias virtual chamada aPost, onde podemos recuperar um array de strings, contendo a lista com os nomes dos parâmetros enviados pelo browser solicitante.

Por exemplo :

```
alInfo:= HttpPost->aPost
For n1:=1 to len(alInfo)
  conout('POST ' +str(n1,3)+' = '+alInfo[n1])
Next
```

Exemplo de criação APW

```
#Include "APWEBEX.CH"

User Function METODOPOST()

Local cHtml := ""

WEB EXTENDED INIT cHtml

//Exemplo do metodo post.

If ! Empty(HttpPost->Nomevariavel)
```

```

ConOut(HttpPost->Nomevariavel)    // Exibe na Console do Servidor.
Endif

cHtml := ExecInPage("METODOPOST")

WEB EXTENDED END

Return(cHtml)

```

Exemplo de criação APH

```

<html>
<head>
<title>AdvPL/ASP</title>
<body>

<H1> Exemplo AdvPL/ASP - metodo POST </H1>

<form name="Dados " action="u_METODOPOST.apw" method="POST">
<Label>
  Nome: <input type="text" name="Nomevariavel">
</Label>

<input type="SUBMIT" value="Enviar">
</form>

</body>
</html>

```

7.3. Os métodos de envio HTTPSESSION

O alias virtual HttpSession foi criado para possibilitar a criação de variáveis 'session' por usuário do site, com controle de identificação nativa da ferramenta através de um cookie de identificação, chamado SESSIONID. No tópico 'Alias Virtual HttpSession' é explicado em detalhes o funcionamento deste mecanismo.

Este recurso nos permite criar, atribuir conteúdo e consultar conteúdo de uma variável relacionada ao usuário que está realizando uma requisição http. Podemos armazenar em uma variável de Session os seguintes tipos de variáveis: A (array), C (character), D (data), L (lógica) e N (numérica). Não são suportados O (Objetos) e/ou B (Code Blocks).

Limitações de uso dos alias virtuais para recebimento de parâmetros

Dadas as características operacionais e de acesso aos alias virtuais, devemos estar atentos à nomenclatura de campos de um formulário HTML, para serem recuperados com sucesso pelos alias virtuais correspondentes. A nomenclatura de campos do formulário deve obedecer à regra de criação de variáveis em Advpl: O campo do formulário deve sempre ser iniciado com um caracter alfabético, pode conter letras ou algarismos no nome, e o caracter "_" (underline). Não são permitidos espaços, hífen ou caracteres acentuados como nome de um campo. Caso utilizado um nome de campo fora do padrão suportado, o conteúdo do mesmo não será recuperável em Advpl.

Através do alias virtual `httpSession`, podemos criar e consultar variáveis do tipo 'session', relacionadas ao usuário que realizou a requisição através do Browser.

Para diferenciar os usuários que estão navegando num site, o Protheus busca por um cookie identificador de usuário, retornado para o browser a cada requisição de link . APW, chamado SESSIONID. Caso o Protheus receba este cookie, ele identifica quais sessions pertencem a este usuário.

Quando um usuário realiza a primeira requisição http ao Protheus, o Protheus não recebe o cookie identificador , e automaticamente inicializa um identificador de sessions para o mesmo, retornando o identificador ao Browser via Header HTTP.

Este identificador pode ser recuperado em uma função advpl através de

`httpSession->SESSIONID`.

Quando criamos uma variável de session, ela pode ser acessada nas próximas requisições provenientes deste mesmo usuário. Caso uma variável de session consultada não exista, ela retorna o valor NIL (nulo).

Vejamos os exemplos abaixo :

Criando variáveis Session

```
HttpSession->UserId:= '123'  
HttpSession->UserName := cUserName
```

Nas linhas acima , criamos uma session para o usuário atual , chamada UserId , com o conteúdo do tipo String, e criamos outra session chamada UserName , com o retorno da variável

Consultando variáveis Session

Ao consultar uma variável 'session', sempre devemos prever que a mesma não pode ter sido criada, de modo que a consulta pode retornar NIL, ou caso a session já exista , retornará o valor do tipo que foi atribuído `a mesma.

```
If HttpSession->UserId = NIL  
  // Session ainda não foi criada ! Usuário não está logado.  
  conout('Usuario não está logado')  
Else  
  // Session já criada, o usuário está logado  
  conout('Usuario está logado : ID = ' + HttpSession->UserId )  
Endif
```

Exemplo de Funcionamento de Session

No exemplo abaixo, criamos uma session para identificar quantas vezes o usuário chamou esta função específica. Damos o nome da session de MyCounter, que irá conter um número. No primeiro acesso do usuário, a session não existe (= NIL), e é criada com o valor numérico 1 (um).

A partir das próximas requisições realizadas ao Protheus através desta página (através do botão 'Refresh' do Browser, por exemplo) , a session já existe, sendo somado o valor 1 ao conteúdo já existente, e devolvido ao browser solicitante um Html informando quantas chamadas já foram realizadas por este usuário.

```
#include 'Protheus.ch'
#include 'apwebex.ch'

User Function TstSession()
Local cHtml := "", cEcho := ""

WEB EXTENDED INIT cHtml

If httpSession->mycounter = NIL
    cEcho := 'Inicializando contador'
    Conout(cEcho)
    cHtml += cEcho
    httpSession->mycounter := 1
Else
    httpSession->mycounter++
    cEcho := 'contador em '+str(httpSession->mycounter,3)
    conout(cEcho)
Endif

cHtml += cEcho + '<hr>'

WEB EXTENDED END
Return cHtml
```

Após compilado o fonte acima e o Server Protheus configurado e iniciado com HTTP habilitado e as working Threads configuradas, abra um Web Browser e solicite a url http://localhost/u_tstsession.apw . Será mostrado no Browse a mensagem 'Inicializando Contador'.

Agora , peça um 'Refresh' desta tela ao Browser:

Será devolvida a mensagem 'Contador em 2' ... e a cada refresh deste Browser , o contador será incrementado.

Uso de Sessions e Paralelismo - Comportamento do Protheus Server

O Protheus Server trata às requisições simultâneas de links .APW em paralelo, desde que estejam disponíveis o numero de Working Threads necessário para tal. Por exemplo, em uma estrutura de Frames , onde cada um deles aponta o SRC (source) para um link .apw, o Browser envia as três requisições de .apw para o Protheus Server , e caso existam 3 working threads disponíveis naquele momento , as três requisições são atendidas em paralelo.

Por outro lado , se em duas destas três requisições faz-se necessária a atualização e/ou consulta a uma variável de Session (httpsession) , este processamento em paralelo , caso não fosse tratado , poderia gerar perdas no conteúdo da session caso a mesma session fosse atualizada simultaneamente.

Para resolver esta questão, de maneira a não sobrecarregar o Servidor com solicitações de Processamento Sequencial (Critical Sessions) , foi montado um esquema de Lock de Session de Usuário automático, com liberação automática após o processamento do APW, ou liberação manual através da chamada da função HttpLeaveSession() antes do processamento ser terminado.

Exemplificando a aplicação prática e funcionamento deste conceito , partimos de um ambiente hipotético utilizando 3 frames , onde um usuário realiza uma requisição à função que retornará o source HTML da página de frames, e a mesma ao chegar no Browser, faz o mesmo realizar as três requisições simultaneamente, todas elas referentes ao mesmo usuário.

Porém , o primeiro e o segundo frames realizam uma operação qualquer com uma ou mais variáveis da Session do usuário , e o terceiro frame realiza um outro processamento que não depende da consulta de nenhuma variável da Session:

As três requisições referente a este usuário serão processadas simultaneamente por working Threads diferentes (vamos supor que naquele momento haviam três Working Threads disponíveis); porém quando uma das duas working Threads que tentarem acesso à uma variável de Session daquele usuário, o Servidor verifica se alguma outra Thread está com o flag de acesso às sessions deste usuário:

Se nenhuma outra thread em uso por este usuário está com a bandeira, então a thread atual pega a bandeira para ela; senão o processamento da Thread é congelado no aguardo da liberação da bandeira.

A liberação da bandeira ocorre automaticamente no retorno da Working Thread para o Browser , antes da chamada do ponto de entrada para Reset do Ambiente, através da chamada na KlibEx da função HttpLeaveSession().

Caso seja viável para o usuário liberar as sessions antes do retorno da função , ele pode utilizar-se da função httpLeaveSession() no seu fonte , sem necessariamente aguardar pelo encerramento efetivo e reset de ambiente da Working Thread.

Logo , retornando ao exemplo acima , os Frames 1 e 2 irão concorrer pela bandeira de atualização de conteúdo de sessions, onde o primeiro frame que a ser executado pegará a bandeira para ele e atualizará a session , e o segundo frame irá esperar o primeiro liberar a bandeira para continuar a ser processado; e o terceiro frame , como não utiliza nenhuma variável da session, será processado sem depender de nenhum dos outros dois frames anteriores.

Quando utilizamos ASP (Microsoft Active Server Pages), o mesmo realiza uma serialização de requisições de páginas ASP por usuário, de modo que, caso o mesmo usuário solicite três frames .asp, as requisições de processamento chegarão ao Servidor ASP simultaneamente, mas a bandeira de processamento é única por página .asp, sendo liberada apenas após o término do processamento da página, de modo que, mesmo que nenhuma das páginas faça uso de sessions, todas as páginas deste usuário serão processadas em sequência.

Conforme observamos podemos analisar o código gerado para receber o login e senha da página U_TlInWB.apw observar que iremos receber os dados via POST

Exemplo: Fonte desenvolvido em APW

```
#Include "APWEBEX.CH"

User Function METODOSESSION()

Local cHtml := ""

WEB EXTENDED INIT cHtml
//Criação das variáveis de Session
HttpSession->dData := Date()
HttpSession->cHora := Time()

HttpSession->aSemana := {"Domingo", "Segunda", "Terça", ;
    "Quarta", "Quinta", "Sexta", "Sábado"}
```

```
cHtml := H_METODOSESSSION()

WEB EXTENDED END

Return( cHtml )
```

Exemplo: Fonte desenvolvido em APH

```
<head>
<title>AdvPL/ASP</title>
<body>

<p>Data: <%=HttpSession->dData%></p>
<p>Hora: <%=HttpSession->cHora%></p>
<p></p>

<% For i:=1 To Len(HttpSession->aSemana)%>
  <p> <%=HttpSession->aSemana[i]%>
  <% If i == dow(HttpSession->dData) %>
    <===== Hoje
  <%Endif%>
</p>
<%Next%>

</body>
</html>
```

7.4. Os métodos de envio COOKIE

Cookie é um pequeno arquivo de texto no qual um site pode armazenar informações. Cookies são gravados no disco rígido do usuário e não no servidor.

A maioria dos cookies expira (se auto apagam) depois de transcorrido um determinado tempo de pode variar de 1 minuto a vários anos. Contudo o usuário pode identificar e apagar cookies do seu computador a qualquer momento.

A maioria dos navegadores, tais como, Microsoft Internet Explorer, Mozilla Firefox e Google Chrome, podem ser configurados de modo a que o usuário seja previamente avisado da ravação de cookie e dar-lhe a opção de aceitar ou não a gravação. Mas, por que simplesmente não permitir a gravação de cookies?

Bem, isto é possível, contudo muitos sites não funcionarão sem uso dos cookies. Isto porque eles dependem de cookies para melhorar a usabilidade e viabilizar a funcionalidade do site.

É fácil configurar ou modificar um cookie com uso da função do ASP/JavaScript

No exemplo a seguir criaremos um cookie e a ele daremos um valor.

```
function gravaCookie(nome, value, horas){
var expire = "";
if(horas != null){
    expire = new Date((new Date()).getTime() + horas * 3600000);
    expire = "; expires=" + expire.toGMTString();
}
document.cookie = nome + "=" + escape(value) + expire;
}
```

Para gravar

```
onclick="gravaCookie('meusite', 'Curso ADVPL ASP', 24)";
```

Primeiro devemos escolher um nome para o cookie. No nosso exemplo escolhemos o nome "meusite". A seguir definimos um valor para o cookie e por final o tempo em que ele ficara disponível.

Por padrão, um cookie expira quando o navegador é fechado, mas isto pode ser facilmente alterado adicionando-se um parâmetro a mais na função definindo o tempo de vida do cookie:

" new Date((new Date()).getTime() + horas * 3600000);" define que o cookie expira em 3600000 Milissegundo (60 minutos) a partir de agora somando 24 horas.

No exemplo mostrado armazenamos informações sobre o nome do usuário e seus interesses. Estas informações podem ser úteis, por exemplo, para direcionar o usuário para uma seção específica do site.

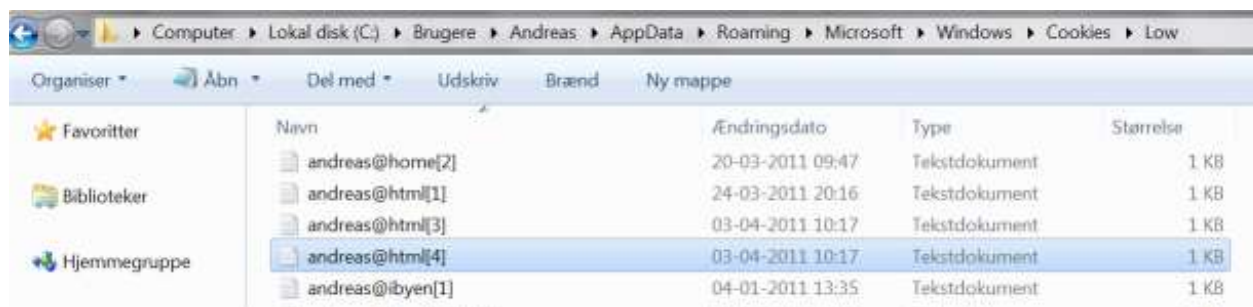
Como recuperar o COOKIE

Para recuperar o valor de um cookie usamos ler COOKIE disponibilizada em qualquer pagina desde que o usuário tenha passado pela pagina a qual gerou o COOKIE

```
function lerCookie(nome)
{
    var valorCookie = "";
    var search = nome + "=";
    if(document.cookie.length > 0) {
        offset = document.cookie.indexOf(search);
        if (offset != -1){
            offset += search.length;
            end = document.cookie.indexOf(";", offset);
            if (end == -1) end = document.cookie.length;
            valorCookie = unescape(document.cookie.substring(offset, end))
        }
    }
    return valorCookie;
}
```

Exemplo do COOKIE

O cookie foi gravado no seu disco rígido. O cookie e gravado em um local do disco rígido que varia com o sistema operacional usado pelo usuário. Uma vez que você consiga localizá-lo o seu formato parecido com o seguinte:



Navn	Ændringsdato	Type	Størrelse
andreas@home[2]	20-03-2011 09:47	Tekstdokument	1 KB
andreas@html[1]	24-03-2011 20:16	Tekstdokument	1 KB
andreas@html[3]	03-04-2011 10:17	Tekstdokument	1 KB
andreas@html[4]	03-04-2011 10:17	Tekstdokument	1 KB
andreas@ibyen[1]	04-01-2011 13:35	Tekstdokument	1 KB

Como você pode ver um cookie é um arquivo de texto que pode ser aberto no Notepad, por exemplo. O conteúdo do cookie que acabamos de criar se parece com o seguinte:

```
HTMLTest TEXT=Curso+ADVPL+ASP 0 80973619229399148 4216577264 29399141 *
```

Não iremos nos aprofundar mais em cookies, contudo note que o usuário detém o controle sobre a gravação de cookies no seu computador.

Mas para o Protheus Através do alias virtual HttpCookies, é possível consultar os Cookies do Header Http enviados pelo Browser, e criar ou alterar o conteúdo de um cookie a ser devolvido ao Browser. Uma variável de Cookie retorna um tipo Advpl String, e apenas aceita uma atribuição de String. Vale lembrar também que um cookie é um recurso do Browser que está realizando a requisição, e existe um limite de tamanho para o total de Cookies utilizados. Este limite costuma ser próximo a 1024 Bytes .

Trata-se portanto de uma propriedade de leitura e gravação, disponível apenas quando a função Advpl é executada através de uma requisição http via link .apw utilizando a configuração de Working Threads WEBEX.

Utilização de Cookies

A utilização de Cookies têm objetivo prático restrito à aplicações onde haja a necessidade implícita de termos uma informação relacionada ao browser utilizado pelo usuário atual, que devam ser interpretadas independente do usuário estar logado ou não (isto é , independam diretamente de sessions).

Um exemplo prático disto é o desenvolvimento de um site onde o conteúdo dinâmico é retornado ao usuário em mais de um idioma. Na entrada do site, apresentamos um formulário ao usuário onde o mesmo irá escolher o idioma de sua preferência. Mesmo que a session de login deste usuário expire no servidor, o cookie com o idioma selecionado ainda está no Browser, de modo que a próxima requisição do usuário pode ser codificada para direcioná-lo para a página de login do site com as mensagens no idioma que o mesmo já estava navegando.

Lendo o valor de um Cookie

Através dos exemplos abaixo , lemos o valor do Cookie de identificação do usuário, e um cookie de usuário criado para identificar no Browse qual é o idioma utilizado pelo usuário atual.

```
cUserId := HttpCookies->USERID // Retorna o Cookie identificador do usuário do Protheus
cIdioma := HttpCookies->SiteLang // Retorna o conteúdo do cookie SiteLang , criado
```

Lendo todos os Cookies recebidos

O alias virtual `HttpCookie` possui uma propriedade chamada **aCookies**, criada apenas para consulta (read-only), que retorna um Array Advpl de Strings , contendo os nomes dos Cookies enviados pelo Browser ao Protheus .

Por exemplo :

```
aInfo:= HttpCookies->aCookies
Fornl:= 1tolen(aInfo)
// Mostra no console do Server todos os cookies recebidos.
conout('Cookie'+str(nl,3)+'=''+aInfo[nl])
Next
```

Criando um Cookie

A criação de um Cookie é realizada através da atribuição de um valor diretamente ao cookie desejado. Por exemplo :

```
HttpCookies->MeuCookie := 'TESTE'
```

A criação de um Cookie merece uma atenção especial, pois um Cookie é retornado ao browser através do Header de Retorno HTTP. De modo que, para que a criação de um cookie seja realizada com sucesso , o mesmo deve ser criado antes de haver qualquer processamento de APH / AHU, caso este que não seria mais possível a criação do Cookie, pois o Header de Retorno HTTP já teria sido enviado ao browser solicitante.

7.5. Os métodos de envio HTTPHEADIN

Para a recepção e tratamento das informações recebidas através do Header do pacote HTTP, foi criado o alias virtual `HttpHeadIn`, que além de consultar as informações constantes no Header HTTP proveniente da requisição do usuário, permite também acesso à propriedades da conexão atual do usuário, como o IP do usuário solicitante.

Através do alias virtual `HttpHeadIn`, podemos consultar os parâmetros enviados pelo Browser solicitante enviados através do Header HTTP ao realizar uma requisição ao Protheus Server.

Trata-se portanto de uma propriedade de leitura (read-only), disponível apenas quando a função Advpl é executada através de uma requisição http via link .apw utilizando a configuração de Working Threads WEBEX.

Consultando um Parâmetro

O retorno da consulta de um parâmetro pode ter dois tipos : NIL , caso o parâmetro não tenha sido enviado , ou String , contendo o conteúdo do parâmetro. Por exemplo, vamos consultar o header http User-Agent , enviado pelo Browser solicitante contendo uma String identificando o modelo de Browser utilizado:

cUserAgent := Httpheadin->User_Agent

Devemos obter como retorno uma string parecida com a mostra abaixo :
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)

Observação Importante:

Qualquer parâmetro no Header HTTP que contenha um ou mais caracteres inválidos para a nomenclatura de variáveis Advpl , (como por exemplo o User-Agent , que contém um hífen), são trocados pelo caractere '_' underline , para ser possível a leitura da propriedade.

Propriedades especiais

HttpHeadIn->aHeaders

Retorna um Array de Strings , contendo todas as linhas do Header HTTP da requisição.

HttpHeadIn->main

Retorna o nome da função chamada através da URL , sem a extensão e sem o host. Por exemplo , ao chamar o link http://localhost/u_tstHeader.apw , o conteúdo de **HttpHeadin->main** será 'u_tstHeader'

HttpHeadIn->REMOTE_ADDR

Retorna uma string , no formato nnn.nnn.nnn.nnn , o IP da estação que realizou a requisição.

httpHeadIn->REMOTE_PORT

Retorna um valor Advpl numérico , informando a porta utilizada para realizar a requisição.

7.6. Retorna um valor Advpl numérico , informando a porta utilizada para realizar a requisição.

Através deste alias virtual de retorno, podemos alterar ou criar um parâmetro no Header de retorno HTTP do Protheus, a ser devolvido ao Browser solicitante de uma requisição de processamento.

Através deste alias virtual de retorno, podemos alterar ou criar um parâmetro no Header de retorno HTTP do Protheus, a ser devolvido ao Browser solicitante de uma requisição de processamento.

Trata-se portanto de uma propriedade de retorno, disponível apenas quando a função Advpl é executada através de uma requisição http via link .apw utilizando a configuração de Working Threads WEBEX.

A criação de uma linha no Header HTTP merece uma atenção especial, pois para que a operação realizada com sucesso, o header deve ser criado antes de haver qualquer processamento de APH / AHU, pois neste caso o Header de Retorno HTTP já teria sido enviado ao browser solicitante.

8. Criação do Potal

8.1. Processamento Para varias Empresas/Filiais

Conforme apresentado anteriormente vimos a criação de um host para todas as empresas e todas as filiais, mas nesse tópico iremos elaborar um método para que o usuário ao entrar na pagina escolha qual será a empresa e Filial que deseja visualizar/alterar os dados.

Para isso iremos analisar o nosso SIGAMAT.EMP e verificar as empresas que devemos apresentar. Após análise de quais empresa ou filiais iremos permitir o acesso na WEB devemos criar um código fonte que permita o usuário escolher a Empresa/Filial e validar o seu acesso em nossa pagina de entrada

Exemplo: Fonte desenvolvido em APW

```
#INCLUDE "Protheus.ch"
#include 'APWEBEX.CH'

User function PORTAL(_cEmp_,_cFil_)
Local aTabs   := {"SA1"}
Local cHtml   := ""
default _cEmp_ := "99"
default _cFil_ := "01"
Private aDados := {}

WEB EXTENDED INIT cHtml
//seta o ambiente com a empresa 99 filial 01 com os direitos do usuario administrador, módulo Financeiro
RpcSetEnv( _cEmp_,_cFil_,, "FIN", "TlInWB", aTabs,,,)

/***** COMANDOS *****/
SM0->(DBGOTOP())
WHILE SM0->(!EOF())
    AADD(aDados,{""+SM0->M0_CODIGO+'-'+SM0->M0_CODFIL+"" ,SM0->M0_NOME})
    SM0->(DBSKIP())
END

// Chamar a pagina
cHtml := h_PORTAL()
```

```
RpcClearEnv() //Limpa o ambiente, liberando a licença e fechando as conexões
WEB EXTENDED END
Return(cHtml)
```

Podemos analisar o código gerado as informações passadas

Variável

aTabs = Array com uma única tabela para que o sistema possa abrir
 cHtml = String com o retorno do html executado pela função h_Posrtal()
 cEmp = String com o Código da empresa para ser aberto
 cFil = String com o código da Filial a ser aberta

aDados= Array com os dados que iremos alimentar ao ler o Sigamat.emp, essa variável foi criada como PRIVADA para fazer a passagem para o fonte h_PORTAL() -> PORTAL.aph

Função

RpcSetEnv

Abertura do ambiente em rotinas automáticas

```
( [ cRpcEmp ] [ cRpcFil ] [ cEnvUser ] [ cEnvPass ] [ cEnvMod ] [ cFunName ]  

[ aTables ] [ IShowFinal ] [ IAbend ] [ IOpenSX ] [ IConnect ] ) --> IRet
```

Nome	Tipo	Descrição	Default
cRpcEmp	Caracter	Código da empresa.	
cRpcFil	Caracter	Código da filial.	
c	Caracter	Nome do usuário.	
cEnvPass	Caracter	Senha do usuário.	
cEnvMod	Caracter	Código do módulo.	'FAT'
cFunName	Caracter	Nome da rotina que será setada para retorno da função FunName().	'RPC'
aTables	Vetor	Array contendo as tabelas a serem abertas.	{}
IShowFinal	Lógico	Alimenta a variável publica IMsFinalAuto.	.F.
IAbend	Lógico	Se .T., gera mensagem de erro ao ocorrer erro ao checar a licença para a estação.	.T.
IOpenSX	Lógico	SE .T. pega a primeira filial do arquivo SM0 quando não passar a filial e realiza a abertura dos SXs.	.T.
IConnect	Lógico	Se .T., faz a abertura da conexão com servidor As400, SQL Server etc.	.T.

RpcClearEnv

Limpa o ambiente, liberando a licença e fechando as conexões

Exemplo: Fonte desenvolvido em APH

```
<HTML>
<style>
div.Center {
margin-top: 30%;

margin-left: 35%;
float: left;
height: 510px;
width: 340px;
border: 1.5px solid #d9d9d9;
background-color: white;
position: relative;
border-radius: 10px;
}
.logo{
background-image: url('_imagens/logo.png');
background-repeat: no-repea
margin-top: 20px;
margin-left: 80px;
height: 60px;
width: 200px;
margin-top: 20px;

}
.bemvindo{
margin-top: 20%;
margin-left: 10%;
font-family: Arial;
font-size: 14px;
Font-family: Arial;
font-size: 15px;
color: #747675;
font-weight: bold;
}

.ID{
position: absolute;
margin-top: 05%;
margin-left: 10%;
font-family: Arial;
font-size: 14px;
color: #747675;
}

.senha{
position: absolute;
margin-top: 20%;
margin-left: 10%;
font-family: Arial;
font-size: 14px;
color: #747675;
}
.botao{
```

```

position: absolute;
margin-top: 50%;
margin-left: 10%;
}
img {
display: block;
margin: 0 auto;
}
input[type=text] {
width: 200px;
display: block;
margin-bottom: 10px;
}

input[type=PASSWORD] {
width: 200px;
display: block;
margin-bottom: 10px;
}
input[type=submit] {
color: #FFFFFF;
width: 100px;
height: 30px;
display: block;
margin-bottom: 10px;
background-color: #1F739E;
border: 1px solid #01669F;
}
</style>

<HEAD>
<TITLE>Pagina Inicial - ADVPL-WEB</TITLE>
</HEAD>

<script language="JavaScript">
function cancelar(){
    document.login.action = "u_ValUser.apw";
    document.login.submit()
}
function ok(){
var ok = true;
missinginfo = "";
if ( document.login.cNome.value == "" ) {
    ok = false;
    missinginfo = "\n    - Nome';
}
if ( document.login.cSenha.value == "" ) {
    ok = false;
    missinginfo += "\n    - Senha';
}
if ( ok ){
    document.login.submit();
}
else{
    missinginfo = '_____'\n' +
        " Não preenchido corretamente:\n" +
        missinginfo + "\n_____' +

```



```

    '\n Tente novamente!';
    alert(missinginfo)
  }
}
</script>

<BODY BACKGROUND="fundo_degrade.png">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Treinamento ADVPL WEB </title>

<div Class="Center">
<div Class="logo"
  
</div>

```

Nesse fonte montamos a pagina com uma seleção para apresentar os dados carregados no fonte APW.

Podemos analisar que criamos um formulário para passar para a próxima pagina utilizando o método "Post"

```
<FORM name="login" id="login" action="u_NOMEDOPRW.apw" METHOD="POST" >
```

Abertura do formulario com o nome Login para tratamento das funções javascript

Ação para qual pagina o site enviara as informações **u_NOMEDOPRW.apw**

método **POST** utilizado para não apresentar na URL

Podemos utilizar funções de JavaScript

```

<script language="JavaScript">
function cancelar(){
    document.login.action = "u_ValUser.apw";
    document.login.submit()
}
function ok(){
var ok = true;
missinginfo = "";
if ( document.login.cNome.value == " " ) {
    ok = false;
    missinginfo = '\n    - Nome';
}
if ( document.login.cSenha.value == " " ) {
    ok = false;
    missinginfo += '\n    - Senha';
}
if ( ok ){
    document.login.submit();
}
else{
    missinginfo ='_____ \n' +
    " Não preenchido corretamente:\n" +
    missinginfo + '\n_____ ' +
    '\n Tente novamente!';
}

```



```
    alert(missinginfo)
  }
}
</script>
```

E na chamada do Botão definimos a chamada da função javascript para fazer a validação desejada em cada campo.

```
<input type="submit" value="Acessar Portal" onClick=javascript:ok()>
```

Podemos utilizar formatação CSS

```
<style>
div.Center {
  margin-top: 30%;
  margin-left: 35%;
  float: left;
  height: 510px;
  width: 340px;
  border: 1.5px solid #d9d9d9;
  background-color: white;
  position: relative;
  border-radius: 10px;
}
.logo{
  background-image: url('_imagens/logo.png');
  background-repeat: no-repeat;
  margin-top: 20px;
  margin-left: 80px;
  height: 60px;
  width: 200px;
  margin-top: 20px;
}
.bemvindo{
  margin-top: 20%;
  margin-left: 10%;
  font-family: Arial;
  font-size: 14px;
  font-family: Arial;
  font-size: 15px;
  color: #747675;
  font-weight: bold;
}
.ID{
  position: absolute;
  margin-top: 05%;
```

```
margin-left: 10%;
font-family: Arial;
font-size: 14px;
color: #747675;
}

.senha{
position: absolute;
margin-top: 20%;
margin-left: 10%;
font-family: Arial;
font-size: 14px;
color: #747675;
}

.botao{
position: absolute;
margin-top: 50%;
margin-left: 10%;
}

img {
display: block;
margin: 0 auto;
}

input[type=text] {
width: 200px;
display: block;
margin-bottom: 10px;
}

input[type=PASSWORD] {
width: 200px;
display: block;
margin-bottom: 10px;
}

input[type=submit] {
color: #FFFFFF;
width: 100px;
height: 30px;
display: block;
margin-bottom: 10px;
background-color: #1F739E;
border: 1px solid #01669F;
}
</style>
```



The image shows a login form for the TOTVS system. It features the TOTVS logo at the top. Below the logo, the text "Seja bem-vindo:" is displayed. There are three input fields: "Usuario:", "Senha:", and "Empresa:". The "Empresa:" field is a dropdown menu with "MATRIZ" selected. Below the input fields is a blue button labeled "Acessar Portal".

Exercício

Desenvolver uma tabela ZZ0 com quatro campos

ZZ0_FILIAL	CHARACTER	02	OBRIGATORIO
ZZ0_USUARI	CHARACTER	10	OBRIGATORIO
ZZ0_NOME	CHARACTER	80	OBRIGATORIO
ZZ0_SENHA	CHARACTER	06	OBRIGATORIO

INDICE

ZZ0_FILIAL+ ZZ0_USUARI

Desenvolver uma pagina que lista as Empresas/Filiais do sigamat.emp e apresente numa pagina.

Nessa pagina deverá conter 2 campos, 2 botões, uma seleção

Campo 1 – Login

Campo 2 – Senha

Botão 1 Entrar

Botão 2 Cancelar

Fazer a validações dos dados da tela.

Obs o método de Envio deve ser feito via POST

8.2. Processo de Gravação de dados Via Páginas da Web

Iniciamos o processo de criação de da opção Visualizar / Incluir / Alterar e Excluir precisamos trabalhar com uma tabela.

Nos exemplos iremos utilizar a tabela SA1 -> Clientes

Para listar os dados do Cliente na pagina é necessario carregar os dados do cliente para alguma estrutura, iremos trabalhar com array

No exemplo iremos varer o metadados SX3 para pegar os campos.

Exemplo do fonte APW carregando a estrutura da tabela de clientes

```
#INCLUDE "TOTVS.CH"
#INCLUDE "APWEBEX.CH"
#INCLUDE "TBICONN.CH"

User Function Listadados()

Local cHtml := ""
Local cAlias := "SA1"
Local aDados := {}
Local aHeader := {}

If Select("SX2") == 0
  Prepare Environment Empresa '99' Filial '01' Tables 'SA1'
EndIf

WEB EXTENDED INIT cHtml
HTTPSESSION->aHeader := {}
HTTPSESSION->aCliente := {}

  dbSelectArea("SX3")
  dbSetOrder(1)
  dbSeek(cAlias)

aAdd( aHeader, { " ", "( Recno() )" } )

  While ! SX3->(EOF()) .And. SX3->X3_ARQUIVO == cAlias
    If X3Uso(SX3->X3_USADO) .And. cNivel >= SX3->X3_NIVEL .And. X3Obrigat(SX3->X3_CAMPO)
      AADD( aHeader, { Trim( X3Titulo() ), SX3->X3_CAMPO } )
    Endif
    SX3->(dbSkip())
  EndDo

  dbSelectArea(cAlias)
  (cAlias)->( dbSetOrder(1) )
  (cAlias)->( dbGoTop() )

  While ! (cAlias)->( EOF() )

    AADD( aDados, Array( Len( aHeader ) ) )
```

```

        For x := 1 To Len(aHeader)

            aDados[Len(aDados),x] := &(amp;cAlias + "->" + aHeader[x,2])

        Next x
        (cAlias)->(dbSkip())

    EndDo

    HTTPSESSION->aHeader := aHeader
    HTTPSESSION->aDados := aDados

    cHtml := H_ListaDados()

WEB EXTENDED END

Return( cHtml)

```

Podemos analisar como foi capturada a tabela

cAlias := "SA1"

Onde essa será a tabela do browse para ser apresentada.

Para isso o código utiliza as funções padrões de pesquisa do SX3 da empresa que esta logado e alimenta a variável aHeader, somente com os campos que estão com X3Uso(SX3->X3_USADO) e X3Obrigat(SX3->X3_CAMPO), poderia carregar a estrutura pelo campo SX3->X3_BROWSE == 'S'

Campos que devem ser apresentados no Browse

Logo após ter carregado os campos ele já alimenta uma sessão para uso futuro com os dados da variável Aheader

HttpSession->aHeader := aHeader

Agora o código fonte abre a batela a qual foi capturada pela sessão cAlias e alimenta a outra variável aDados com os dados encontrados

Iremos criar uma pagina em HTML para interpretar os dados.

Exemplo do fonte APH

```

<% Local x ; Local Y %>
<!DOCTYPE html>
<html>
<head>
<style>

table {
    font-family: arial, sans-serif;
    border-collapse: collapse;
    width: 100%;
}

```

```

td {
border: 1px solid RGB(119,119,119);
text-align: left;
padding: 8px;
}

th {
border: 1px solid RGB(151,151,151);
text-align: center;
padding: 8px;
color: RGB(255,255,255);
background-color: RGB(115,115,115);
}

tr:nth-child(even) {
background-color: RGB(109,217,255);
}

div.tabela {
position: relative;
margin-top: 160px;
Top: 50%;
border-radius: 10px;
}

</style>
</head>
<body>
<BODY >

<Div Class="tabela" >

<form name="form1" action="u_xAltera.apw" method="POST">

    <table>
        <tr>

            <% For x:= 1 To Len(HTTPSESSION->aHeader) %>
                <TH> <%= HTTPSESSION->aHeader[x][1] %> </TH>
            <% Next x %>

        </tr>

        <% For Y:= 1 To Len(HTTPSESSION->aDados) %>
            <tr>

                <% For x := 1 To Len(HTTPSESSION->aHeader) %>
                    <% if x = 1 %>
                        <TD> <input type="radio" name="recno"
value=<%= HTTPSESSION->aDados[Y][X] %> > </TD>
                    <% Else %>
                        <TD> <%= HTTPSESSION->aDados[Y][X] %>
                    <% Endif %>
                <% Next x %>
            </tr>
        <% Next Y %>
    </table>
</form>

```

```

        </tr>
        <% Next Y %>

    </table>
</Div>
<input type="submit">
</body>
</html>

```

Na criação do HTML possui um campo Radio com o numero do RECNO da tabela para manipular o registro posicionado

```
<input type="radio" name="recno" value=<%= HTTPSESSION->aDados[Y][X] %> />
```

Pronto a principio a pagina do Mbrowse/Axcadastro esta criada agora devemos criar as telas dos respectivos botões

Pesquisar, Visualizar, Alterar, Incluir, Excluir que já definimos que o Pop-up ira abri-las.

Para realizar a edição dos dados, devemos criar um formulário para edição.

Conforme a rotina ira chamar a função/pagina u_xtelaA.apw iremos cria-la!

xtelaA.apw

```

#include "protheus.ch"
#include "apwebex.ch"

user function xtelaA()
Local cHtml      := ""
Local xRecno     := HTTPGET->xRecnos
Local xTab       := HttpSession->cTab
Local cNome      := ""
Local nVolta     := 0
Local cFolder    := ""
Private aDados1  := {}
Private aDados2 := {}
pPRIVATE INCLUI := .F.
pPRIVATE ALTERA := .T.
pPRIVATE DELETA := .F.
Default xRecno  := ""
WEB EXTENDED INIT cHtml
CONOUT('xRecno')
CONOUT(xRecno)

if empty(xRecno) .or. upper(xRecno)='UNDEFINED'
    //cHtml := h_xtelaQ()
    cHtml := " <html> "

```

```

cHtml += " <body> "
cHtml += " <form name='login' method='post' action='u_TIIniWB3.apw'> "
cHtml += " <script language='JavaScript'><INPUT TYPE='hidden' VALUE=xRecno
NAME='xRecnos'></script> "
cHtml += " </form> "
cHtml += " </body> "
cHtml += " </html> "
cHtml += " <script language='JavaScript'> "
cHtml += " document.login.submit(); "
cHtml += " </script> "
else
HttpSession->xRecno := val(xRecno)
//UAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA□□¿
//³processo para pegar o registro³
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA□□Ù
SX3->(DBSETORDER(4))
SX3->(DBSEEK(xTab))
(xTab)->(dbgoto(val(xRecno)))
cFolder := "x"
While SX3->X3_ARQUIVO == (xTab)
    nVolta++
    IF EMPTY(SX3->X3_FOLDER)
        if !(SX3->X3_FOLDER $ cFolder)
            IF SXA->(DBSEEK(SX3->X3_ARQUIVO+SX3->X3_FOLDER))
                cNomFolder := alltrim(SXA->XA_DESCRIC)
            ELSE
                cNomFolder := "Outros"
            ENDIF
            aadd(aDados1,'<TR><TD COLSPAN="4"
BGCOLOR="#A9C6CF">'+cNomFolder+'<HR WIDTH="100%" SIZE="10" ALIGN="Center"
COLOR="#a8d1f2"></TD> </TR>')
            cFolder += SX3->X3_FOLDER
        endif
        if nVolta = 1
            aadd(aDados1,'<TR>')
        endif

        IF EMPTY(SX3->X3_VISUAL) .OR. SX3->X3_VISUAL != 'V' .OR. SX3-
>X3_CONTEXT != 'V'
            aadd(aDados1,'<TD>'+SX3->X3_TITULO+'</TD>')
            xVar := IIF(VALTYPE(&((xTab)+'->'+SX3-
>X3_CAMPO))=="N",cValtoChar(&((xTab)+'->'+SX3-
>X3_CAMPO)),IIF(VALTYPE(&((xTab)+'->'+SX3->X3_CAMPO))=="D",DTOC(&((xTab)+'-
>'+SX3->X3_CAMPO)),IIF(VALTYPE(&((xTab)+'->'+SX3->X3_CAMPO))=="U",
",&((xTab)+'->'+SX3->X3_CAMPO))))
            if VALTYPE(&((xTab)+'->'+SX3->X3_CAMPO)) == "N"
                aadd(aDados1,'<TD><INPUT TYPE="text" VALUE="'+xVar+'"'
NAME="'+alltrim(SX3->X3_CAMPO)+'" id="'+alltrim(SX3->X3_CAMPO)+'"
SIZE="'+cValtoChar(SX3->X3_TAMANHO)+'" MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO)+'" onkeypress='return SomenteNumero(event);'></TD>')
            elseif VALTYPE(&((xTab)+'->'+SX3->X3_CAMPO)) == "D"
                aadd(aDados1,'<TD><INPUT TYPE="text" VALUE="'+xVar+'"'
NAME="'+alltrim(SX3->X3_CAMPO)+'" id="'+alltrim(SX3->X3_CAMPO)+'"

```



```

SIZE="" + cValtoChar(SX3->X3_TAMANHO+2) + "" MAXLENGTH="" + cValtoChar(SX3-
>X3_TAMANHO+2) + "" onchange="validaDat(this,this.value)" ></TD>')
    else
        aadd(aDados1,'<TD><INPUT TYPE="text" VALUE="" + xVar + ""
NAME="" + alltrim(SX3->X3_CAMPO) + "" id="" + alltrim(SX3->X3_CAMPO) + ""
SIZE="" + cValtoChar(SX3->X3_TAMANHO) + "" MAXLENGTH="" + cValtoChar(SX3-
>X3_TAMANHO) + "" onchange="upperCase(this,this.value)" ></TD>')
    endif
ELSE
    aadd(aDados1,'<TD>' + SX3->X3_TITULO + '</TD>')
    xVar := IIF(VALTYPE(&(SX3-
>X3_RELACAO))='N',cValtoChar(&(SX3->X3_RELACAO)),IIF(VALTYPE(&(SX3-
>X3_RELACAO))='D',dtoc(&(SX3->X3_RELACAO)),IIF(VALTYPE(&(SX3-
>X3_RELACAO))='U'," ",&(SX3->X3_RELACAO))))
    if VALTYPE(&(SX3->X3_RELACAO))='N'
        aadd(aDados1,'<TD><INPUT TYPE="text" VALUE="" + xVar + ""
NAME="" + alltrim(SX3->X3_CAMPO) + "" id="" + alltrim(SX3->X3_CAMPO) + ""
SIZE="" + cValtoChar(SX3->X3_TAMANHO) + "" MAXLENGTH="" + cValtoChar(SX3-
>X3_TAMANHO) + "" + "" onkeypress='return SomenteNumero(event);' READONLY DISABLED
></TD>')
    elseif VALTYPE(&(SX3->X3_RELACAO))='D'
        aadd(aDados1,'<TD><INPUT TYPE="text" VALUE="" + xVar + ""
NAME="" + alltrim(SX3->X3_CAMPO) + "" id="" + alltrim(SX3->X3_CAMPO) + ""
SIZE="" + cValtoChar(SX3->X3_TAMANHO+2) + "" MAXLENGTH="" + cValtoChar(SX3-
>X3_TAMANHO+2) + "" onchange="validaDat(this,this.value)" READONLY DISABLED
></TD>')
    else
        aadd(aDados1,'<TD><INPUT TYPE="text" VALUE="" + xVar + ""
NAME="" + alltrim(SX3->X3_CAMPO) + "" id="" + alltrim(SX3->X3_CAMPO) + ""
SIZE="" + cValtoChar(SX3->X3_TAMANHO) + "" MAXLENGTH="" + cValtoChar(SX3-
>X3_TAMANHO) + "" onchange="upperCase(this,this.value)" READONLY DISABLED ></TD>')
    endif
ENDIF
if nVolta = 2
    aadd(aDados1,'</TR>')
    nVolta := 0
endif
ELSE
    if !(SX3->X3_FOLDER $ cFolder)
        IF SXA->(DBSEEK(SX3->X3_ARQUIVO+SX3->X3_FOLDER))
            cNomFolder := alltrim(SXA->XA_DESCRIC)
        ELSE
            cNomFolder := "Outros"
        ENDIF
        aadd(aDados2,'<TR><TD COLSPAN="4"
BGCOLOR="#A9C6CF">' + cNomFolder + '<HR WIDTH="100%" SIZE="10" ALIGN="Center"
COLOR="#a8d1f2"></TD> </TR>')
        cFolder += SX3->X3_FOLDER
    endif
    if nVolta = 1
        aadd(aDados2,'<TR>')
    endif
    onblur="ValIntegerCHR(this.value,this)"

```

```

IF EMPTY(SX3->X3_VISUAL) .OR. SX3->X3_VISUAL != 'V' .OR. SX3-
>X3_CONTEXT != 'V'
    aadd(aDados2,'<TD>'+SX3->X3_TITULO+'</TD>')
    xVar := IIF(VALTYPE(&((xTab)+'->'+SX3-
>X3_CAMPO))=="N",cValtoChar(&((xTab)+'->'+SX3-
>X3_CAMPO)),IIF(VALTYPE(&((xTab)+'->'+SX3->X3_CAMPO))=="D",DTC(&((xTab)+'-
>'+SX3->X3_CAMPO)),IIF(VALTYPE(&((xTab)+'->'+SX3->X3_CAMPO))=="U",
",&((xTab)+'->'+SX3->X3_CAMPO))))
    if valtype(&((xTab)+'->'+SX3->X3_CAMPO)) == "N"
        aadd(aDados2,'<TD><INPUT TYPE="text" VALUE="'+xVar+"
NAME="'+alltrim(SX3->X3_CAMPO)+" id="'+alltrim(SX3->X3_CAMPO)+"
SIZE="'+cValtoChar(SX3->X3_TAMANHO)+" MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO)+" onkeypress='return SomenteNumero(event);'></TD>')
        elseif valtype(&((xTab)+'->'+SX3->X3_CAMPO)) == "D"
            aadd(aDados2,'<TD><INPUT TYPE="text" VALUE="'+xVar+"
NAME="'+alltrim(SX3->X3_CAMPO)+" id="'+alltrim(SX3->X3_CAMPO)+"
SIZE="'+cValtoChar(SX3->X3_TAMANHO+2)+" MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO+2)+" onchange='validaDat(this,this.value)'></TD>')
        else
            aadd(aDados2,'<TD><INPUT TYPE="text" VALUE="'+xVar+"
NAME="'+alltrim(SX3->X3_CAMPO)+" id="'+alltrim(SX3->X3_CAMPO)+"
SIZE="'+cValtoChar(SX3->X3_TAMANHO)+" MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO)+" onchange='upperCase(this,this.value)'></TD>')
        endif
    ELSE
        aadd(aDados2,'<TD>'+SX3->X3_TITULO+'</TD>')
        xVar := IIF(VALTYPE(&(SX3-
>X3_RELACAO))=='N',cValtoChar(&(SX3->X3_RELACAO)),IIF(VALTYPE(&(SX3-
>X3_RELACAO))=='D',dtoc(&(SX3->X3_RELACAO)),IIF(VALTYPE(&(SX3-
>X3_RELACAO))=='U',"",&(SX3->X3_RELACAO))))
        if VALTYPE(xVar) == "N"
            aadd(aDados2,'<TD><INPUT TYPE="text" VALUE="'+xVar+"
NAME="'+alltrim(SX3->X3_CAMPO)+" id="'+alltrim(SX3->X3_CAMPO)+"
SIZE="'+cValtoChar(SX3->X3_TAMANHO)+" MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO)+" onkeypress='return SomenteNumero(event);' READONLY DISABLED
></TD>')
            elseif VALTYPE(xVar) == "D"
                aadd(aDados2,'<TD><INPUT TYPE="text" VALUE="'+xVar+"
NAME="'+alltrim(SX3->X3_CAMPO)+" id="'+alltrim(SX3->X3_CAMPO)+"
SIZE="'+cValtoChar(SX3->X3_TAMANHO+2)+" MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO+2)+" onchange='validaDat(this,this.value)' READONLY DISABLED
></TD>')
            else
                aadd(aDados2,'<TD><INPUT TYPE="text" VALUE="'+xVar+"
NAME="'+alltrim(SX3->X3_CAMPO)+" id="'+alltrim(SX3->X3_CAMPO)+"
SIZE="'+cValtoChar(SX3->X3_TAMANHO)+" MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO)+" onchange='upperCase(this,this.value)' READONLY DISABLED ></TD>')
            endif
        ENDIF
    if nVolta = 2
        aadd(aDados2,'</TR>')
        nVolta := 0
    endif
ENDIF

```

```

        SX3->(dbskip())
    END
    cHtml := h_xtelaA()
endif
WEB EXTENDED END

```

return(cHtml)

Nesse fonte utilizamos dois processos de advpl-web e Html dentro do fonte apw

A principio capturamos a variável do RECNO que iremos trabalhar:

Local xRecno := HTTPGET->xRecnos

Essa informação veio no campo URL, pois foi passada pelo método GET

xTab := HttpSession->cTab

Capturamos de qual tabela se trata esse **RECNO**

Logo a seguir temos uma analise se o xRecno vir vaziu ou indefinido pelo HTML

```

if empty(xRecno) .or. upper(xRecno)=='UNDEFINED'

```

Caso ele entrar nessa condição foi criada a variavel cHtml que ira receber o código HTML em formato String para ser executada no Return da Função/Pagina

```

cHtml := " <html> "
cHtml += " <body> "
cHtml += " <form name='login' method='post' action='u_TIIniWB3.apw'> "
cHtml += "   <script language='JavaScript'><INPUT TYPE='hidden' VALUE=xRecno
NAME='xRecnos'></script> "
cHtml += " </form> "
cHtml += " </body> "
cHtml += " </html> "
cHtml += " <script language='JavaScript'> "
cHtml += " document.login.submit(); "

```

cHtml += " </script> "

Case venha preenchido a variável xRecno começa a processar todos os campos

Em primeiro momento é adicionada uma sessão do recno processado

E em seguida um loop que ira ler todos os campos reais/virtuais da tabela xTab no dicionario de Dados (SX3).

Analisando em cada campo do SX3 sua respectiva pasta (Folder) do cadastro

Para isso demonstraremos o código HTML encapsulado no ADVPL como String, para ser executado posteriormente na Pagina

```
<TR>
<TD>Codigo      </TD>
<TD><INPUT      TYPE="text"    VALUE="000001"    NAME="A1_COD"    id="A1_COD"
SIZE="6" MAXLENGTH="6" onchange="upperCase(this,this.value)" ></TD>
<TD>Loja        </TD>
<TD><INPUT      TYPE="text"    VALUE="01"        NAME="A1_LOJA"    id="A1_LOJA"
SIZE="2" MAXLENGTH="2" onchange="upperCase(this,this.value)" ></TD>
</TR>
```

A informação será criada dessa forma apresentada será feita uma tabela com n Linhas e 4 colunas por linha.

Onde na primeira coluna será o nome do campo e a segunda será o input da informação existente do campo na tabela.

E em cada linha ira possuir 2 campos de input e de string

Para entendermos melhor cada valor de um input iremos citar

Função que ira trocar o Action ="pagina definida para o envio das informações" e executa a pagina

Ja na pagina existe existe um loop que lista toda a variavel *aDados2* e *aDados1* para atribuir os campos criados na função *xTelaA.apw*

Quando clicamos no botão cancelar a pagina ira executar a função javascript cancelar que executa o fechamento da janela

Quando clicamos no botão Gravar a pagina ira executar a função javascript ok que executa o a chamada da função/pagina *u_gravar.apw*

```
#include "Protheus.ch"
#include "apwebex.ch"

user function gravar()
Local cHtml := ""
Local aDados := {}
Local aDados1 := {}
Local aParms := {}
Local xRecno := HttpSession->xRecno
Local xTab := HttpSession->cTab
WEB EXTENDED INIT cHtml
// a linha abaixo monta um array com os parametros recebidos via post
// no formato ( [1] = nome / [2] conteudo )
aEval( HttpPost->aPost , { |x| iif(x=='SALVAR',NIL,aadd(aParms,{ x , &("HttpPost->"+x)} ) ) })
// TRATAMENTO DOS CAMPOS VIRTUAIS
DBSELECTAREA("SX3")
SX3->(dbsetorder(1))
SX3->(DBSEEK(xTab))
```

```

While SX3->X3_ARQUIVO ==(xTab)
  CONOUT('ENTREI')
  CONOUT('SX3->X3_CAMPO')
  CONOUT(SX3->X3_CAMPO)
  CONOUT('SX3->X3_CONTEXT')
  CONOUT(SX3->X3_CONTEXT)
  IF EMPTY(X3_CONTEXT) .OR. X3_CONTEXT == 'R'
    AADD(aDados,SX3->X3_CAMPO)
    AADD(aDados1,SX3->X3_TIPO)
  endif
  SX3->(dbskip())
end

//Varinfo('aParms',aParms)
//Varinfo('aDados',aDados)
conout(xTab)
conout(xRecno)
// agora gravar os dados
(xTab)->(dbgoto(xRecno))
Reclock((xTab),f.)
aEval(aParms,{|x|      (NN      :=      aScan(aDados,x[1]),      IIF(NN=0,nil,(&(((xTab)+'->'+x[1])))      :=
IIF(aDados1[NN]=='N',VAL(x[2]),IIF(aDados1[NN]=='D',CTOD(x[2]),x[2]))))})
Msunlock()

cHtml := " <html> "
cHtml += " <body> "
cHtml += " <form name='login' method='post' action='u_TELA.apw'> "
cHtml += " <script language='JavaScript'><INPUT TYPE='hidden' VALUE=xRecno NAME='xRecnos'></script> "
cHtml += " </form> "
cHtml += " </body> "
cHtml += " </html> "
cHtml += " <script language='JavaScript'> "
cHtml += " window.opener.parent.direita.location.reload();window.close() "
cHtml += " </script> "
WEB EXTENDED END
Return(cHtml)

```

Observamos o código acima citado temos a captura do recno alterado pela sessão

```
xRecno := HttpSession->xRecno
```

Da tabela que a qual refere-se o Recno

```
xTab := HttpSession->cTab
```

Já que obtivemos N campos apresentados na tela anterior e não sabemos o nome de todos utilizamos a regra do **aPost**

```
aEval( HttpPost->aPost,{|x| iif(x=='SALVAR',NIL,aadd(aParms,{ x , &("HttpPost->"+x)} ) )})
```

Onde transforma todas as variaveis enviadas num outro array chamado aParms

aParms[n][1] – nome da variável
aParms[n][2] – conteúdo da variável

a gravação é efetuada no block abaixo:

```
Reclock((xTab),.f.)
aEval(aParms,{|x|      (NN      :=      aScan(aDados,x[1]),      IIF(NN=0,nil,(&((xTab)+'->'+x[1]))      :=
IIF(aDados1[NN]='N',VAL(x[2]),IIF(aDados1[NN]='D',CTOD(x[2]),x[2]))))})
Munlock()
```

Onde executa a leitura do array aParms e a gravação do campo na tabela informada.

Logo após a gravação é adicionada na variável cHtml em formato string o processo de carregar a página TlInWB3.apw

Exercício

1. Desenvolver um fonte que lista dos dados do cliente
2. Desenvolver uma página para aparecer os dados do cliente
3. Desenvolver uma rotina automática de gravação

9. Página Modelo 2/ modelo 3

Iremos apresentar a criação de um modelo 3 onde definiremos na parte superior a qual chamamos no Protheus de Enchoice a tabela SA1 o cadastro do Cliente e na parte de baixo da página a qual chamamos no Protheus GetDados utilizaremos um frameset da tabela SE1 títulos do cliente a Receber.

Iniciamos a análise referente a criação dos dados a serem apresentados no modelo3.

Lembrando que para isso devemos utilizar na página aph o modelo Frameset com 3 linhas:

- Linha 1 com os botões desejados
- Linha 2 com a enchoice
- Linha 3 com a GetDados

xModel3.apw

```
#include "Protheus.ch"
#include "ApWebex.ch"

User function xModel3()
```



```

Local cHtml := ""
Local xRecno      := HTTPGET->xRecnos
Local cNome       := ""
Local nVolta      := 0
Local cFolder     := ""
Local cQuery      := ""
Local nVolta      := 0
Local cTabS       := "SA1"
Local cTabI       := "SE1"
Private aDados0   := {}
Private aDados1 := {}
Private aDados2   := {}
Private aFrame1 := {}
pPRIVATE INCLUI := .F.
pPRIVATE ALTERA := .T.
pPRIVATE DELETA := .F.
Private xTab      := "SA1"
Default xRecno    := ""

HttpSession->cTabS := "SA1"
HttpSession->cTabI := "SE1"
HttpSession->xRecno := val(xRecno)
xRecno := ALLTRIM(xRecno)
WEB EXTENDED INIT cHtml

SX3->(DBSETORDER(4))
SX3->(DBSEEK(xTab))
(xTab->(dbgoto(val(xRecno))))
cFolder := "x"

// alimenta o Enchoice

While SX3->X3_ARQUIVO == (xTab)
  nVolta++
  IF EMPTY(SX3->X3_FOLDER)
    if !(SX3->X3_FOLDER $ cFolder)
      IF SXA->(DBSEEK(SX3->X3_ARQUIVO+SX3->X3_FOLDER))
        cNomFolder := alltrim(SXA->XA_DESCRIC)
      ELSE
        cNomFolder := "Outros"
      ENDIF
      aadd(aDados1,'<TR><TD COLSPAN="4" BGCOLOR="#A9C6CF">'+cNomFolder+'<HR
WIDTH="100%" SIZE="10" ALIGN="Center" COLOR="#a8d1f2"></TD> </TR>')
      cFolder += SX3->X3_FOLDER
    endif
    if nVolta = 1
      aadd(aDados1,'<TR>')
    endif

    IF EMPTY(SX3->X3_VISUAL) .OR. SX3->X3_VISUAL != 'V' .OR. SX3->X3_CONTEXT != 'V'
      aadd(aDados1,'<TD>'+SX3->X3_TITULO+'</TD>')
    
```



```

xVar := IIF(VALTYPE(&((xTab)+'-'>+SX3->X3_CAMPO))=="N",cValtoChar(&((xTab)+'-'>+SX3-
>X3_CAMPO)),IIF(VALTYPE(&((xTab)+'-'>+SX3->X3_CAMPO))=="D",DIOC(&((xTab)+'-'>+SX3-
>X3_CAMPO)),IIF(VALTYPE(&((xTab)+'-'>+SX3->X3_CAMPO))=="U","",&((xTab)+'-'>+SX3->X3_CAMPO))))
if VALTYPE(&((xTab)+'-'>+SX3->X3_CAMPO)) == "N"
    aadd(aDados1,'<TD><INPUT                TYPE="text"                VALUE="+xVar+"
NAME="_'+cTabS+'_' +cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+" id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+" SIZE="+cValtoChar(SX3->X3_TAMANHO)+" MAXLENGTH="+cValtoChar(SX3-
>X3_TAMANHO)+" onkeypress='return SomenteNumero(event);' onchange='passar(this)'></TD>')
elseif VALTYPE(&((xTab)+'-'>+SX3->X3_CAMPO)) == "D"
    aadd(aDados1,'<TD><INPUT                TYPE="text"                VALUE="+xVar+"
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+" id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+" SIZE="+cValtoChar(SX3->X3_TAMANHO+2)+" MAXLENGTH="+cValtoChar(SX3-
>X3_TAMANHO+2)+" onchange="validaDat(this,this.value);passar(this)" ></TD>')
else
    aadd(aDados1,'<TD><INPUT                TYPE="text"                VALUE="+xVar+"
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+" id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+" SIZE="+cValtoChar(SX3->X3_TAMANHO)+" MAXLENGTH="+cValtoChar(SX3-
>X3_TAMANHO)+" onchange="upperCase(this,this.value);passar(this)" ></TD>')
endif
ELSE
    aadd(aDados1,'<TD>'+SX3->X3_TITULO+'</TD>')
    xVar := IIF(VALTYPE(&(SX3->X3_RELACAO))=="N",cValtoChar(&(SX3-
>X3_RELACAO)),IIF(VALTYPE(&(SX3->X3_RELACAO))=="D",dtoc(&(SX3->X3_RELACAO)),IIF(VALTYPE(&(SX3-
>X3_RELACAO))=="U","",&(SX3->X3_RELACAO))))
    if VALTYPE(&(SX3->X3_RELACAO))=="N"
        aadd(aDados1,'<TD><INPUT                TYPE="text"                VALUE="+xVar+"
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+" id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+" SIZE="+cValtoChar(SX3->X3_TAMANHO)+" MAXLENGTH="+cValtoChar(SX3-
>X3_TAMANHO)+" onkeypress='return SomenteNumero(event);' onchange='passar(this)' READONLY DISABLED
></TD>')
    elseif VALTYPE(&(SX3->X3_RELACAO))=="D"
        aadd(aDados1,'<TD><INPUT                TYPE="text"                VALUE="+xVar+"
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+" id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+" SIZE="+cValtoChar(SX3->X3_TAMANHO+2)+" MAXLENGTH="+cValtoChar(SX3-
>X3_TAMANHO+2)+" onchange="validaDat(this,this.value);passar(this)" READONLY DISABLED ></TD>')
    else
        aadd(aDados1,'<TD><INPUT                TYPE="text"                VALUE="+xVar+"
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+" id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+" SIZE="+cValtoChar(SX3->X3_TAMANHO)+" MAXLENGTH="+cValtoChar(SX3-
>X3_TAMANHO)+" onchange="upperCase(this,this.value);passar(this)" READONLY DISABLED ></TD>')
    endif
ENDIF
if nVolta = 2
    aadd(aDados1,'</TR>')
    nVolta := 0
endif
ELSE
    if !(SX3->X3_FOLDER $ cFolder)
        IF SXA->(DBSEEK(SX3->X3_ARQUIVO+SX3->X3_FOLDER))
            cNomFolder := alltrim(SXA->XA_DESCRIC)
        ELSE
            cNomFolder := "Outros"
        ENDIF
    ENDIF

```

```

        ENDIF
        aadd(aDados2,'<TR><TD    COLSPAN="4"    BGCOLOR="#A9C6CF">'+cNomFolder+'<HR
WIDTH="100%" SIZE="10" ALIGN="Center" COLOR="#a8d1f2"></TD>    </TR>')
        cFolder += SX3->X3_FOLDER
    endif
    if nVolta = 1
        aadd(aDados2,'<TR>')
    endif
    onblur="ValIntegerCHR(this.value,this)"
    IF EMPTY(SX3->X3_VISUAL) .OR. SX3->X3_VISUAL != 'V' .OR. SX3->X3_CONTEXT != 'V'
        aadd(aDados2,'<TD>'+SX3->X3_TITULO+'</TD>')
        xVar := IIF(VALTYPE(&((xTab)+'->'+SX3->X3_CAMPO))=="N",cValtoChar(&((xTab)+'->'+SX3-
>X3_CAMPO)),IIF(VALTYPE(&((xTab)+'->'+SX3->X3_CAMPO))=="D",DTC(&((xTab)+'->'+SX3-
>X3_CAMPO)),IIF(VALTYPE(&((xTab)+'->'+SX3->X3_CAMPO))=="U",",",&((xTab)+'->'+SX3->X3_CAMPO))))
        if valtype(&((xTab)+'->'+SX3->X3_CAMPO)) == "N"
            aadd(aDados2,'<TD><INPUT                TYPE="text"                VALUE="'+xVar+'"'
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+'"                id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+'"                SIZE="'+cValtoChar(SX3->X3_TAMANHO)+'"                MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO)+'"' + onkeypress='return SomenteNumero(event);' onchange='passar(this)'></TD>')
            elseif valtype(&((xTab)+'->'+SX3->X3_CAMPO)) == "D"
                aadd(aDados2,'<TD><INPUT                TYPE="text"                VALUE="'+xVar+'"'
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+'"                id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+'"                SIZE="'+cValtoChar(SX3->X3_TAMANHO+2)+'"                MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO+2)+'"' onchange="validaDat(this,this.value);passar(this)"></TD>')
            else
                aadd(aDados2,'<TD><INPUT                TYPE="text"                VALUE="'+xVar+'"'
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+'"                id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+'"                SIZE="'+cValtoChar(SX3->X3_TAMANHO)+'"                MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO)+'"' onchange="upperCase(this,this.value);passar(this)"></TD>')
            endif
        ELSE
            aadd(aDados2,'<TD>'+SX3->X3_TITULO+'</TD>')
            xVar := IIF(VALTYPE(&(SX3->X3_RELACAO))=="N",cValtoChar(&(SX3-
>X3_RELACAO)),IIF(VALTYPE(&(SX3->X3_RELACAO))=="D",dtoc(&(SX3->X3_RELACAO)),IIF(VALTYPE(&(SX3-
>X3_RELACAO))=="U",",",&(SX3->X3_RELACAO))))
            if VALTYPE(xVar) == "N"
                aadd(aDados2,'<TD><INPUT                TYPE="text"                VALUE="'+xVar+'"'
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+'"                id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+'"                SIZE="'+cValtoChar(SX3->X3_TAMANHO)+'"                MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO)+'"' + onkeypress='return SomenteNumero(event);' onchange='passar(this)' READONLY DISABLED
></TD>')
            elseif VALTYPE(xVar) == "D"
                aadd(aDados2,'<TD><INPUT                TYPE="text"                VALUE="'+xVar+'"'
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+'"                id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+'"                SIZE="'+cValtoChar(SX3->X3_TAMANHO+2)+'"                MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO+2)+'"' onchange="validaDat(this,this.value);passar(this)" READONLY DISABLED ></TD>')
            else
                aadd(aDados2,'<TD><INPUT                TYPE="text"                VALUE="'+xVar+'"'
NAME="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3->X3_CAMPO)+'"                id="_'+cTabS+'_' +xRecno+'_' +alltrim(SX3-
>X3_CAMPO)+'"                SIZE="'+cValtoChar(SX3->X3_TAMANHO)+'"                MAXLENGTH="'+cValtoChar(SX3-
>X3_TAMANHO)+'"' onchange="upperCase(this,this.value);passar(this)" READONLY DISABLED ></TD>')
            endif
        ENDIF
    ENDIF

```

```

ENDIF
if nVolta = 2
    aadd(aDados2,'<TR>')
    nVolta := 0
endif
ENDIF
aadd(aFrame1,'<INPUT TYPE="hidden" VALUE=" " NAME="_'+cTabS+'_'+xRecno+'_'+alltrim(SX3-
>X3_CAMPO)+'" id="_'+cTabS+'_'+xRecno+'_'+alltrim(SX3->X3_CAMPO)+'">')
SX3->(dbskip())
END
aadd(aDados0,aDados2)
aadd(aDados0,aDados1)
HttpSession->xTelaE := aDados0
// alimenta a getdados
aDados0 := {}
aDados1 := {}
aDados2 := {}
cQuery      := "SELECT E1_PREFIXO, E1_NUM, E1_PARCELA, E1_TIPO, E1_VALOR, E1_VENCTO, E1_HIST,
R_E_C_N_O_ AS REC FROM "+RETSQNAME("SE1")+ " A WHERE A.E1_FILIAL = '"+xFilial("SE1")+"' AND
E1_CLIENTE = '"+SA1->A1_COD+"' AND E1_LOJA = '"+SA1->A1_LOJA+"' AND D_E_L_E_T_ = ''"
dbUseArea( .T., "TOPCONN", TcGenQry(,cQuery), "TRBSE1", .T., .F. )
TCSetField( "TRBSE1",'E1_PREFIXO','C',3,0)
TCSetField( "TRBSE1",'E1_NUM','C',8,0)
TCSetField( "TRBSE1",'E1_PARCELA','C',2,0)
TCSetField( "TRBSE1",'E1_TIPO','C',3,0)
TCSetField( "TRBSE1",'E1_VALOR','N',14,2)
TCSetField( "TRBSE1",'E1_VENCTO','D',8,0)

aDados3 := TRBSE1->(DBSTRUCT())
aadd(aDados1,'<TR>')
SX3->(DBSETORDER(2))
aadd(aDados1,'<TD BGCOLOR="#BBBBBB" >DELETAR?</TD>')
aadd(aDados1,'<TD BGCOLOR="#BBBBBB">Prefixo</TD>')
aadd(aDados1,'<TD BGCOLOR="#BBBBBB">No. Titulo</TD>')
aadd(aDados1,'<TD BGCOLOR="#BBBBBB">Parcela</TD>')
aadd(aDados1,'<TD BGCOLOR="#BBBBBB">Tipo</TD>')
aadd(aDados1,'<TD BGCOLOR="#BBBBBB">Vlr.Titulo</TD>')
aadd(aDados1,'<TD BGCOLOR="#BBBBBB">Vencimento</TD>')
aadd(aDados1,'<TD BGCOLOR="#BBBBBB">HISTORICO</TD>')
aadd(aDados1,'</TR>')
WHILE TRBSE1->(!EOF())
    aadd(aDados1,'<TR>')
    aadd(aDados1,'<TD><INPUT TYPE="checkbox" VALUE="0" NAME="_'+cTabI+'_'+cValtoChar(TRBSE1-
>REC)+'_DELETAR' onClick="delet(this,this.value);passar(this)"></TD>')
    aadd(aFrame1,'<TD><INPUT TYPE="hidden" VALUE="0" NAME="_'+cTabI+'_'+cValtoChar(TRBSE1-
>REC)+'_DELETAR' ID="_'+cTabI+'_'+cValtoChar(TRBSE1->REC)+'_DELETAR"></TD>')
    for nFor := 1 to len(aDados3)-1
        xVar      := IIF(VALTYPE(&('TRBSE1->'+aDados3[nFor][1]))=="N",cValtoChar(&('TRBSE1-
>'+aDados3[nFor][1])),IIF(VALTYPE(&('TRBSE1->'+aDados3[nFor][1]))=="D",DLOC(&('TRBSE1-
>'+aDados3[nFor][1])),IIF(VALTYPE(&('TRBSE1->'+aDados3[nFor][1]))=="U",",",&('TRBSE1->'+aDados3[nFor][1]))))
    if VALTYPE(&('TRBSE1->'+aDados3[nFor][1])) == "N"

```

```

aadd(aDados1,'<TD><INPUT    TYPE="text"    VALUE="" +xVar+""    NAME="_'+cTabl+'_'+cValtoChar(TRBSE1->REC)+'_'+alltrim(aDados3[nFor][1])+"" id="_'+cTabl+'_'+cValtoChar(TRBSE1->REC)+'_'+alltrim(aDados3[nFor][1])+"" SIZE="" +cValtoChar(aDados3[nFor][3])+"" MAXLENGTH="" +cValtoChar(aDados3[nFor][3])+"" onkeypress='return SomenteNumero(event);' onchange='passar(this)'></TD>')
elseif VALTYPE(&('TRBSE1->' + aDados3[nFor][1])) == "D"
aadd(aDados1,'<TD><INPUT    TYPE="text"    VALUE="" +xVar+""    NAME="_'+cTabl+'_'+cValtoChar(TRBSE1->REC)+'_'+alltrim(aDados3[nFor][1])+"" id="_'+cTabl+'_'+cValtoChar(TRBSE1->REC)+'_'+alltrim(aDados3[nFor][1])+"" SIZE="" +cValtoChar(aDados3[nFor][3]+2)+"" MAXLENGTH="" +cValtoChar(aDados3[nFor][3]+2)+"" onchange="validaDat(this,this.value);passar(this)" ></TD>')
else
aadd(aDados1,'<TD><INPUT    TYPE="text"    VALUE="" +xVar+""    NAME="_'+cTabl+'_'+cValtoChar(TRBSE1->REC)+'_'+alltrim(aDados3[nFor][1])+"" id="_'+cTabl+'_'+cValtoChar(TRBSE1->REC)+'_'+alltrim(aDados3[nFor][1])+"" SIZE="" +cValtoChar(aDados3[nFor][3])+"" MAXLENGTH="" +cValtoChar(aDados3[nFor][3])+"" onchange="upperCase(this,this.value);passar(this)" ></TD>')
endif
aadd(aFrame1,'<INPUT    TYPE="hidden"    VALUE=""    NAME="_'+cTabl+'_'+cValtoChar(TRBSE1->REC)+'_'+alltrim(aDados3[nFor][1])+"" id="_'+cTabl+'_'+cValtoChar(TRBSE1->REC)+'_'+alltrim(aDados3[nFor][1])+"" >')
next nfor
aadd(aDados1,'</TR>')

TRBSE1->(DBSKIP())
END
HttpSession->xTelaG := aDados1
httpSession->aFrame1 := aFrame1
cHtml := h_xModel3()
TRBSE1->(dbclosetarea())
WEB EXTENDED end
Return(cHtml)

```

O código fonte apresentado gera uma variável `aDados0` com o tipo Vetor com os dados da parte superior da página registro selecionado SA1, possuindo todas as Tags de input dos campos permitidos da tabela SA1

A Variável `aDados0` é atribuída em uma nova sessão **HttpSession->xTelaE**

Em seguida é executada uma query para trazer todos os títulos do cliente selecionado

```

cQuery := "SELECT E1_PREFIXO, E1_NUM, E1_PARCELA, E1_TIPO, E1_VALOR, E1_VENCTO, E1_HIST, R_E_C_N_O_ AS REC FROM "+RETSQNAME("SE1")+ " A WHERE A.E1_FILIAL = '"+xFilial("SE1")+"' AND E1_CLIENTE = '"+SA1->A1_COD+"' AND E1_LOJA = '"+SA1->A1_LOJA+"' AND D_E_L_E_T_ = ''"
dbUseArea( .T., "TOPCONN", TcGenQry(,cQuery), "TRBSE1", .T., .F. )
TCSetField( "TRBSE1", 'E1_PREFIXO', 'C', 3, 0)
TCSetField( "TRBSE1", 'E1_NUM', 'C', 8, 0)
TCSetField( "TRBSE1", 'E1_PARCELA', 'C', 2, 0)
TCSetField( "TRBSE1", 'E1_TIPO', 'C', 3, 0)
TCSetField( "TRBSE1", 'E1_VALOR', 'N', 14, 2)
TCSetField( "TRBSE1", 'E1_VENCTO', 'D', 8, 0)

```

Observamos que todos os registros poderia ser apresentados dependendo da necessidade do usuário, pois somente os campos foi adicionado serão apresentados

(E1_PREFIXO, E1_NUM, E1_PARCELA, E1_TIPO, E1_VALOR, E1_VENCTO, E1_HIST, R_E_C_N_O_)

Utilizamos o campo R_E_C_N_O_ para saber qual o registro devemos alterar ou excluir o campo ficara na tag do inputy como o nome do campo

Foi adicionado todos os campos na no vetor aFrame1 como **Hidden** para não ser exibido na tela mas possuindo informação.

Esse processo servira para trasportar os dados de variaveis de um Frame para outro

```
aadd(aFrame1,'<INPUT      TYPE="hidden"      VALUE="      "      NAME="_'+cTabl+'_'+cValtoChar(TRBSE1-
>REC)+'_'+alltrim(aDados3[nFor][1])+""      id="_'+cTabl+'_'+cValtoChar(TRBSE1-
>REC)+'_'+alltrim(aDados3[nFor][1])+"" >')
```

Visualizamos agora a pagina

```
<HTML>

<HEAD>
<TITLE>modelo 3</TITLE>
</HEAD>
<frameset rows="7%,40%,*" border=0>
  <frame name="top"      src="u_xTelaG0.APW"      marginwidth="10"      marginheight="10"      scrolling="No"
frameborder="no" noresize>
  <frame name="middle"      src="U_xTelaG1.APW"      marginwidth="10"      marginheight="10"      scrolling="Auto"
frameborder="no" noresize>
  <frame name="bottom"      src="U_xTelaG2.apw"      marginwidth="10"      marginheight="10"      scrolling="Auto"
frameborder="no" noresize>
</frameset>
<body>
</body>
</frameset>

</BODY>
</HTML>
```

A principio cada função/pagina tera a sua variavel para trabalhar e ser apresentada na tela

A função

- u_xTelaG0.apw frame dos botões superiores ira utilizar a sessão HttpSession->aFrame1
- u_xTelaG1.apw frame dos campos da enchoice ira utilizar a sessão HttpSession->xTelaG
- u_xTelaG2.apw frame da getdados ira utilizar a sessão HttpSession->aFrame1

```
#INCLUDE "Protheus.ch"
#include "APWEBEX.CH"

User Function XTELAG0()
Local cHtml := ""
Private aFrame1 := HttpSession->aFrame1
WEB EXTENDED INIT cHtml
cHtml := h_XTELAG0()
```



```
WEB EXTENDED end
Return(cHtml)
```

Utilizando a sessão aFrame1 para adicionar a uma variavel privada para ser transportada para a função da pagina h_XTELAGO()

```
<HTML>
  <HEAD>
    <TITLE>GETDADOS</TITLE>
  </HEAD>
  <script language="JavaScript">

var aRoda =new Array();
function ok(){
  document.modx31.action = "u_gravar3.apw";
  document.modx31.submit()
}

function recebi(cCpo,cValor){
  document.getElementById(cCpo).value = cValor;
}

</script>
  <BODY BGCOLOR="#E6EEEE">
    <FORM ACTION="#" METHOD="POST" TITLE="mod3" name="modx31">
      <% for nFor := 1 to len(aFrame1) %>
        <%= aFrame1[nFor]%>
      <% next nFor%>
      <TABLE BORDER="0" WIDTH="50%" CELLPADDING="0" CELLSPACING="0">
        <TR>
          <TD><INPUT TYPE="submit" VALUE="Salvar" NAME="Salvar" onClick=javascript:ok()></TD>
          <TD><INPUT TYPE="submit" VALUE="Cancelar" NAME="Cancelar"
onClick="javascript:parent.window.close();"></TD>
        </TR>
      </TABLE>
    </FORM>
  </BODY>
</HTML>
```

Para a pagina apresentada fora adicionada duas funções javascript para tratar as variáveis de outra pagina apresentada no Frame

```
function ok(){
  document.modx31.action = "u_gravar3.apw";
  document.modx31.submit()
}
```

Função que ira fazer a adição da pagina para salvar os dados

```
function recebi(cCpo,cValor){
  document.getElementById(cCpo).value = cValor;
```

```
}

```

Função que ira alimentar todos os campos Hidden adicionados na pagina
Esse processo ira se realizar quando o usuario alterar qualquer campos dos Frames (middle,button)

getElementByld método que busca o acesso do elemento com o ID especificado.

```
#INCLUDE "Protheus.ch"
#INCLUDE "APWEBEX.CH"

User Function XTELAG1()
Local cHtml := ""
Private aDados := HttpSession->xTelaE
WEB EXTENDED INIT cHtml
//VARINFO('aDados',aDados)
cHtml := h_XTELAG1()
WEB EXTENDED end
Return(cHtml)

```

Utilizando a sessão xTelaE para adicionar a uma variavel privada para ser transportada para a função da pagina
h_XTELAG1()

```
<HTML>

<HEAD>
<TITLE></TITLE>
</HEAD>
<script language="JavaScript">

function SomenteNumero(e){
    var tecla=(window.event)?event.keyCode:e.which;
    if((tecla > 47 && tecla < 58) || tecla == 46 || tecla == 44)
    {
        return true;
    }
    else
    {
        if (tecla != 8 )
        {
            return false;
        }
    }
    else
    {
        return true;
    }
}

function passar(campo) {
parent.frames["top"].recebi( campo.name,campo.value );
}

```



```

function upperCase(campo,valor){
    campo.value = valor.toUpperCase()
}

function validaDat(campo,valor) {
    var date=valor;
    var ardt=new Array;
    var ExpReg=new RegExp("(0[1-9]|[12][0-9]3[01])|(0[1-9]1[012])|[12][0-9]{3}");
    ardt=date.split("/");
    erro=false;
    if ( date.search(ExpReg)==-1){
        erro = true;
    }
    else if (((ardt[1]==4)||((ardt[1]==6)||((ardt[1]==9)||((ardt[1]==11))&&(ardt[0]>30))
        erro = true;
    else if ( ardt[1]==2) {
        if ((ardt[0]>28)&&((ardt[2]%4)!=0))
            erro = true;
        if ((ardt[0]>29)&&((ardt[2]%4)==0))
            erro = true;
    }
    if (erro) {
        alert(""" + valor + "" não é uma data válida!!! utilize dd/mm/aaaa");
        campo.focus();
        campo.value = " / / ";
        return false;
    }
    return true;
}

</script>
<BODY BGCOLOR="#E6EEED">
    <FORM ACTION="#" METHOD="POST" TITLE="alterar" name="modx32" id="modx32" >
        <TABLE BORDER="0" WIDTH="100%" CELLPADDING="0" CELLSPACING="0">
            <% For nFor := 1 to len(aDados) %>
                <% For nFor2 := 1 to len(aDados[nFor]) %>
                    <%= aDados[nFor][nFor2] %>
                <% next nFor2 %>
            <% next nFor %>
        </TABLE>
    </FORM>
</BODY>
</HTML>

```

Conforme a análise do código fonte observamos 4 funções JavaScript

```

function SomenteNumero(e){
    var tecla=(window.event)?event.keyCode:e.which;
    if((tecla > 47 && tecla < 58) || tecla == 46 || tecla == 44)
    {

```

```

    return true;
}
else
{
    if (tecla != 8 )
    {
        return false;
    }

    else
    {
        return true;
    }
}
}

```

Função que permite somente a inclusão de numero e pontuação

```

function upperCase(campo,valor){
    campo.value = valor.toUpperCase()
}

```

Função que transforma qualquer caracter em maiusculo

```

function validaDat(campo,valor) {
    var date=valor;
    var ardt=new Array;
    var ExpReg=new RegExp("(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[012])/[12][0-9]{3}");
    ardt=date.split("/");
    erro=false;
    if ( date.search(ExpReg)==-1){
        erro = true;
    }
    else if (((ardt[1]==4)||((ardt[1]==6)||((ardt[1]==9)||((ardt[1]==11))&&(ardt[0]>30))
        erro = true;
    else if ( ardt[1]==2) {
        if ((ardt[0]>28)&&((ardt[2]%4)!=0))
            erro = true;
        if ((ardt[0]>29)&&((ardt[2]%4)==0))
            erro = true;
    }
    if (erro) {
        alert(" " + valor + " não é uma data válida!!! utilize dd/mm/aaaa");
        campo.focus();
        campo.value = " / / ";
        return false;
    }
    return true;
}

```

Função que valida a data digitada

```
function passar(campo) {
parent.frames["top"].recebi( campo.name,campo.value );
}
```

Função de chamada de funções de outro Frame a qual indicamos o Frame chamado **TOP** passando por parâmetro na função dois conteúdo o nome do campo e o seu valor a qual observamos na pagina

```
<TR>
<TD>Codigo          </TD>
<TD><INPUT          TYPE="text"          VALUE="000006"          NAME="_SA1_134_A1_COD"
id="_SA1_134_A1_COD"          SIZE="6"          MAXLENGTH="6"
onchange="upperCase(this,this.value);passar(this)" ></TD>
<TD>Loja            </TD>
<TD><INPUT          TYPE="text"          VALUE="01"          NAME="_SA1_134_A1_LOJA"
id="_SA1_134_A1_LOJA"          SIZE="2"          MAXLENGTH="2"
onchange="upperCase(this,this.value);passar(this)" ></TD>
</TR>
```

Observamos que a função onchange esta sendo executada duas funções UpperCase() depois separada por ponto e virgula a função passar que ira alimentar o campo Hidden do frame Top

Vejamos o ultimo frame u_xTelaG2.apw frame da getdados

```
#INCLUDE "Protheus.ch"
#INCLUDE "APWEBEX.CH"

User Function XTELAG2()
Local cHtml := ""
Private aDados1 := HttpSession->xTelaG
WEB EXTENDED INIT cHtml

cHtml := h_XTELAG2()
WEB EXTENDED end
Return(cHtml)
```

Utilizando a sessão xTelaG para adicionar a uma variavel privada para ser transportada para a função da pagina h_XTELAG2()

```
<HTML>
<HEAD>
<TITLE>GETDADOS</TITLE>
<META NAME="LEANDRO_DUARTE" CONTENT="ADVPL-WEB">
</HEAD>
<script language="JavaScript">

function SomenteNumero(e){
var tecla=(window.event)?event.keyCode:e.which;
if((tecla > 47 && tecla < 58) || tecla == 46 || tecla == 44)
{
return true;
}
```

```

}
else
{
  if (tecla != 8 )
  {
    return false;
  }

  else
  {
    return true;
  }
}
}

function passar(campo) {
parent.frames["top"].recebi( campo.name,campo.value );
}

function delet(campo,valor){
if (valor==1){
valor=0;
}
else {
valor=1;
}
campo.value=valor;
}

function upperCase(campo,valor){
  campo.value = valor.toUpperCase()
}

function validaDat(campo,valor) {
  var date=valor;
  var ardt=new Array;
  var ExpReg=new RegExp("(0[1-9]|[12][0-9]3[01])/(0[1-9]1[012])/[12][0-9]{3}");
  ardt=date.split("/");
  erro=false;
  if ( date.search(ExpReg)==-1){
    erro = true;
  }
  else if (((ardt[1]==4)||((ardt[1]==6)||((ardt[1]==9)||((ardt[1]==11))&&(ardt[0]>30))
    erro = true;
  else if ( ardt[1]==2) {
    if ((ardt[0]>28)&&((ardt[2]%4)!=0))
      erro = true;
    if ((ardt[0]>29)&&((ardt[2]%4)==0))
      erro = true;
  }
  if (erro) {
    alert(" " + valor + " não é uma data válida!!! utilize dd/mm/aaaa");
  }
}

```

```

        campo.focus();
        campo.value = " / / ";
        return false;
    }
    return true;
}

</script>
<BODY BGCOLOR="#E6EEEE">
<FORM ACTION="#" METHOD="POST" TITLE="MODELO3" NAME="modx33">
    <TABLE BORDER="1" WIDTH="100%" CELLPADDING="0" CELLSPACING="0">
        <% For nFor := 1 to len(aDados1) %>
            <%= aDados1[nFor] %>
        <% next nFor %>
    </TABLE>
</FORM>
</BODY>
</HTML>

```

Vejamos que ambos os frames possuem as funções (*SomenteNumero*, *validaDat*, *passar*), mas nessa pagina possui uma função chamada Delet

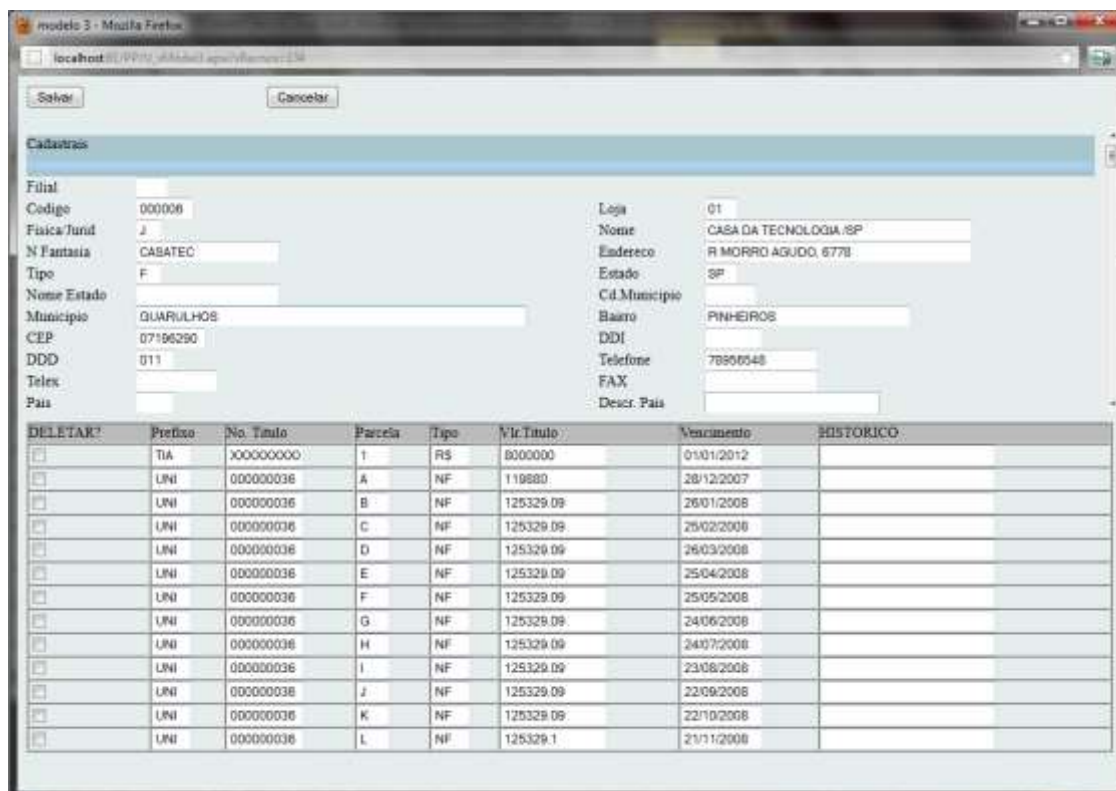
```

function delet(campo,valor){
if (valor==1){
valor=0;
}
else {
valor=1;
}
campo.value=valor;
}

```

Função que iremos utilizar para atribuir um valor de "0" quando não for selecionado o check de deletar o registro e "1" quando for selecionado o registro para excluir

Vejamos a pagina



Cadastro

Filial:
 Código: 000000
 Física/Jurid: J
 N Fantasia: CABATEC
 Tipo: F
 Nome Estado:
 Município: GUARULHOS
 CEP: 07196290
 DDD: 011
 Telex:
 País:

Loja: 01
 Nome: CASA DA TECNOLOGIA /SP
 Endereço: R MORRO AGUDO, 6778
 Estado: SP
 Cd. Município:
 Bairro: PINHEIROS
 DDI:
 Telefone: 78806548
 FAX:
 Descr. País:

DELETAR?	Prefixo	No. Título	Parcela	Tipo	Vlr. Título	Vencimento	HISTORICO
<input type="checkbox"/>	TIA	XXXXXXXXXX	1	RS	800000	01/01/2012	
<input type="checkbox"/>	UNI	000000036	A	NF	119880	28/12/2007	
<input type="checkbox"/>	UNI	000000036	B	NF	125329.09	26/01/2008	
<input type="checkbox"/>	UNI	000000036	C	NF	125329.09	25/02/2008	
<input type="checkbox"/>	UNI	000000036	D	NF	125329.09	26/03/2008	
<input type="checkbox"/>	UNI	000000036	E	NF	125329.09	25/04/2008	
<input type="checkbox"/>	UNI	000000036	F	NF	125329.09	25/05/2008	
<input type="checkbox"/>	UNI	000000036	G	NF	125329.09	24/06/2008	
<input type="checkbox"/>	UNI	000000036	H	NF	125329.09	24/07/2008	
<input type="checkbox"/>	UNI	000000036	I	NF	125329.09	23/08/2008	
<input type="checkbox"/>	UNI	000000036	J	NF	125329.09	22/09/2008	
<input type="checkbox"/>	UNI	000000036	K	NF	125329.09	22/10/2008	
<input type="checkbox"/>	UNI	000000036	L	NF	125329.1	21/11/2008	

Observamos na função do frame **TOP** o botões salvar

```
#include "Protheus.ch"
#include "apwebex.ch"

user function gravar3()
Local cHtml := ""
Local aDados := {}
Local aDados1 := {}
Local aParms := {}
Local xRecno := HttpSession->xRecno
Local xTabS      := HttpSession->cTabS
Local xTabI := HttpSession->cTabI
WEB EXTENDED INIT cHtml

// a linha abaixo monta um array com os parametros recebidos via post
// no formato ( [1] = nome / [2] conteudo )
aEval( HttpPost->aPost      , {x|x| iif(alltrim(x)=='SALVAR' .AND. EMPTY("&('HttpPost->"+x)) ,NIL,aadd(aParms,{ x
, &('HttpPost->"+x)) } ) } )
// TRATAMENTO DOS CAMPOS VIRTUAIS
varinfo('aParms',aParms)

DBSELECTAREA("SX3")
SX3->(dbsetorder(1))
SX3->(DBSEEK(xTabS))
While SX3->X3_ARQUIVO == (xTabS)
  IF EMPTY(X3_CONTEXT) .OR. X3_CONTEXT == 'R'
    AADD(aDados,alltrim(SX3->X3_CAMPO))
```

```

        AADD(aDados1,SX3->X3_TIPO)
    endif
    SX3->(dbskip())
end
SX3->(DBSEEK(xTabl))
While SX3->X3_ARQUIVO == (xTabl)
    IF EMPTY(X3_CONTEXT) .OR. X3_CONTEXT == 'R'
        AADD(aDados,alltrim(SX3->X3_CAMPO))
        AADD(aDados1,SX3->X3_TIPO)
    endif
    SX3->(dbskip())
end
//rotina de gravação
For nFor := 1 to len(aParms)
    IF !EMPTY(aParms[nFor][2]) .AND. aParms[nFor][1] != 'SALVAR' .and. !('DELETAR' $ aParms[nFor][1])
        aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
        xTab := substr(aParms[nFor][1],1,at("_",aParms[nFor][1])-1)
        aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
        nRec := val(substr(aParms[nFor][1],1,at("_",aParms[nFor][1])-1))
        aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
        cCpo := alltrim(aParms[nFor][1])
        xValor := aParms[nFor][2]
        NN := aScan(aDados,cCpo)
        if NN>0
            xValor := IIF(aDados1[NN]=='N',VAL(xValor),IIF(aDados1[NN]=='D',CTOD(xValor),xValor))
            (xTab)->(dbgoto(nRec))
            Reclock((xTab),.f.)
            &((xTab)+->''+cCpo) := xValor
            Msunlock()
        ENDIF
    elseif ('DELETAR' $ aParms[nFor][1])
        aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
        xTab := substr(aParms[nFor][1],1,at("_",aParms[nFor][1])-1)
        aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
        nRec := val(substr(aParms[nFor][1],1,at("_",aParms[nFor][1])-1))
        (xTab)->(dbgoto(nRec))
        xValor := aParms[nFor][2]
        IF ALLTRIM(xValor)=='1'
            CONOUT('ENTREI')
            Reclock((xTab),.f.)
            DBDELETE()
            Msunlock()
        ENDIF
    ENDIF
Next nFor
cHtml := " <html> "
cHtml += " <body> "
cHtml += " <form name='login' method='post' action='u_TlInWB3.apw'> "
cHtml += " <script language='JavaScript'><INPUT TYPE='hidden' VALUE=xRecno NAME='xRecnos'></script> "
cHtml += " </form> "
cHtml += " </body> "
cHtml += " </html> "

```



```
cHtml += " <script language='JavaScript'> "
//cHtml += " window.opener.parent.direita.location.reload();javascript:parent.window.close(); "
cHtml += " javascript:parent.window.close(); "
cHtml += " </script> "
WEB EXTENDED END
Return(cHtml)
```

O código fonte apresenta uma série de análises de gravação definidas no código fonte xmodel3.apf regras que o código fonte definiu qual seria a tabela e o Recno para ser atualizado.

Como podemos observar foi feito um tratamento com o nome de cada campo possuindo a sua respectiva tabela e seu **recno**

EX: **NAME=SE1_17_E1_CODIGO**

```
// a linha abaixo monta um array com os parametros recebidos via post
// no formato ( [1] = nome / [2] conteudo )
aEval( HttpPost->aPost , {x| iif(alltrim(x)=='SALVAR' .AND. EMPTY("&("HttpPost->" +x)) ,NIL,aadd(aParms,{ x
, "&("HttpPost->" +x)) } ) } )
// TRATAMENTO DOS CAMPOS VIRTUAIS
varinfo('aParms',aParms)
```

Nesse bloco acima apresentado foi feito o tratamento da matriz HTTPPOST capturando todos os campos e alimentando e seus valores e adicionando na variável **aParms**

```
aParms[1][1] = SE1_17_E1_CODIGO
aParms[1][2] = 000001
```

```
SX3->(dbsetorder(1))
SX3->(DBSEEK(xTabS))
While SX3->X3_ARQUIVO == (xTabS)
  IF EMPTY(X3_CONTEXT) .OR. X3_CONTEXT == 'R'
    AADD(aDados,alltrim(SX3->X3_CAMPO))
    AADD(aDados1,SX3->X3_TIPO)
  endif
  SX3->(dbskip())
end
SX3->(DBSEEK(xTabI))
While SX3->X3_ARQUIVO == (xTabI)
  IF EMPTY(X3_CONTEXT) .OR. X3_CONTEXT == 'R'
    AADD(aDados,alltrim(SX3->X3_CAMPO))
    AADD(aDados1,SX3->X3_TIPO)
  endif
  SX3->(dbskip())
end
```

Lembrando que estamos trabalhando com o modelo3 com duas tabelas na mesma tela.

Depois temos o processo mais rápido de gravação entre os campos enviados pois foram os únicos que sofreram alteração.

```
//rotina de gravação
```

```

For nFor := 1 to len(aParms)
  IF !EMPTY(aParms[nFor][2]) .AND. aParms[nFor][1] != 'SALVAR' .and. !('DELETAR' $ aParms[nFor][1])
    aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
    xTab := substr(aParms[nFor][1],1,at("_",aParms[nFor][1])-1)
    aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
    nRec := val(substr(aParms[nFor][1],1,at("_",aParms[nFor][1])-1))
    aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
    cCpo := alltrim(aParms[nFor][1])
    xValor := aParms[nFor][2]
    NN := aScan(aDados,cCpo)
    if NN>0
      xValor := IIF(aDados1[NN]=='N',VAL(xValor),IIF(aDados1[NN]=='D',CTOD(xValor),xValor))
      (xTab)->(dbgoto(nRec))
      Reclock((xTab),.f.)
      &((xTab)+->'+cCpo) := xValor
      Msunlock()
    ENDIF
  elseif ('DELETAR' $ aParms[nFor][1])
    aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
    xTab := substr(aParms[nFor][1],1,at("_",aParms[nFor][1])-1)
    aParms[nFor][1] := substr(aParms[nFor][1],at("_",aParms[nFor][1])+1)
    nRec := val(substr(aParms[nFor][1],1,at("_",aParms[nFor][1])-1))
    (xTab)->(dbgoto(nRec))
    xValor := aParms[nFor][2]
    IF ALLTRIM(xValor)=='1'
      CONOUT('ENTREI')
      Reclock((xTab),.f.)
      DBDELETE()
      Msunlock()
    ENDIF
  ENDIF
ENDIF
Next nFor

```

E no final executa o script de encerramento da pagina

```

cHtml := " <html> "
cHtml += " <body> "
cHtml += " <form name='login' method='post' action='u_TlInWB3.apw'> "
cHtml += " <script language='JavaScript'><INPUT TYPE='hidden' VALUE=xRecno NAME='xRecnos'></script> "
cHtml += " </form> "
cHtml += " </body> "
cHtml += " </html> "
cHtml += " <script language='JavaScript'> "
//cHtml += " window.opener.parent.direita.location.reload();javascript:parent.window.close();"
cHtml += " javascript:parent.window.close();"
cHtml += " </script> "

```

Exercício

Criar uma pagina para apresentar os dados do fornecedor SA2 associado aos títulos SE2 que esse fornecedor possui

10. Desenvolvimento e impressão de Relatorio na WEB

O processo de emitir relatorio em uma pagina de internet é muito simples.

Para que possamos fazer isso devemos elaborar uma pagina a onde queremos que seja apresentada os parametros de seleção dos registros buscando os parametros do SX1

E logo em seguida a apresentação dos dados selecionados por uma **QUERY** ou por **DBSKIP**

Vamos ver os exemplos

```
#INCLUDE "PROTHEUS.CH"
#INCLUDE "APWEBEX.CH"

USER FUNCTION relatoMI()

Local cHtml := ""
Local cQuery := ""
Private aDados := {}
Private cDados1 := HttpGET->MV_PAR02
DEFAULT cDados1 := ""
WEB EXTENDED INIT cHtml
CONOUT('HttpGET->MV_PAR01')
CONOUT(HttpGET->MV_PAR01)
CONOUT('HttpGET->MV_PAR02')
CONOUT(HttpGET->MV_PAR02)
CONOUT('HttpGET->MV_PAR03')
CONOUT(HttpGET->MV_PAR03)
CONOUT('HttpGET->MV_PAR04')
CONOUT(HttpGET->MV_PAR04)

if empty(cDados1)
    CONOUT('PARAMETRO')
    SX1->(DBGOTOP())
    SX1->(DBSEEK("SA1SA1 01"))
    WHILE SX1->(IEOF()) .AND. ALLTRIM(SX1->X1_GRUPO) == "SA1SA1"
        AADD(aDados,{SX1->X1_PERGUNT,SX1->X1_TAMANHO})
        SX1->(DBSKIP())
    END
    cHtml := H_relatoMI()
ELSE
    CONOUT('RELATORIO')
```

```

cQuery := " SELECT A1_COD, A1_NOME, A1_NREDUZ, A1_END, A1_EST, A1_ESTADO "
cQuery += " FROM "+retsqliName("SA1")
cQuery += " where A1_FILIAL = '"+XfiliAl("SA1")+" "'
cQuery += " AND A1_COD BETWEEN '"+HttpGET->MV_PAR01+"' AND '"+HttpGET->MV_PAR02+"' "
cQuery += " AND A1_EST BETWEEN '"+HttpGET->MV_PAR03+"' AND '"+HttpGET->MV_PAR04+"' "
dbUseArea( .T., "TOPCONN", TcGenQry(,cQuery), "TRBSA1", .T., .F. )
CONOUT(cQuery)
cHtml := H_relatoMI()
TRBSA1->(dbclosearea())
ENDIF
WEB EXTENDED END
RETURN(cHtml)

```

Conforme analisamos o fonte foi apresentado uma regra de recebimento de dados

```

Private cDados1 := HttpGET->MV_PAR02
DEFAULT cDados1 := ""

```

A representação do código fonte foi se a variável cDados1 receber o parâmetro via GET como **Nil** será atribuída um valor vazio.

E logo em seguida terá uma condição se a variável for vazia habilite os parâmetros para a seleção e execução da mesma página, para quando for preenchido os parâmetros a variável cDados1 estará com o valor e executará a query para trazer os dados desejados.

Após a execução da query será levada a tabela temporária para a página APH os dados da tabela!

H_relatoMI

```

<HTML>

<HEAD>
<TITLE>
    <% IF SELECT("TRBSA1")>0 %>
        RELATORIO
    <% ELSE%>
        PARAMETROS
    <%ENDIF%>
</TITLE>
<META NAME="LEANDRO_DUARTE" CONTENT="ADVPL-WEB">
</HEAD>
<BODY onunload="window.opener.location.reload()">
    <% IF SELECT("TRBSA1")>0 %>
        <TABLE BORDER="1" WIDTH="100%" CELLPADDING="0" CELLSPACING="0">
            <TR>

```

```

        <TD>Codigo    </TD>
        <TD>Nome      </TD>
        <TD>N Fantasia </TD>
        <TD>Endereco   </TD>
        <TD>Estado     </TD>
        <TD>Nome Estado </TD>
    </TR>
    <% WHILE TRBSA1->(EOF())%>
        <TR>
            <TD><%= TRBSA1->A1_COD %></TD>
            <TD><%= TRBSA1->A1_NOME %></TD>
            <TD><%= TRBSA1->A1_NREDUZ %></TD>
            <TD><%= TRBSA1->A1_END %></TD>
            <TD><%= TRBSA1->A1_EST %></TD>
            <TD><%= TRBSA1->A1_ESTADO %></TD>
        </TR>
    <% TRBSA1->(DBSKIP())
    END
    %>
    <TR>
        <TD COLSPAN="3" ALIGN="CENTER"><INPUT TYPE="submit" VALUE="IMPRIMIR"
NAME="IMPRIMIR" onclick="javascript:window.print();"></TD>
        <TD COLSPAN="3" ALIGN="CENTER"><INPUT TYPE="submit" VALUE="FECHAR"
NAME="FECHAR" onclick="javascript:window.close();"></TD>
    </TR>
</TABLE>
<% ELSE%>
    <FORM ACTION="U_relatoMI.APW" METHOD="GET" TITLE="PARAMETROS"
NAME="PARAMETROS" ID="PARAMETROS" >
        <TABLE BORDER="1" WIDTH="40%" CELLPADDING="0" CELLSPACING="0">
            <TR>
                <TD>CLIENTE DE</TD>
                <TD><INPUT TYPE="text" VALUE=" " NAME="MV_PAR01" SIZE="6"
MAXLENGTH="7"></TD>
            </TR>
            <TR>
                <TD>CLIENTE ATÉ</TD>
                <TD><INPUT TYPE="text" VALUE="ZZZZZZ" NAME="MV_PAR02" SIZE="6"
MAXLENGTH="7"></TD>
            </TR>
            <TR>
                <TD>ESTADO DE</TD>
                <TD><INPUT TYPE="text" VALUE=" " NAME="MV_PAR03" SIZE="2"
MAXLENGTH="2"></TD>
            </TR>
            <TR>
                <TD>ESTADO ATÉ</TD>
                <TD><INPUT TYPE="text" VALUE="ZZ" NAME="MV_PAR04" SIZE="2"
MAXLENGTH="2"></TD>
            </TR>
            <TR>
                <TD><INPUT TYPE="submit" VALUE="OK" NAME="OK" ></TD>
            </TR>
        </TABLE>
    </FORM>
</%>

```

```

                                <TD><INPUT   TYPE="submit"   VALUE="CANCELAR"   NAME="CANCELAR"
onclick="opener.location.reload();window.close()"></TD>
                                </TR>
                                </TABLE>
                                </FORM>
                                <%ENDIF%>
</BODY>
</HTML>

```

Analizamos o código Fonte

```

<TITLE>
    <% IF SELECT("TRBSA1")>0 %>
        RELATORIO
    <% ELSE%>
        PARAMETROS
    <%ENDIF%>
</TITLE>

```

Nesse bloco definimos o título da página com a seleção <% IF SELECT("TRBSA1")>0 %> caso seja verdadeiro a tabela foi criada e executada para apresentar a informação caso contrário não foi alimentada a tabela e será apresentada o título de parâmetros

Logo abaixo o próprio código fonte apresenta outra análise da página para saber se deve apresentar os parâmetros antes listados ou os dados da própria tabela.

```

<% IF SELECT("TRBSA1")>0 %>
    <TABLE BORDER="1" WIDTH="100%" CELLPADDING="0" CELLSPACING="0">
    <TR>
        <TD>Codigo    </TD>
        <TD>Nome      </TD>
        <TD>N Fantasia </TD>
        <TD>Endereco   </TD>
        <TD>Estado     </TD>
        <TD>Nome Estado </TD>
    </TR>
    <% WHILE TRBSA1->(!EOF())%>
        <TR>
            <TD><%= TRBSA1->A1_COD %></TD>
            <TD><%= TRBSA1->A1_NOME %></TD>
            <TD><%= TRBSA1->A1_NREDUZ %></TD>
            <TD><%= TRBSA1->A1_END %></TD>
            <TD><%= TRBSA1->A1_EST %></TD>

```

```

        <TD><%= TRBSA1->A1_ESTADO %></TD>
    </TR>
    <%
        TRBSA1->(DBSKIP())
    END
    %>
    <TR>
        <TD COLSPAN="3" ALIGN="CENTER"><INPUT TYPE="submit" VALUE="IMPRIMIR"
NAME="IMPRIMIR" onclick="javascript:window.print();"></TD>
        <TD COLSPAN="3" ALIGN="CENTER"><INPUT TYPE="submit" VALUE="FECHAR"
NAME="FECHAR" onclick="javascript:window.close();"></TD>
    </TR>
</TABLE>
<% ELSE%>
    <FORM ACTION="U_relatoMI.APW" METHOD="GET" TITLE="PARAMETROS"
NAME="PARAMETROS" ID="PARAMETROS" >
        <TABLE BORDER="1" WIDTH="40%" CELLPADDING="0" CELLSPACING="0">
            <TR>
                <TD>CLIENTE DE</TD>
                <TD><INPUT TYPE="text" VALUE=" " NAME="MV_PAR01" SIZE="6"
MAXLENGTH="7"></TD>
            </TR>
            <TR>
                <TD>CLIENTE ATÉ</TD>
                <TD><INPUT TYPE="text" VALUE="ZZZZZZ" NAME="MV_PAR02" SIZE="6"
MAXLENGTH="7"></TD>
            </TR>
            <TR>
                <TD>ESTADO DE</TD>
                <TD><INPUT TYPE="text" VALUE=" " NAME="MV_PAR03" SIZE="2"
MAXLENGTH="2"></TD>
            </TR>
            <TR>
                <TD>ESTADO ATÉ</TD>
                <TD><INPUT TYPE="text" VALUE="ZZ" NAME="MV_PAR04" SIZE="2"
MAXLENGTH="2"></TD>
            </TR>
            <TD><INPUT TYPE="submit" VALUE="OK" NAME="OK" ></TD>
            <TD><INPUT TYPE="submit" VALUE="CANCELAR" NAME="CANCELAR"
onclick="opener.location.reload();window.close();"></TD>
        </TR>
    </TABLE>
</FORM>
<%ENDIF%>

```

Na impressão da tabela temporaria é apresentado duas funções javascript no final da pagina

```

<TD COLSPAN="3" ALIGN="CENTER"><INPUT TYPE="submit" VALUE="IMPRIMIR" NAME="IMPRIMIR"
onclick="javascript:window.print();"></TD>
<TD COLSPAN="3" ALIGN="CENTER"><INPUT TYPE="submit" VALUE="FECHAR" NAME="FECHAR"
onclick="javascript:window.close();"></TD>

```


A informação de Imprimir executa o função **javascript:window.print()** que executa a rotina de impressão própria da pagina.

A informação de Fechar executa o função **javascript:window.close ()** que executa a rotina de fechar a pagina.

Exercício

Desenvolver um relatório com os dados da Tabela SA1

11. UPLOAD / DOWNLOAD

Para simular os processos da Web demonstraremos os dados de Upload e download utilizando as funções protheus ADVPL juntamente com Javascript.

```
#INCLUDE "PROTHEUS.CH"
#INCLUDE "APWEBEX.CH"

USER FUNCTION UPLOAD()
Local cHtml := ""
Local cRootWeb := GetSrvProfString("RootWeb","web")
Local lUnix := IsSrvUnix()
Private aDir := {}
WEB EXTENDED INIT cHtml
nTamRoot := len(alltrim(cRootWeb))
cCaracter := iif(lUnix,"/","\")

// funcao para descobrir se o ambiente eh linux - IsSrvUnix()

// retirar barra sobressalente
if substr(cRootWeb,nTamRoot,1) == "/" .OR. substr(cRootWeb,nTamRoot,1) == "\"
    cRootWeb := substr(cRootWeb,1,nTamRoot-1)
endif

//colocar o parametro no mesma situacao do que a do windows...
if ! lUnix
    cRootWeb := STRTRAN(cRootWeb,"/","\")
else
    cRootWeb := STRTRAN(cRootWeb,"","\")
endif

// se nao existir diretorios, cria!
nRetorno := MAKEDIR( cRootWeb+"arquivos", 1)

cDiretorio := cRootWeb+"arquivos"
nRetorno := MAKEDIR( cDiretorio , 1 )

cDiretorio := cDiretorio+"\"
```

```

if !Unix
  cDiretorio := Strtran(cDiretorio,"\\","/")
endif
cDiretorio := cRootWeb+"arquivos*.*"
conout(cDiretorio)
aDire := DIRECTORY(cDiretorio, "D", 1 )
For nFor := 1 to len(aDire)
  IF aDire[nFor][2] != 0
    AADD(aDir,{aDire[nFor][1],aDire[nFor][2],"http://localhost:81/PP/arquivos/"+aDire[nFor][1]})
  ENDIF
Next nFor
varinfo('aDir',aDir )
cHtml := h_UPLOAD()
WEB EXTENDED END
RETURN(cHtml)

```

Nesse código fonte demonstramos o processos de subir um arquivo para o servidor (Upload) e disponibilizar para baixar o arquivo (Download).
Para isso vamos analisar o código fonte.

Funções:

GetSrvProfString

Recupera o conteúdo de uma chave de configuração, do ambiente em uso, no arquivo de configuração (.INI) do TOTVS Application Server.

GetSrvProfString (< cChave>, < cDefault>) --> cRet

cChave Caracter Indica a chave que deve ser lida do arquivo de configuração.

cDefault Caracter Indica o conteúdo da chave a ser retornada, caso a chave não seja encontrada.

IsSrvUnix

Informa se Application Server está sendo executado em ambiente Unix®, Linux® ou Microsoft Windows®.

IsSrvUnix () --> lRet

Ret(logico)

Retorna verdadeiro (.T.), se o Application Server estiver sendo executado em ambiente Unix® ou Linux®; caso contrário, retornará falso (.F.), se estiver sendo executado em ambiente Microsoft Windows®.

MAKEDIR

Cria um diretório.

MakeDir (< cNovoDir>, [xParam2]) --> nRet

cNovoDir Caracter Indica o nome do diretório que será criado.

xParam2 Qualquer Compatibilidade.

nRet(numerico)

Retorna zero (0), se o diretório for criado com sucesso.

DIRECTORY

Cria um array bidimensional com o conteúdo de um diretório.

Para isso, retorna informações a respeito dos arquivos no diretório corrente ou especificado. Essa função é semelhante a `ADir()`, porém, retorna um único array ao invés de adicionar valores a uma série de arrays existentes passados por referência

`Directory (< cDirEsp>, [cAtributos], [xParam3], [lCaseSensitive]) --> aRet`

cDirEsp Caracter Indica o diretório que será pesquisado e os arquivos que serão apresentados. Além disso, caracteres do tipo curinga são permitidos na especificação de arquivos. Caso esse parâmetro não seja especificado, o valor padrão é *.*.

CAtributos Caracter Indica quais arquivos com atributos especiais devem ser incluídos no array. Esse parâmetro consiste em uma string que contém um ou mais dos caracteres H, S, D e V. Para mais detalhes, consulte a tabela A na área Observações.

xParam3 Nulo Compatibilidade. Deve ser informado o valor nulo (NIL)

lCaseSensitive Lógico Indica se, verdadeiro (.T.), o nome do arquivo será transformado para letra maiúscula; caso contrário, falso (.F.), o nome do arquivo será retornado conforme escrito no disco rígido.

VARINFO

Gera um texto ASCII e/ou Html, com possibilidade de ECHO para o Console do Protheus Server (caso habilitado), com as informações sobre o conteúdo de uma variável de memória Advpl, de qualquer tipo.

Cada tipo de variável possui um tratamento para conversão em String:

CodeBlock: É exibido apenas o tipo da mesma (B)

Array: Todos os níveis e elementos do mesmo são explorados recursivamente.

Objeto: No caso de um Objeto de XML e/ou Web Services, são exploradas todas as suas propriedades recursivamente.

(demais tipos): São convertidos para String através da função `AllToChar`

Observação: O segundo parâmetro (`xVar`) deve ser uma variável Advpl que deve existir no escopo de variáveis. Caso a variável não exista, o processamento é abortado com a ocorrência de erro "Variável does not exist" . Para saber se uma determinada variável existe no escopo da execução da função atual, deve ser utilizada a função `Advpl TYPE()`, onde passamos a variável a ter seu tipo determinado como string (entre aspas) .

`VARINFO (< cld >, < xVar >, [nMargem], [lHtml], [lEcho]) --> cVarInfo`

Cid Caracter `cld` corresponde a um nome atribuído à variável para análise. Internamente, apenas é utilizado para prefixar o retorno das informações da `VarInfo`.

xVar (Qualquer) Variável de qualquer tipo a ser examinada

nMargem Numérico Corresponde à margem esquerda inicial de espaços daString de retorno , multiplicado por 5. Default = 0

lHtml Lógico Identifica se a String de retorno será montada em formato Html (.T. / Default) ou ASCII (.F.)

lEcho Lógico Define se o Echo do retorno deve ser enviado ao console do Protheus Server , caso habilitado. (Default = .T.)

O código gerado analisa qual é o caminho disponibilizado no INI do servidor **RootWeb** e a partir desse caminho o sistema ira criar uma pasta **arquivos** onde serão disponibilizados os arquivos que os usuários fizerem o **UPLoad**
Foi utilizado a função `IsSrvUnix` para definir o processo do caminho das pastas Exemplo se utilizaremos "\ ou "/" para chegar no destino.

Utilizamos também **MAKEDIR** e **DIRECTORY** para criar e listar todo o conteúdo da pasta e depois apresentar na pagina **UPLOAD.apf**

<HTML>

```

<HEAD>
<TITLE>UPLOAD DE ARQUIVO</TITLE>
<META name=LEANDRO_DUARTE content=ADVPL-WEB>
</HEAD>
<script language="JavaScript">
function delet(campo,valor){

if (campo.value!=0){
valor=0;
}

campo.value=valor;
}
function OK(){
document.form.action = "u_subir.apw";
document.form.submit()
}
function SAIR(){
document.form.action = "u_TlInWB2.apw";
document.form.submit()
}
</script>
<BODY BGCOLOR="#C0EDC7">
<FORM title=UPLOAD NAME="form" method="GET" action="u_subir.apw" ENCTYPE="multipart/form-data">
<TABLE border=0 cellSpacing=0 cellPadding=0 width="100%">
<TBODY>
<TR>
<TD>EXCLUIR?</TD>
<TD>NOME</TD>
<TD>TAMANHO</TD>
</TR>
<% For nFor := 1 to len(aDir) %>
<TR>
<TD><INPUT TYPE="checkbox" VALUE="0" NAME="EXCLUIR" onclick=javascript:delet(this,'<%= aDir[nFor][1]
%>')></TD>
<TD><a href="<%= aDir[nFor][3] %>"><%= aDir[nFor][1] %></a></TD>
<TD><a href="<%= aDir[nFor][3] %>"><%= aDir[nFor][2] %></a></TD>
</TR>
<% Next nFor %>
</TBODY>
</TABLE>
<TABLE border=0>
<TR>
<TD><BR><BR></TD>
</TR>
<TR>
<TD><BR><BR></TD>
</TR>
<TR>
<TD><BR><BR></TD>
</TR>
</TABLE>

```

```

<TABLE border=1 cellSpacing=0 cellPadding=0 width="100%">
<TBODY>
<TR>
<TD>INFORME O ARQUIVO PARA FAZER O UPLOAD</TD>
<TD>
<INPUT style="WIDTH: 892px; HEIGHT: 22px" name="file" value=" " size=120>
</TD>
</TR>
</TBODY>
</TABLE>
<INPUT TYPE="submit" VALUE="ATUALIZAR" NAME="Atualizar" onclick="JAVASCRIPT:OK()">
<INPUT TYPE="submit" VALUE="Sair" NAME="Sair" onclick="JAVASCRIPT:SAIR()">
</FORM>
</BODY>
</HTML>

```

No código fonte acima representa a pagina HTML com os devidas funções JavaScript (OK, Cancelar e delet)

Para que o sistema exclua o arquivo desejado pelo usuário na pagina criamos uma tag de Check para que o usuário possa selecionar a exclusão do arquivo.

```

<TD><INPUT TYPE="checkbox" VALUE="0" NAME="EXCLUIR" onclick=javascript:delet(this,'<%=
aDir[nFor][1] %>')></TD>

```

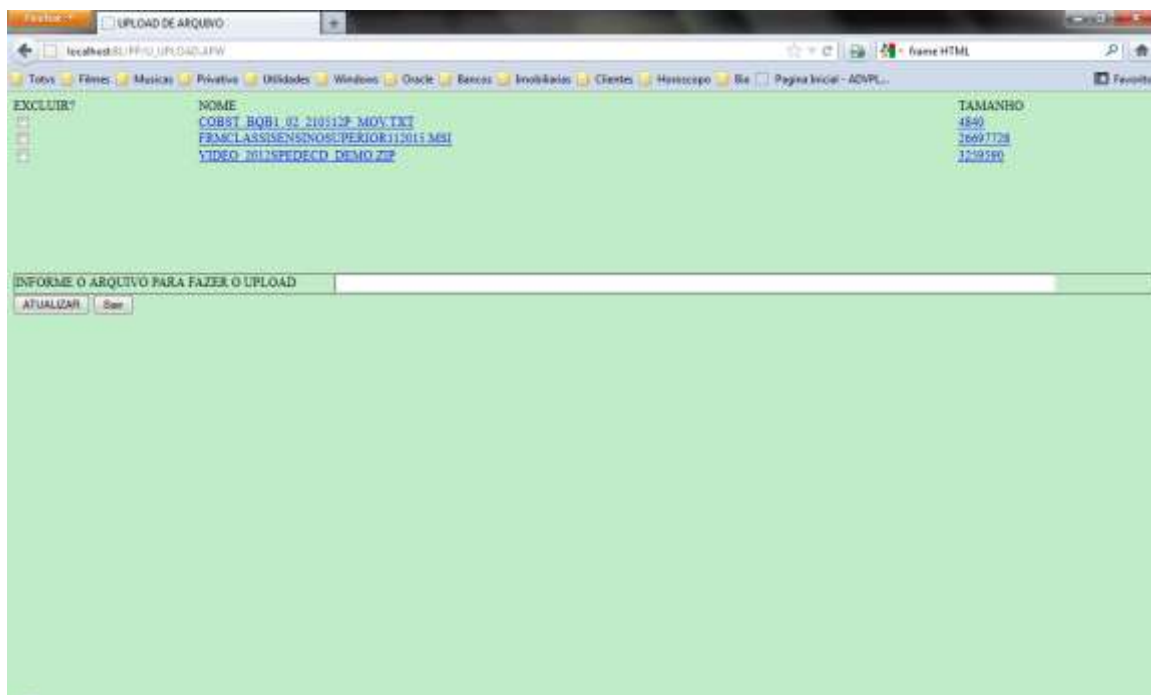
No evento **onclick** colocamos a função javascript para ativar o nome do arquivo que desejamos excluir

Na função Ok colocamos a pagina desejada para fazer o upload do arquivo ou deletar o arquivo desejado

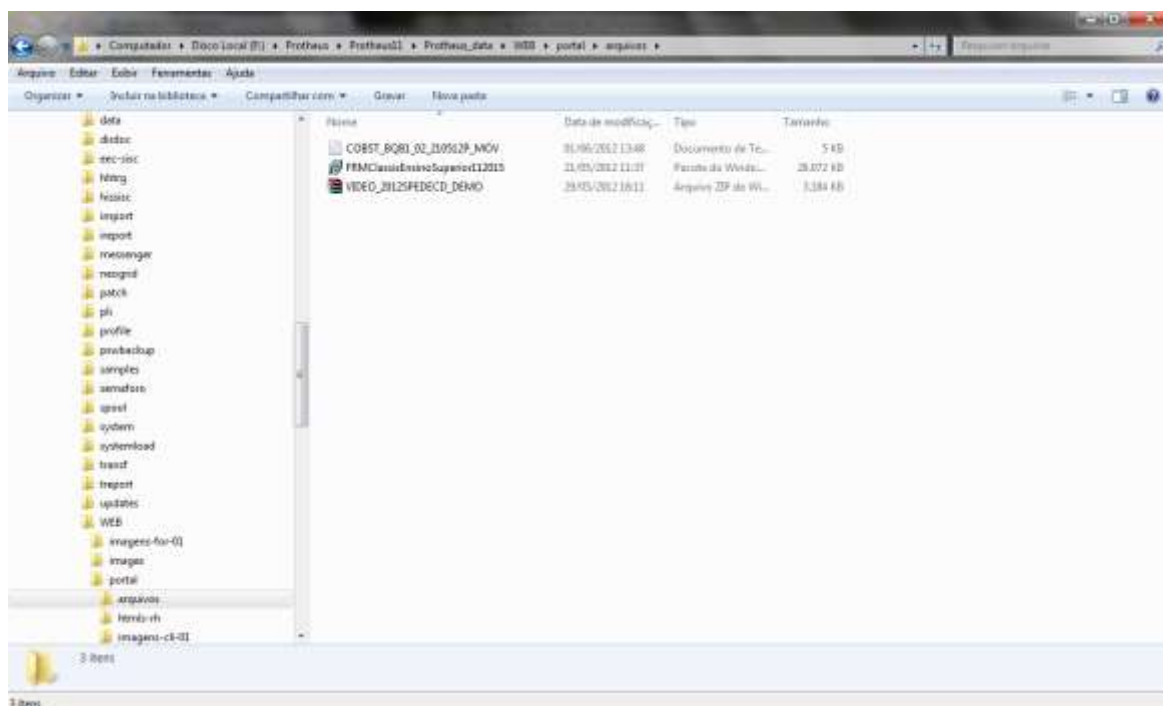
```

function OK(){
document.form.action = "u_subir.apw";
document.form.submit()
}

```



Local dos arquivos



Vamos analisar a pagina/Função **Subir.apw**

```
#INCLUDE "APWEBEX.CH"
#include 'PROTHEUS.CH'

User Function subir()

Local cHtml      := ""
Local cDiretorio := ""
Local cDir1      := ""
Local cArquivo   := ""
Local cNickArq   := ""
Local cNickArqDest := ""
Local cRootWeb   := GetSrvProfString("RootWeb","\web")
Local lUnix      := IsSrvUnix()
Local cCaracter  := ""

Local lStatus    := 0
Local nRetorno   := 0
Local i          := 0
Local nTamRoot   := 0

Private aDir      := {}
Private nTotal    := 0
Private cMsg      := ""

WEB EXTENDED INIT cHtml
conout('cRootWeb')
conout(cRootWeb)
nTamRoot := len(alltrim(cRootWeb))
cCaracter := iif(lUnix,"/","\")

// funcao para descobrir se o ambiente eh linux - IsSrvUnix()

// retirar barra sobressalente
if substr(cRootWeb,nTamRoot,1) == "/" .OR. substr(cRootWeb,nTamRoot,1) == "\"
    cRootWeb := substr(cRootWeb,1,nTamRoot-1)
endif

//colocar o parametro no mesma situacao do que a do windows...
if ! lUnix
    cRootWeb := STRTRAN(cRootWeb,"/","\")
else
    cRootWeb := STRTRAN(cRootWeb,"\""/")
endif

// se nao existir diretorios, cria!
nRetorno := MAKEDIR( cRootWeb+"\arquivos", 1)
cDiretorio := cRootWeb+"\arquivos"
nRetorno := MAKEDIR( cDiretorio , 1 )
```



```

cDiretorio := cDiretorio+"\\"

if !Unix
    cDiretorio := Strtran(cDiretorio,"\\","/")
endif

// :: EXCLUSAO
if HttpGET->EXCLUIR!= nil // solicitacao de exclusao
    conout('entrei')
    conout(HttpGET->EXCLUIR)
    cNickArq := HttpGET->EXCLUIR
    cNickArqDest := Renomeia(cNickArq)
    conout('excluir '+cDiretorio+cNickArqDest)
    fErase(cDiretorio+cNickArqDest,1)
endif

if HttpGET->file != nil
    cArquivo := HttpGET->file
    conout('cArquivo')
    conout(cArquivo)
    //colocar o parametro no mesma situacao do que a do windows...
    if ! Unix
        cArquivo := STRTRAN(cArquivo,"/","\\")
    else
        cArquivo := STRTRAN(cArquivo,"\\","/")
    endif

    CONOUT("*** FILE TO UPLOAD ***")
    CONOUT(cArquivo)

    i := Len(cArquivo)
    While Substr(cArquivo,i,1) != cCaracter // achar a barra de diretorio - ultima
        cNickArq := Substr(cArquivo,i,1)+cNickArq
        i := i - 1
        if i <= 0
            exit
        endif
    End

    cNickArqDest := Renomeia(cNickArq)

    conout("*** UPLOAD FILES ***")
    conout(cArquivo)
    conout(cDiretorio + cNickArqDest)
    conout("*****")

    IStatus := __COPYFILE( cArquivo , cDiretorio + cNickArqDest, 1 )

    if IStatus
        fErase(cArquivo,1)
    endif

```

```

cDiretorio := cRootWeb+"\arquivos\*.*"
aDire := DIRECTORY(cDiretorio, "D", 1 )

i := 0
nTotal := 0

for i := 1 to len(aDir)
    nTotal += aDir[i][2]
next i
For nFor := 1 to len(aDire)
    IF aDire[nFor][2] != 0

        AADD(aDir,{aDire[nFor][1],aDire[nFor][2],"http://localhost:81/PP/arquivos/"+aDire[nF
or][1]})
    ENDIF
Next nFor

cHtml := H_upload()

endif

WEB EXTENDED END
Return cHtml

```

Nesse código fonte temos o processo de capturar o arquivo via o método GET e processar em copiar o arquivo do caminho original do usuário e subir na pasta arquivos

Logo em seguida disponibilizando na variável aDir os arquivos encontrados na pasta

Caso o usuário escolha-se excluir o arquivo o sistema iria analisar qual é o arquivo pela variável HTTPGET->EXCLUIR que vem com o valor o nome do arquivo

```

Static Function Renomeia(cCampo)
Local cAcentos := "ãäåââÁÁÃÃÊÊÊÊÊÍíÓôôÔÔÕúüÚÜÇabcdefghijklmnopqrstuvwxyz ao"
Local cTraducao :=
"AAAAAAAAAAEEEEIIIOOOOOOUUUUCCABCEFGHIJKLMNOPQRSTUVWXYZ_AO"
Local i := 0
for i := 0 to Len(cAcentos)
    cCampo := STRTRAN(cCampo, Substr(cAcentos, i, 1), Substr(cTraducao, i, 1))
Next i
Return cCampo

```

Função para renomear o arquivo onde o arquivo que esta sendo feita a copia não fique com caracter invalido

Exercício

Desenvolver uma página onde esse usuário deverá permitir criar as sub-pastas e inserir arquivos para ele guardar – Upload e depois fazer os devidos Download.

APÊNDICES

12. Funções e Comandos ADVPL

Neste guia de referência rápida serão descritas as funções básicas da linguagem ADVPL, incluindo as funções herdadas da linguagem Clipper, necessárias ao desenvolvimento no ambiente ERP.

INPUT

Descrevendo o elemento de entrada simplesmente não é fácil, pois há muita variação na forma de uma entrada aparece e os atributos que ele usa ou requer dependendo do atributo tipo especificado. Mas seja qual for o tipo, a característica que é comum a todos os elementos de entrada é que eles permitem que os usuários insiram dados.

As características específicas de cada tipo de entrada são cobertas na seção atributo do tipo desta referência. Outros tipos de atributos específicos são indicados como tal em suas próprias seções.

Além dos atributos padrão listados na seção sintaxe, há um outro atributo que você pode encontrar que é IE-specific, autocomplete com valores de "on" ou "off". Como o próprio nome sugere, a intenção é permitir que o navegador lembre e complete os campos do formulário para você. No entanto, porque não é cross-browser seguro e também é ignorada qualquer maneira em determinadas circunstâncias (se a página foi entregue por HTTPS ou foi entregue com cabeçalhos ou uma tag META que impede cache), não é o atributo mais útil / confiável para usar em entradas de formulário.

ACCEPT

O atributo accept é usado para filtrar os tipos de arquivos que podem ser apresentadas através de um upload do arquivo de entrada. Suporte do navegador para este atributo é incompatível, no entanto, o trabalho de validação de uploads de arquivos é melhor deixar para o servidor.

```
<input type="file" name="picture" id="picture" accept="image/jpeg"/>
```

O atributo tem um tipo MIME no tipo de formato / subtipo, por exemplo, "text / html", "image / x-rgb", ou "application / java". Vários valores podem ser aplicados, mas devem ser separados por vírgulas.

ACCESSKEY

O atributo `accesskey` permite ao usuário ativar um controle em uma página usando um atalho de teclado. Isso pode economizar tempo para os usuários que de outra forma, precisam percorrer uma série de controles de formulário ou mover o mouse para chegar ao link desejado. A combinação de teclas que activa a ligação para o qual o `accesskey` é aplicada varia, dependendo da combinação de plataforma e browser. Para o IE / Windows, os usuários pressionem `Alt + accesskey`, enquanto o Firefox / Windows usuários pressionem `Alt + Shift + accesskey`, os usuários da maioria dos navegadores Mac pressionem `Ctrl + accesskey`; no Opera, pressionando `Shift + Esc` exibe uma lista de links para os quais são atributos `accesskey` definido, permitindo aos usuários escolher a chave que deseja utilizar.

De um modo geral, os navegadores não indicam que um atributo `accesskey` é definido para um controle de formulário, e esta falta de descoberta é um problema. O método mais comum para indicar o valor `accesskey` é colocá-lo em um atributo de título do elemento ao qual ela é aplicada. No entanto, para que essa abordagem funcione, o usuário deve mouse sobre o elemento em questão. Você pode querer indicar o valor `accesskey` de alguma outra maneira, por exemplo:

```
<label for="firstname">First name [access key = f]</label>
<input type="text" name="firstname" id="firstname" accesskey="f"/>
```

ALIGN

Em vez de usar um botão para enviar detalhes do formulário, você pode querer usar uma imagem (ver imagem de entrada para mais detalhes). Assim como você pode alinhar o elemento `img` em HTML, você fazer o mesmo por um usado como entrada. No entanto, a utilização deste atributo não é recomendado, em vez disso, usar CSS para alinhar a imagem. Sem CSS, você tem controle limitado sobre o projeto, como a Figura 1 mostra, onde o texto é disposto ao redor da imagem pouco atraente.

```
<form>
:
<input type="image" src="submit.jpg" alt="Submit your details"
  align="right"/>
<p>This a load of explanatory text that we need at the end of the
  form. This a load of explanatory text that we need at the end of
  the form. This a load of explanatory text that we need at the end
  of the form. </p>
</form>
```

Os valores possíveis para esse atributo incluem "fim", "esquerda", "meio", "direita" e "top". O navegador padrão é "esquerda".

CHECKED

Em algumas circunstâncias, pode ser necessário apresentar uma ou mais caixas de tal forma que o estado do controle no ponto de carregamento da página está marcada, e o usuário tem de optar por sair ao invés de optar dentro Ou talvez você está apresentando o usuário com um formulário, incluindo uma série de caixas de seleção, que ele ou ela tenha previamente preenchido, e você quer preservar o estado das seleções que foram feitas. O atributo `checked` permite que você defina o estado selecionado para "on" por padrão.

O atributo verificado também é usado para os controles de entrada de rádio. Ao contrário da caixa de seleção, apenas um em uma série de entradas de rádio relacionados podem ser selecionadas (veja a seção sobre entradas de rádio no tipo para saber mais sobre isso), e o atributo verificado é usado para identificar qual deles está selecionado.

Se você acidentalmente marcar como "verificados" um número de entradas de rádio que compartilham um atributo nome dado, o último que está marcado como tal será selecionado.

Note-se que a aparência da caixa de seleção é diferente entre os navegadores e sistemas operacionais de em alguns, a caixa de seleção marcada aparece como uma caixa assinalada ou marcada, enquanto em outros, é uma caixa cruzado. A entrada de rádio que está "marcada" quase certamente não parece um carrapato ou um cheque, mas não vamos ficar presos em aparências. A chave é saber que o direito de controle é selecionado!

```
<form>
  <input type="checkbox" name="chknewsletter" id="chknewsletter"
    checked="checked"/>
  <label for="chknewsletter">Send me the monthly newsletter</label>
</form>
```

DISABLED

O atributo desativado impede que o usuário interagir com o controle de forma a que ele é aplicado. Neste caso, ele pára de o usuário clicar no checkbox, ou tabulações para com o teclado, para que o usuário não pode ativar o controle.

O uso mais provável para este atributo é desativar uma caixa de seleção até o momento em que alguma outra condição foi atendida, em que ponto JavaScript seria necessária para remover o valor " desativado", tornando o controle de formulário utilizável novamente.

```
<form>
  <input type="checkbox" name="chknewsletter" id="chknewsletter"
    checked="checked" disabled="disabled"/>
  <label for="chknewsletter">Send me the monthly newsletter</label>
</form>
```

" desativado" é apenas um valor possível para este atributo. Se a entrada deve ser ativado, simplesmente omitir o atributo inteiramente.

MAXLENGTH

Embora seja possível usar o JavaScript para validar a entrada de dados do formulário, esta abordagem não pode ser totalmente invocado (JavaScript pode ser desativado, e os dados do formulário não pode ser confiável se o cliente tem sido afetada por um script nefasto).

Do lado do servidor de validação é uma opção muito mais seguro, e poderia ser utilizado, entre outras coisas, para verificar que a quantidade de dados apresentados não é muito longo.

O atributo maxlength proporciona um meio para reduzir, mas ainda não completamente eliminando-a probabilidade de que muitos dados serão enviados para o servidor para processamento.

É principalmente um aperfeiçoamento de usabilidade, indicando para os usuários quando eles atingiram o comprimento máximo de caracteres para uma entrada, evitando assim a chance de escrever um ensaio só deve ser dito mais tarde que não há uma contagem de caracteres máximo de 4.

```
<form>
  <label for="pin">Your 4-digit PIN:</label>
  <input type="password" name="pin" id="pin"
    maxlength="4" size="6"/>
  :
</form>
```

O valor para este atributo é um número que representa o total de caracteres que podem ser apresentados.

NAME

O atributo nome é usado para fazer referência a dados do formulário depois que ele é submetido, e para fazer referência aos dados usando JavaScript no lado do cliente.

Ao contrário do atributo id, que deve ser dado um valor único de cada vez que é aplicado a um controle nova forma, um atributo de nome com um dado valor pode ser aplicada a controles de formulário numerosos (embora na prática, esta abordagem é sempre apenas observada em uso com rádio entrada de botões).

Observe que os elementos única forma que têm um atributo de nome terão seus valores passados para a página ou o script especificado no atributo do formulário ação.

```
<form>
  <label for="pin">Your 4-digit PIN:</label>
  <input type="password" name="pin" id="pin" maxlength="4" size="6"/>
  :
</form>
```

Este atributo tem qualquer nome que o desenvolvedor escolhe, contanto que ele não contém quaisquer espaços ou caracteres especiais.

READONLY

O atributo "somente leitura" pára o usuário altere o valor de uma entrada (no caso de uma entrada de texto ou senha), mas não impede que o usuário interaja com o conteúdo controle de formulário.

É ainda possível para o usuário clicar dentro da guia de entrada, para isso, destacar o texto dentro dele, e até mesmo copiar e colar esse conteúdo é apenas que o valor não pode ser alterado ou cancelado.

Este atributo é mais comumente usado para parar o usuário de interferir com o valor de uma entrada de texto até que alguma outra condição foi atendida (por exemplo, até uma caixa de seleção está marcada para confirmar a aceitação do usuário dos termos e condições).

Nesse ponto, JavaScript seria necessário para a retirada "readonly" valor, tornando o controle de forma completamente usável.

```
<form>
  <label for="town">Postal Town:</label>
  <input type="text" name="town" id="town" readonly="readonly"
    value="Southampton"/>
</form>
```

"readonly" é o único valor possível para este atributo.

SIZE

O atributo de tamanho é usado para definir a largura de um campo de entrada de senha de texto ou arquivo. O comprimento destes campos é determinada pelo número de caracteres que devem ser visíveis para todos os outros tipos de entrada, o tamanho refere-se à largura do controle em pixels.

Dada a natureza de apresentação desse atributo, normalmente é melhor evitar.

Em vez disso, use CSS para definir as larguras de campo com medições mais precisas, com exceção de um arquivo de entrada onde CSS controle sobre a aparência da entrada é muito limitada (por razões de segurança muito válidos) eo atributo de tamanho é a sua única opção real.

```
<form>
  <label for="pin">Your 4-digit PIN:</label>
  <input type="password" name="pin" id="pin" maxlength="4" size="6"/>
  :
</form>
```

Este atributo tem um número que reflete a largura do campo em caracteres, por exemplo, "5", "10", e assim por diante.

SRC

O atributo src diz ao navegador para onde olhar para a imagem que deverá ser apresentada para o usuário. Esta imagem pode ser localizado no mesmo diretório que a página que está referenciando o arquivo, em outro diretório no mesmo servidor, ou em outro servidor por completo.

O exemplo especifica uma imagem que está localizado no mesmo diretório que a página web que está chamando, mas se a imagem submit foi localizado em um diretório que estava um nível acima do que o documento referência, a sintaxe seria a seguinte:

```
<input type="image" src="../submit.jpg" alt="Submit your details"/>
```

Aqui, ../ equivale a "subir um diretório na hierarquia."

Você também pode referenciar uma imagem relativa à raiz do site (a pasta ou arquivo após o nome de domínio):

```
<input type="image" src="/submit.jpg" alt="Submit your details"/>
```


Este atributo de marcação, basicamente, diz, "exibir o submit.jpg imagem que pode ser encontrado em www.TOTVS.com/." Esta é uma forma muito útil de arquivos de referência, como você pode mover o documento que contém a imagem enviar para outro local o sistema de arquivos sem ter que mudar o link.

Se você estivesse fazendo referência a uma imagem que é realizada em outro servidor, você expressar o src usando um URL completo, como segue:

```
<input type="image" src="http://www.TOTVS.com/submit.jpg"
      alt="Submit your details"/>
```

Este atributo tem como seu valor a localização da imagem do botão em relação ao documento referência, em relação à raiz do servidor, ou como um URI completo contendo `http://` ou `https://`, o nome do servidor e do caminho para o documento nesse servidor.

TABINDEX

A tabindex é usado para definir uma sequência que os usuários devem seguir quando usar a tecla Tab para navegar por uma página.

Por padrão, a ordem natural de tabulação irá coincidir com a ordem de origem na marcação.

Em determinadas circunstâncias pode ser necessário para substituir a ordem padrão de tabulação, mas é fortemente recomendado que você criar uma página em um fluxo lógico e deixar o navegador trabalhar com ele no padrão abordagem fim-um que nega a necessidade para o atributo tabindex.

A tabindex pode começar em 0 e incrementa em qualquer valor.

Como tal, a sequência de 1, 2, 3, 4, 5 seria fino, tal como seria de 10, 20, 30, 40, 50. Se você precisa de introduzir uma tabindex, é aconselhável a utilização de uma sequência que contém intervalos (como o segundo exemplo fornecido), pois isso lhe dará a oportunidade de injetar outros controles mais tarde se for necessário (por exemplo, 10, 15, 20) sem ter de reindexar todos os valores tabindex na página.

Se um valor tabindex dado ser aplicada a mais de um elemento (por exemplo, todos os links em uma determinada seção de um tabindex "1", e links da barra lateral dado um tabindex de "2"), a ordem de tabulação dos elementos afetados será de acordo com a ordem de marcação de origem.

Muitas pessoas vão optar por usar essa abordagem, em vez de uma sequência com um intervalo definido, como 5, 10, 15, pois permite ligações adicionais ou controles de formulário a ser adicionados sem a dor de cabeça de re-numeração.

Se um tabindex é colocado em qualquer lugar em uma página, mesmo se é o link centésimo quinquagésimo ou a forma de controle a ordem de tabulação terá início no elemento com o menor valor tabindex, e trabalhar com os incrementos.

Só então a ordem de tabulação tomar os restantes elementos para os quais não tabindex foi definido.

Como tal, muito cuidado deve ser tomado para assegurar que a adição de um tabindex não prejudicar a usabilidade da página como um todo.

Se o atributo `disabled` é definida em um elemento que tem uma `tabindex`, que `tabindex` será ignorado.

```
<form>
:
<label for="pin">Your 4-digit PIN:</label>
<input type="password" name="pin" id="pin" tabindex="2"/>
:
</form>
```

Este atributo pode tomar apenas um número como o seu valor.

TYPE

De todos os atributos que podem ser aplicados a uma entrada, o atributo de tipo é aquele que tem o maior impacto.

Com uma mudança deste valor, a aparência eo comportamento da entrada pode mudar drasticamente, a partir de uma entrada de texto simples ou um campo de senha mascarado, para controles, como botões, caixas, e até mesmo imagens.

Os valores possíveis e suas aplicações são detalhadas na lista abaixo:

"botão"

Este é um simples botão clicável que não faz nada por si só (ao contrário de um botão Enviar, que envia os dados do formulário para a localização definida em ação).

O tipo de botão de entrada é normalmente usado junto com o JavaScript para desencadear algum tipo de comportamento (por exemplo, usando um botão para abrir uma página de ajuda em uma nova janela).

Aqui está uma definição do botão:

```
<input type="button" value="Click me. AGORA!!!" />
```

"checkbox"

Este tipo de entrada é usada quando você precisa de usuários para responder a uma pergunta com um sim ou não resposta.

Não há nenhuma maneira que os usuários podem inserir os dados de sua própria sua ação é simplesmente um caso de marcando ou desmarcando a caixa.

Quando o formulário é enviado, o servidor estará interessado no valor do atributo anexado ao controle, e o estado da caixa de seleção (marcado ou desmarcado).

Quando estiver especificando uma entrada caixa, o controle deve estar em primeiro lugar, e ser seguido pelo texto associado, como mostrado na Figura 1:

```
<input type="checkbox" name="chknewsletter" id="chknewsletter"
value="newsletter-opt-in"/>
<label for="chknewsletter"> Envie-me o boletim mensal </label>
```

“FILE”

Este tipo de entrada é usado para upload de arquivos.

Quando o atributo é definido como "arquivo", dois controles aparecem no navegador: um campo que é semelhante a uma entrada de texto, e um controle de botão que está marcado Browse

O texto que aparece neste botão não pode ser mudado, e um arquivo de controle de upload é geralmente difícil de estilo com CSS.

O arquivo de controle de carregamento tem um atributo opcional, aceite, o qual é utilizado para definir os tipos de ficheiros que podem ser enviados.

```
<input type="file" name="picture" id="picture"
accept="image/jpeg, image/gif"/>
```

“HIDDEN”

Uma entrada de forma oculta é uma que, embora não aparecem na página, pode ser usado para armazenar um valor.

O valor em um campo oculto pode ser definido na página carga tempo (que pode ser escrito usando server-side scripting, como no exemplo abaixo), ou pode ser inserido ou alterado de forma dinâmica, através de JavaScript.

Aqui está um exemplo em que o elemento de entrada "escondido" obtém seu valor de uma variável

```
<input type="hidden" name="hdnCustId" value="<%= RECNO %>"/>
```

“IMAGE”

O tipo de entrada de imagem geralmente é usado no lugar de um botão de envio padrão, que pode ser um pouco chato e conservador para seu projeto de forma (ou talvez o seu departamento de marketing está exigindo algo um pouco mais "na marca").

Em alguns casos, você poderia, em vez usar um botão de submit padrão e estilo-lo em conformidade com CSS (cor de fundo, etc estilo da fonte), mas a quantidade de controle que você tem em estilizar botões de envio depende do navegador (Safari no Mac é um caso em ponto).

Mas quando o submit CSS de estilo só não é cortá-lo, você pode precisar usar uma entrada de "imagem" do tipo.

Ao usar uma entrada de "imagem" tipo, você também precisará especificar um src (onde a imagem pode ser encontrado) e um atributo alt (texto alternativo para a imagem, no caso que está faltando ou não pode ser visualizado por qualquer outro motivo).

Finalmente, quando se usa este tipo de entrada, não é só esta mudança como o controle olha para o usuário, ele também está passando por informações adicionais para o servidor de onde a imagem que o usuário clica (com x, y coordenadas como uma imagem mapa), deve ser útil que a informação ou crítica à forma como você processa a forma como um resultado.

```
<input type="image" src="submit.jpg" alt="Submit your details"/>
```

“PASSWORD”

O campo de senha é quase idêntico ao de entrada de texto padrão:

```
<input type="password" name="pin" id="pin" maxlength="4" size="6"/>
```

A entrada de senha tem exactamente o mesmo conjunto de atributos como a entrada de texto, mas difere visualmente em que os caracteres introduzidos neste tipo de campo são mascarados no navegador, de modo que caracteres inseridos aparecem como asteriscos ou gotas

“RADIO”

O “rádio” controle de botão é sem dúvida o tipo mais complexo de controle de entrada.

Para explicar como esse controle funciona, vale a pena pensar sobre a origem de seu nome.

Em conjuntos antigos de rádio, sintonizando entre as estações não foi alcançado usando o “scan” instalação, ou até mesmo rodar um mostrador.

Em vez disso, uma série de botões que estavam ligados a um punhado de estações de rádio podiam ser escolhidos, um de cada vez, pressionando um botão na faixa causaria qualquer outro botão para aparecer.

O mesmo efeito é em jogo aqui. Apenas um controle pode ser selecionado a partir de um intervalo, e se você selecionar outra, a entrada previamente selecionada é desmarcada.

Para este controle funcione, porém, cada botão de rádio no intervalo que você deseja que os usuários escolham a partir de deve ter o atributo mesmo nome, como mostrado abaixo.

Com uma entrada de rádio, o controle deve estar em primeiro lugar, seguido do texto associado:

```
<div><input type="radio" name="station" id="rad1"
  value="Radio 1"/> <label for="rad1">Radio 1</label></div>
<div><input type="radio" name="station" id="rad2"
  value="Radio 2"/> <label for="rad2">Radio 2</label></div>
<div><input type="radio" name="station" id="rad3"
  value="XFM"/> <label for="rad3">XFM</label></div>
<div><input type="radio" name="station" id="rad4"
  value="4Music"/> <label for="rad4">4Music</label></div>
```

Porque cada controle tem o mesmo nome (neste caso, “estação”), apenas um poderá ser escolhido a partir da lista.

O valor para cada controle diferente, embora, por isso se os usuários escolheu “XFM” como sua estação de rádio favorita no exemplo acima, que seria o valor associado ao nome do campo de “estação”.

Observe também que, ao contrário de todos os exemplos de marcação outra forma, os valores de nome e ID diferente para este tipo de controle.

A ID é utilizado para efeitos de ligação do controle para a sua texto da etiqueta, e deve ser exclusiva.

“RESET”

A entrada de “reset” é visualmente parecido com o “botão” e “enviar” tipos—é como um controle de botão que pode ser clicado ou pressionado, e não aceita os dados do usuário.

No entanto, um controle de redefinição é um controle muito destrutivo para apresentar a uma página web, como a pressão que irá limpar todos os dados já inseridos no formulário em que o botão de reset reside, geralmente

causando uma certa quantidade de raiva usuário! Embora o uso desse controle constitui HTML perfeitamente válido, ele é usado cada vez menos nos dias de hoje, como é muito fácil para os usuários acidentalmente a ação do botão de reset com qualquer um clique do mouse ou enquanto tabulação através da página com o teclado.

Se você sente que um botão de "reset" não se justifica, acrescentando considerar alguns JavaScript para a página algo que pode verificar novamente a pressionar o botão com uma mensagem que pergunta:
"Você tem certeza que quer apagar tudo e começar de novo ? "

No entanto, é provavelmente a melhor maneira de evitar essa entrada, permitindo aos usuários facilmente atualizar ou recarregar uma página para começar de novo, se quiserem.
A marcação a seguir mostra um botão de reset:

```
<input type="reset" name="cmdReset" id="cmdReset"
value="Clear form data"/>
```

“SUBMIT”

Esta é a magia botões o que os usuários pressionam para enviar todos os dados do formulário que eles entraram.

Uma vez que é clicado, não há como voltar atrás!

O "enviar" input mostra o texto que você especifique no atributo de valor, mas se nenhum valor for especificado, a face do botão simplesmente exibir a palavra em Enviar.

A marcação a seguir cria um botão de envio que exibe as palavras Enviar esta aplicação:

```
<input type="submit" name="cmdSubmit" id="cmdSubmit"
value=" Enviar esta aplicação" />
```

“TEXT”

Este é o tipo mais comum de entrada que você vai encontrar na Web, e que você precisa na maioria das vezes como você está construindo suas próprias formas.

O "texto" cria uma caixa de entrada de linha única que o usuário pode inserir texto em, e tem um número de atributos, como tamanho, maxlength, deficientes, readonly, nome, e tabindex.

VALUE

O valor de atributo é usado e apresentado de forma diferente dependendo do tipo de controle de forma a que ela é aplicada, tal como detalhado abaixo: type = "button", "enviar", "reset"

O texto indicado no atributo valor é usado como o texto do botão, e não pode ser alterado pelo utilizador (embora possa ser alterado dinamicamente via JavaScript). type = "text", "password"

O valor será exibido dentro do controle de formulário e pode ser overtyped, copiado ou cortado pelo usuário. Observe que, se ele é exibido no campo de senha, será ofuscado, mas o número correto de caracteres irá aparecer, no entanto. type = "radio", "checkbox", "imagem", "escondido"

O valor não será exibido para o usuário, nem pode ser mudado, mas será associado ao controle, e é o valor que é repassado quando o formulário é enviado.

Note-se que o gráfico de propriedades não mostra o valor de atributo como sendo necessário para todos os tipos de entrada, mas no caso de tipos de entrada de rádio e de caixa, o valor é necessária.

Eventos

Onkeypress

O evento onkeypress executa um script quando uma tecla alfanumérica é pressionada;

Onchange

O evento onchange executa um script quando o valor de um elemento é modificado

onclick

O evento onclick é ativado em um clique do mouse

Apos identificar todos os tipos de tag utilizados no HTML iremos analisar o fonte/pagina xTelaA.apw que chama a pagina h_xTelaA()

```
<HTML>
<HEAD>
<TITLE></TITLE>
<META NAME="ADVPL-WEB" CONTENT="ADVPL-WEB">
</HEAD>
<script language="JavaScript">

function SomenteNumero(e){
    var tecla=(window.event)?event.keyCode:e.which;
    if((tecla > 47 && tecla < 58) || tecla == 46 || tecla == 44)
    {
        return true;
    }
    else
    {
        if (tecla != 8 )
        {
            return false;
        }
        else
        {
            return true;
        }
    }
}

function upperCase(campo,valor){
    campo.value = valor.toUpperCase()
}

function validaDat(campo,valor) {
```

```

var date=valor;
var ardt=new Array;
var ExpReg=new RegExp("(0[1-9][12][0-9]3[01])/(0[1-9]1[012])/[12][0-9]{3}");
ardt=date.split("/");
erro=false;
if ( date.search(ExpReg)==-1){
    erro = true;
}
else if (((ardt[1]==4)||(ardt[1]==6)||(ardt[1]==9)||(ardt[1]==11))&&(ardt[0]>30))
    erro = true;
else if ( ardt[1]==2) {
    if ((ardt[0]>28)&&((ardt[2]%4)!=0))
        erro = true;
    if ((ardt[0]>29)&&((ardt[2]%4)==0))
        erro = true;
}
if (erro) {
    alert(""" + valor + "" não é uma data válida!!! utilize dd/mm/aaaa");
    campo.focus();
    campo.value = " / / ";
    return false;
}
return true;
}

function cancelar(){
    javascript:window.close();
}

function ok(){
    document.alterar.action = "u_gravar.apw";
    document.alterar.submit()
}

</script>
<BODY BGCOLOR="#E6EEED">
    <FORM ACTION="" METHOD="POST" TITLE="alterar" name="alterar" id="alterar" >
        <TABLE BORDER="0" WIDTH="100%" CELLPADDING="0" CELLSPACING="0">
            <% For nFor := 1 to len(aDados2) %>
                <%= aDados2[nFor] %>
            <% next nFor %>
            <% For nFor := 1 to len(aDados1) %>
                <%= aDados1[nFor] %>
            <% next nFor %>
        </TABLE>
        <INPUT TYPE="submit" VALUE="GRAVAR" NAME="SALVAR" onclick="javascript:ok()">
        <INPUT TYPE="submit" VALUE="CANCELAR" NAME="SAIR" onClick="javascript:cancelar()">
    </FORM>
</BODY>
</HTML>

```


Conforme a análise do código fonte observamos 5 funções JavaScript

```
function SomenteNumero(e){
    var tecla=(window.event)?event.keyCode:e.which;
    if((tecla > 47 && tecla < 58) || tecla == 46 || tecla == 44)
    {
        return true;
    }
    else
    {
        if (tecla != 8 )
        {
            return false;
        }
        else
        {
            return true;
        }
    }
}
```

Função que permite somente a inclusão de número e pontuação

```
function upperCase(campo,valor){
    campo.value = valor.toUpperCase()
}
```

Função que transforma qualquer caractere em maiúsculo

```
function validaDat(campo,valor) {
    var date=valor;
    var ardt=new Array;
    var ExpReg=new RegExp("(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[012])/[12][0-9]{3}");
    ardt=date.split("/");
    erro=false;
    if ( date.search(ExpReg)==-1){
        erro = true;
    }
    else if (((ardt[1]==4)||((ardt[1]==6)||((ardt[1]==9)||((ardt[1]==11))&&(ardt[0]>30))
        erro = true;
    else if ( ardt[1]==2) {
        if ((ardt[0]>28)&&((ardt[2]%4)!=0))
            erro = true;
        if ((ardt[0]>29)&&((ardt[2]%4)==0))
            erro = true;
    }
    if (erro) {
        alert("\n" + valor + "\n não é uma data válida!!! utilize dd/mm/aaaa");
        campo.focus();
        campo.value = " / / ";
    }
}
```

```

        return false;
    }
    return true;
}
    
```

Função que valida a data digitada

```

function cancelar(){
    javascript:window.close();
}
    
```

Função que apenas fecha a janela

```

function ok(){
    document.alterar.action = "u_gravar.apw";
    document.alterar.submit()
}
    
```

CLOSE QUERY

Sintaxe: CLOSE QUERY cAlias

Parâmetros:

Argumento	Tipo	Descrição
cAlias	Caracter	Alias sob o qual o cursor da Query foi aberto. Caso o alias passado como parâmetro não se encontre aberto , a função não gera nenhuma ocorrência de erro.

Descrição

Através do comando Close Query , realizamos o fechamento de uma query aberta através do comando OPEN QUERY.

ATENÇÃO : Uma query aberta pelo comando OPEN QUERY deve ser fechada pelo comando CLOSE QUERY . Poderíamos fechar o alias aberto através de uma Query simplesmente com a função DbCloseArea(), porém isto deixaria em aberto elementos internos de controle criados pelo comando OPEN QUERY.

OPEN QUERY

Sintaxe: OPEN QUERY <cQuery> ALIAS <cAlias> [[NOCHANGE]]

Parâmetros

Argumento	Tipo	Descrição
<cQuery>	Caracter	cQuery corresponde à String contendo a Query a ser executada no banco de dados
ALIAS <cAlias>	Caracter	cAlias corresponde ao nome do alias sob o qual o cursor de retorno dos dados pesquisados será aberto no ambiente Advpl. Não pode ser especificado um nome de alias já em uso, senão a aplicação será finalizada com a ocorrência de erro "Alias already in Use"
[NOCHANGE]	Caracter	Caso especificada a cláusula NOCHANGE na abertura da query , a string em cQuery não será submetida à função ChangeQuery()

Descrição

Através do comando OPEN QUERY , realizamos a abertura de uma Query de busca no Banco de Dados através do RDD TOPCONN, retornando os dados consultados através de um 'ALIAS' Advpl.

Caso a Query nao possa ser aberta, por erro de sintaxe , devido à thread atual não estar conectada com o TopConnect , ou outro erro , será gerado um log de erro , informando o Alias , o Stack (Pilha de Chamadas) de execução , e o conteúdo da Query para Debug.

OBSERVAÇÕES IMPORTANTES

- Na montagem da string da Query , devemos especificar os comandos SQL , alias e nomes de campos em letras maiúsculas.
- Quando utilizamos o comando OPEN QUERY , não precisamos passar a expressão da Query através da função ChangeQuery(). Este procedimento já é realizado internamente pelo comando OPEN QUERY. Para que a query não seja submetida à função ChangeQuery(), devemos utilizar o parâmetro NOCHANGE.
- A utilização deste comando é implícita à LIB APWEBEX , e necessita da utilização do #include 'Apwebex.ch'

WEB EXTENDED END

Sintaxe: WEB EXTENDED END <cHtml> [START <cFnStart>]

Parâmetros

Argumento	Tipo	Descrição
<cHtml>	Caracter	cHtml corresponde à variavel que será utilizada para armazenar a String Html que será retornada ao Browser solicitante do processamento. Deve ser especificada uma variável String , com conteúdo vazio. ("")
START <cFnStart>	Caracter	cFnStart corresponde ao nome de uma função Advpl que será executada

para pré-validar a execução do resto do código. A função deve ser passada SEM parênteses () .

Descrição

Devemos utilizar este comando para fechar uma seção aberta pelo comando WEB EXTENDED INIT . Para cada ocorrência do comando WEB EXTENDED INIT , deve-se ter um fechamento da mesma através do comando WEB EXTENDED END , devendo haver apenas uma ocorrência desta estrutura por função.

A utilização deste comando é implícita à Working Threads inicializadas pela Lib APWEBEX , e a definição do mesmo está no arquivo #include 'apwebex.ch' , que deve ser declarado no início do arquivo fonte Advpl.

WEB EXTENDED INIT

Sintaxe: WEB EXTENDED INIT <cHtml> [START <cFnStart>]

Parâmetros

Argumento	Tipo	Descrição
<cHtml>	Caracter	cHtml corresponde à variável que será utilizada para armazenar a String Html que será retornada ao Browser solicitante do processamento. Deve ser especificada uma variável String , com conteúdo vazio. ("")
START <cFnStart>	Caracter	cFnStart corresponde ao nome de uma função Advpl que será executada para pré-validar a execução do resto do código. A função deve ser passada SEM parênteses () .

Descrição

Devemos utilizar este comando , juntamente com o comando WEB EXTENDED END quando montamos uma função (Web Function) que foi construída para ser chamada a partir de um Web Browser , quando nos utilizamos das funções de Infra-Estrutura APWEBEX.

Através dele , é realizada uma pré-validação que certifica que a execução da função somente será realizada caso a thread atual seja realmente uma Thread montada no ambiente WEBEX, além de podermos inserir uma pré-validação (START) de execução específica para esta função.

Caso seja especificada uma função na cláusula START, a mesma deverá retornar uma String. Retornando uma String em branco , o processamento da função original será efetuado normalmente . Caso a função retorne uma string não-vazia , esta string será retornada para a variável cHtml , e o processamento do programa será desviado para a linha do código-fonte imediatamente posterior ao comando WEB EXTENDED END .

Para cada ocorrência do comando WEB EXTENDED INIT , deve-se ter um fechamento da mesma através do comando WEB EXTENDED END , devendo haver apenas uma ocorrência desta estrutura por função.

A utilização deste comando é implícita à Working Threads inicializadas pela Lib APWEBEX , e a definição do mesmo está no arquivo #include 'apwebex.ch' , que deve ser declarado no início do arquivo fonte Advpl.

Exemplo da função ESCAPE

Nos exemplo abaixo, utilizamos a função escape() para formatar parâmetros para inserir em uma URL.

```
cUrl := 'http://localhost/webinfo.apw'

cPArAm1 := 'Teste de Parametro 01-02'
cPArAm2 := '#reserva#'
cPArAm3 := '1+2+3'

cUrl += '?Par01=' + escape(cPArAm1) + '&Par02=' + escape(cPArAm2) + '&Par03=' + escape(cPArAm3)

// O conteudo de cUrl deverá ser "http://localhost/webinfo.apw?Par01=Teste%20de%20Parametro%2001-02&Par02=%23reserva%23&Par03=1%2B2%2B3", próprio para a monyagem de um link .
```

Exemplo da função GETJOBPROFSTRING

O exemplo abaixo , executado em uma thread iniciada a partir de um JOB WEBEX, recupera algumas configurações atuais em uso para este JOB.

```
cJobType := GetJobProfString('type','(empty)' )
cInstances := GetJobProfString('Instances','(empty)' )
cInacTime := GetJobProfString('InactiveTimeout','(default)' )
cExpTime := GetJobProfString('ExpirationTime','(default)' )
```

Exemplo da função GETPARVALUE

No exemplo abaixo , são montados dois arrays multi-dimensionais , com 2 dimensões , e são realizadas buscas nos mesmos explorando todas as possibilidades de uso da função GetParValue()

```
Local aTeste1 := {}
Local aTeste2 := {}

Aadd(aTeste1,{"Alias","TMP1"})
Aadd(aTeste1,{"Relacao","2x3"})

Aadd(aTeste2,{"Alias","TMP2"})
Aadd(aTeste2,{"Info","---Informação adicional---"})

// Busca apenas no array ateste1
cAlias := GetParValue("ALIAS",aTeste1)
cRelacao := GetParValue("RELACAO",aTeste1)
cInfo := GetParValue("INFO",aTeste1)

DEFAULT cAlias := "(nao encontrado)"
DEFAULT cRelacao := "(nao encontrado)"
```

```

DEFAULT cInfo := "(nao encontrado)"

conout(cAlias) // TMP1
conout(cRelacao) // 2x3
conout(cInfo) // (nao encontrado)

// Busca apenas no array ateste2
cAlias := GetParValue("ALIAS",aTeste2)
cRelacao := GetParValue("RELACAO",aTeste2)
cInfo := GetParValue("INFO",aTeste2)
DEFAULT cAlias := "(nao encontrado)"
DEFAULT cRelacao := "(nao encontrado)"
DEFAULT cInfo := "(nao encontrado)"
conout(cAlias) // TMP2
conout(cRelacao) // (nao encontrado)
conout(cInfo) // ---Informação Adicional---

// Busca em ambos os Arrays
// Primeiro no aTeste1 e depois no aTeste2
cAlias := GetParValue("ALIAS",aTeste1,aTeste2)
cRelacao := GetParValue("RELACAO",aTeste1,aTeste2)
cInfo := GetParValue("INFO",aTeste1,aTeste2)

conout(cAlias) // TMP1
conout(cRelacao) // 2x3
conout(cInfo) // ---Informação Adicional---

```

Exemplo da função HEXSTRDUMP

Através do exemplo abaixo, geramos a string com o DUMP de um arquivo HTML, salvo na pasta WEB a partir do RootPath do Environment. O Dump do arquivo será mostrado no Console do servidor e no Web Browser que solicitou a função U_DumpTest.apw .

```

#include "protheus.ch"
#include "apwebex.ch"

User Function DumpTest()
Local cHtml := ""
Local cTXTFile := ""
Local cDump := ""

WEB EXTENDED INIT cHtml

// Le o arquivo
cTXTFile := memoread("\Web\Default.htm")

// Gera a string com o Dump do arquivo
cDump := HExStrDump(cTXTFile)

```

```
// Mostra o Dump no console
conout(cDump)

// Gera HTML para a visualização do DUMP
cHtml := VarInfo('DUMP',HtmlNotags(cDump),,t.,f.)
```

WEB EXTENDED END

Return cHtml

/*

Exemplo do Texto mostrado no Console

HexSTRDump (String 237 / Start 1 / Length 237)

```
-----
3C 48 54 4D 4C 3E 3C 48 45 41 44 3E 0D 0A 3C 4D | <HTML><HEAD>__<M
45 54 41 20 48 54 54 50 2D 45 51 55 49 56 3D 22 | ETA HTTP-EQUIV="
43 6F 6E 74 65 6E 74 2D 54 79 70 65 22 20 63 6F | Content-Type" co
6E 74 65 6E 74 3D 22 74 65 78 74 2F 68 74 6D 6C | ntent="text/html
22 0D 0A 3C 4D 45 54 41 20 48 54 54 50 2D 45 51 | " __<META HTTP-EQ
55 49 56 3D 22 70 72 61 67 6D 61 22 20 63 6F 6E | UIV="pragma" con
74 65 6E 74 3D 22 6E 6F 2D 63 61 63 68 65 22 3E | tent="no-cache">
0D 0A 3C 4D 45 54 41 20 48 54 54 50 2D 45 51 55 | __<META HTTP-EQU
49 56 3D 22 45 78 70 69 72 65 73 22 20 63 6F 6E | IV="Expires" con
74 65 6E 74 3D 22 2D 31 22 3E 0D 0A 3C 4D 45 54 | tent="-1"> __<MET
41 20 48 54 54 50 2D 45 51 55 49 56 3D 22 52 65 | A HTTP-EQUIV="Re
66 72 65 73 68 22 20 63 6F 6E 74 65 6E 74 3D 22 | fresh" content="
30 3B 20 75 72 6C 3D 2F 77 5F 77 45 78 30 30 30 | 0; url=/w_wEx000
2E 61 70 77 22 3E 3C 2F 48 45 41 44 3E 0D 0A 3C | .apw"></HEAD>__<
2F 48 45 41 44 3E 3C 2F 48 54 4D 4C 3E | /HEAD></HTML>
-----
```

*/

Exemplo da função HTMLNOTAGS

No exemplo abaixo , a função HtmlNoTags é utilizada para permitir a utilização de caracteres especiais no conteúdo de um input para um formulário Html.

```
Local cHtml := ""
Local cInput := ""

// Conteudo do campo com tags HTML interpretables
cInput := "

// Ao montar o Input , aplicar a HtmlNoTags() ao conteudo do mesmo.

cHtml += '+HtmlNoTags(cInput)+'
```


Exemplo da função REDIRPAGE

Nos exemplos abaixo, a função redirpage é utilizada para gerar o script de redirecionamento em duas situações específicas.

```
/*
Em uma determinada função , caso um parâmetro não seja passado , o usuário deverá retornar a uma outra tela
*/

...
If empty(httpget->meuparam)
  // Parâmetro não informado , volta pro login
  cHtml := RedirPage('/W_Login.apw')
Else
  // Parametro Ok , executa o APH formteste
  cHtml := ExecInpage('FormTeste')
Endif
...

/* Ao chamar uma tela de download , mostrar uma mensahem e iniciar um download automaticamente */

...
cHtml += '

...mensagem de download...

'

// Devolve script de redirecionamento apontando para o arquivo
// com o target _blank , para ser aberto em uma nova janela.
cHtml += RedirPage('/downloads/arquivo.zip','_blank')
...
```

Exemplo da função RETSQLACE

No exemplo abaixo , utilizamos a função RetSqlAce para montar uma query de busca por título de uma determinada informação , considerando todas as possibilidades de acentuação , independentemente de como o banco foi alimentado e/ou a string de busca foi digitada.

IMPORTANTE : Na expressão da Query , o campo da tabela deve ser passado pela função LOWER do BAnco , pois a função retsqlace monta a string para busca com letras minúsculas.

```
cFind := 'acentuação'
cQuery := "SELECT * FROM " + RetSqlTab('ZZ1')
cQuery += "WHERE LOWER(ZZ1_TITULO) LIKE '%" + RetSqlAce(cFind) + "%' "
```

Exemplo da função VALTOSQL

```
cQuery := "SELECT * FROM FA2010 "
cQuery += "WHERE FA2_FILIAL = " + ValToSql('02')
cQuery += "AND FA2_DTINC <= " + ValToSql(date())
```

O exemplo acima, caso escrito de forma a realizar as conversões específicas para cada tipo de conteúdo seria o equivalente ao código abaixo:

```
cQuery := "SELECT * FROM FA2010 "
cQuery += "WHERE FA2_FILIAL = '02' "
cQuery += "AND FA2_DTINC <= " + DTOS( date() )
```

Exemplo da função VARINFO

No exemplo abaixo, é gerada uma string HTML com as informações do retorno da chamada de duas funções básicas da Linguagem Advpl.

```
User Function InfoTeste()
Local cHtml := "
cHtml += VarInfo('Date',date())
cHtml += VarInfo('Time',time())
Return cHtml

/* Deve ser gerado um echo no Console do Servidor parecido com este abaixo

Date -> D ( 10) [08/12/2003]

Time -> C ( 8) [20:17:48]
*/
```

Exemplo das funções de acentuação ApWebEX

No exemplo abaixo, vemos a diferença de comportamento entre as funções básicas da Linguagem Advpl Upper() e Lower(), e a função de Infra-estrutura Capital(), em relação às funções da Infra-estrutura APWEBEX UpperAce(), LowerAce() e CapitalAce(), quando utilizamos caracteres acentuados.

```
cRetorno := ""
cFrase := "não há EXPLICAÇÕES considerando excessões PARA O inexplicável."

cRetorno += "Original ..... " + cFrase + CRLF
cRetorno += "Upper() ..... " + upper(cFrase) + CRLF
cRetorno += "Lower() ..... " + lower(cFrase) + CRLF
cRetorno += "Capital() ..... " + capital(cFrase) + CRLF
cRetorno += "UPPERACE() ..... " + UPPERACE(cFrase) + CRLF
cRetorno += "LOWERACE() ..... " + LOWERACE(cFrase) + CRLF
cRetorno += "CAPITALACE() ..... " + CAPITALACE(cFrase) + CRLF
```

/*

Neste ponto , a variável cRetorno deverá conter :

Original não há EXPLICAÇÕES considerando excessões PARA O inexplicável.

Upper() Não Há EXPLICAÇÕES CONSIDERANDO EXCESSÕES PARA O INEXPLICÁVEL.

Lower() não há explicações considerando excessões para o inexplicável.

Capital() Não Há Explicações Considerando Excessões Para O Inexplicável.

UPPERACE() NÃO HÁ EXPLICAÇÕES CONSIDERANDO EXCESSÕES PARA O INEXPLICÁVEL.

LOWERACE() não há explicações considerando excessões para o inexplicável.

CAPITALACE() Não Há Explicações Considerando Excessões Para O Inexplicável.

*/

Exemplos das funções NTOC e CTON

No exemplo abaixo , utilizamos as funções cton e ntoc para realizar conversões de números em base decimal para outras bases numéricas e vice-versa.

```
nNum1 := CTON('01101001',2) // Converte binário para decimal
nNum2 := CTON('00DA25FE',16) // Converte Hexadecimal para decimal
```

```
nNum1++ // Soma 1 ao numero em nNum1
nNum2++ // Soma 1 ao numero em nNum2
```

```
cNum1 := NtoC(nNum1,2,8) // Converte para binário novamente
cNum2 := NtoC(nNum2,16,8) // Converte para Hexa novamente
```

```
/* -----
Ao final do programa , cNum1 será "01101010" e cNum2 será "00DA25FF"
----- */
```

APWEXADDERR

Sintaxe: APWEXADDERR ([cTitulo] , [cInfo]) --> .T.

Parâmetros

Argumento	Tipo	Descrição
cTitulo	Caracter	cTitulo corresponde a um título identificador da informação. Deve ter no máximo 20 caracteres. Os caracteres acima da vigésima posição serão ignorados.
cInfo	Caracter	cInfo corresponde à Informação a ser acrescentada ao ERROR.LOG em

caso de erro.

Retorno

Tipo	Descrição
Lógico	Esta função sempre retorna .T.

Descrição

Através da função ApWExAddErr(), podemos acrescentar uma string de informações adicionais em um buffer em memória, descarregado na geração do ERROR.LOG no caso de uma ocorrência de erro fatal na working thread atual.

Caso a função seja chamada sem nenhum parâmetro, a última ocorrência acrescentada pela função é eliminada da pilha interna de informações.

ATENÇÃO : Esta função deve ser apenas utilizada em casos de necessidade de obtenção de informações específicas acerca de uma ocorrência de erro não reproduzida em ambiente de testes e/ou não depurável, pois seu uso desnecessário prejudica a performance da aplicação final.

CAPITALACE

Sintaxe: CAPITALACE (< cString >) --> cStrCapital

Parâmetros

Argumento	Tipo	Descrição
cString	Caracter	String a ser convertida.

Retorno

Tipo	Descrição
Caracter	String resultante convertida para minúsculo , com as primeiras letras das palavras significantes em maiúsculo.

Descrição

Semelhante à função de Infra-estrutura Capital() , porém converte também caracteres acentuados.

A função CapitalAce() converte todos os caracteres de uma String para 'minúsculo' , e a primeira letra das palavras significantes para maiúsculo, semelhante à função Capital() , porém considera e converte também caracteres acentuados em OEM e/ou ANSI.

CTON

Sintaxe: CTON (< cString > , < nBase >) --> nNumero

Parâmetros

Argumento	Tipo	Descrição
cString	Caracter	cString corresponde à representação de um número em outra base numérica, compreendida entre 2 e 36.
nBase	Numérico	nBase corresponde à base numérica utilizada pelo numero representado em cString.

Retorno

Tipo	Descrição
Numérico	Número recebido como parâmetro em notação decimal (Base 10)

Descrição

Converte um número representado em String , de base 2 a 36 , para um número em base decimal (10).

Observação Importante :

São considerados caracteres válidos para compor um número de base 36 os 10 algarismos numéricos de 0 a 9 e os 26 caracteres alfabéticos maiúsculos compreendidos entre A e Z. Quaisquer caracteres presentes na String de parâmetro fora desta faixa de dados e/ou fora da base (por exemplo , uma conversão de string base 2 - binário - da string '01001020') retornará -1 (menos um).

ESCAPE

Sintaxe: ESCAPE (< cString >) --> cEscaped

Parâmetros

Argumento	Tipo	Descrição
cString	Caracter	cString é uma sequência de caracteres a ser processada.

Retorno

Tipo	Descrição
------	-----------

Caracter cEscaped corresponde à string original , com os caracteres reservados utilizando a notação %HH .

Descrição

A função `Escape()` deve ser utilizada para realizar conversões de caracteres especiais e reservados quando da necessidade de passagem de parâmetros via URL .

A sintaxe de uma requisição via URL para a passagem de parâmetros é:

(link)?param=conteudo¶m2=conteudo2&...

Quando passamos parâmetros via url , devemos tomar o cuidado de não utilizar caracteres reservados e especiais nos nomes e conteúdos de parâmetros. Para realizar estas conversões, utilizamos a função `Escape()`

EXECINPAGE

Sintaxe: `EXECINPAGE (< cAPHPage >) --> cHTMLPage`

Parâmetros

Argumento	Tipo	Descrição
cAPHPage	Caracter	Corresponde ao nome do APH que deve ser executado, sem a extensão.

Retorno

Tipo	Descrição
Caracter	cHTMLPage corresponde à string HTML gerada pela página APH / AHU.

Descrição

Através da função `ExecInPage()`, executamos uma página APH passada como parâmetro. A função deverá retornar a String HTML correspondente à página processada.

Porém a função `ExecInPage()` realiza tratamentos adicionais padrão relacionado ao comportamento dos Projetos WEB referentes à customizações, da seguinte maneira :

1. Primeiro é verificado se existe uma página AHU compilada com o nome cAPHPage. Caso exista , a mesma será executada pela `execinpage`.
2. Caso não exista AHU com este nome , é procurado pelo APH. Caso o APH exista , o mesmo será executado pela `ExecInPage`.

3. Caso não existam no RPO atual o APH nem o AHU com o nome especificado no parâmetro cAPHFile, o processamento é abortado com a ocorrência de erro [APWEXERR_0007] APH page [<cAPHPage>] not found in .RPO
4. Antes de executar o APH ou AHU identificado nos passos anteriores, é verificado se existe um ponto de entrada (User Function) compilado com o mesmo nome do APH. Caso exista , o ponto de entrada é executado . Ele deverá retornar uma String HTML. Se for retornada alguma string , a função retorna a string retornada e não processa o APH / AHU. Observação : A função ExecInPage não irá executar este tratamento da User Function com o nome de cAPHPage caso a função ExecInPage() esteja sendo executada através de uma USER FUNCTION .

Observação Importante - Envio parcial de HTML ao Browser.

A função ExecInpage(), juntamente com o APH compilado, ao serem processados irão tentar enviar o conteúdo HTML para o Browser solicitante durante o processamento, de modo que a função normalmente irá retornar uma String vazia.

Caso seja necessária a execução de uma página APH ou AHU e o não-envio da mesma para o Browse; por exemplo para a geração de um código HTML a ser enviado via e-mail; reavemos utilizar a função HttpSetPart(), realizando uma chamada da mesma antes da ExecInPage() , passando o parâmetro .F. para desabilitar temporariamente o envio de HTML simultâneo ao Browser, e após a execução da ExecInPage(), devemos re-habilitar o envio simultâneo através da chamada da função HttpSetPart() com o parâmetro .T.

EXISTPAGE

Sintaxe: EXISTPAGE (< cAphFile >) --> IFound

Parâmetros

Argumento	Tipo	Descrição
cAphFile	Caracter	Nome do arquivo APH, sem extensão e sem path , a ser verificado.

Retorno

Tipo	Descrição
Caracter	Retorna um valor booleano indicando se o arquivo APH especificado está compilado no RPO do ambiente atual.

Descrição

Utilizamos a função ExistPage() para identificarmos no ambiente atual se um determinado arquivo .APH encontra-se compilado atualmente no RPO em uso.

Exemplo :

```
If ExistPage('teste')
    conout('teste.aph compilado neste RPO')
```



```
Else
  conout('teste.aph NAO compilado neste RPO')
Endif
```

EXISTUSRPAGE

Sintaxe: EXISTUSRPAGE (< cAhuFile >) --> lExist

Parâmetros

Argumento	Tipo	Descrição
cAhuFile	Caracter	Nome do arquivo AHU, sem extensão e sem path , a ser verificado

Retorno

Tipo	Descrição
Lógico	Retorna um valor booleano indicando se o arquivo APH especificado está compilado no RPO do ambiente atual.

Descrição

Utilizamos a função ExistUSRPage() para identificarmos no ambiente atual se um determinado arquivo .AHU encontra-se compilado atualmente no RPO em uso.

Exemplo :

```
If ExistUSRPage('teste')
  conout('teste.ahu compilado neste RPO')
Else
  conout('teste.ahu NAO compilado neste RPO')
Endif
```

GETJOBPROFSTRING

Sintaxe: GETJOBPROFSTRING (< cKey > , < cDefault >) --> cKeyValue

Parâmetros

Argumento	Tipo	Descrição
cKey	Character	cKey corresponde à chave da seção de configuração da Working Thread atual a ser retornada.
cDefault	Character	Valor default (string) a ser retornado pela função caso a chave especificada não seja encontrada no .INI

Retorno

Tipo	Descrição
Character	Conteúdo da Chave solicitada . Caso a chave não seja encontrada , é retornado o conteúdo de cDefault . Caso esta função não seja executada a partir de uma Working Thread , ela retornará uma string em branco ("")

Descrição

Através desta função , podemos recuperar as configurações do Job da Working Thread atual.

GETWEXVERSION

Sintaxe: GETWEXVERSION () --> cBuildId

Retorno

Tipo	Descrição
Character	Corresponde à String Identificadora da versão , no formato <LIB> V.AAMMDHHmm . Vide Tabela A

Descrição

Esta função não requer argumentos , e retorna o Identificador do build / versão de Release das funções de Infra-Estrutura APWEBEX.

Observação : A data informada pela versão não corresponde à ultima compilação do RPO de um determinado Projeto WEB, mas sim à data de release da LIB de Infra-Estrutura APWEBEX.

Tabela A

Simbolo	Descrição
AA	Ano de geração da Lib
MM	Mês da geração da Lib
DD	Dia da geração da Lib
HH	Horário da geração da Lib

mm	Minutos do Horário de Geração da Lib
(HTTP)	Indica que a versão foi compilada com a configuração de envio progressivo de HTML simultâneo para o Browse . Esta opção é imprescindível para projetos que se utilizam desta LIB.

Por exemplo :

APWEBEX Version 3.0312021900 (HTTP)

HEXSTRDUMP

Sintaxe: HEXSTRDUMP (< cString > , [nStart] , [nLength]) --> cHExDump

Parâmetros

Argumento	Tipo	Descrição
cString	Caracter	String a ser lida para a geração do Dump
nStart	Numérico	Corresponde à posição inicial de cString a ser considerada para a geração do Dump, a partir da posição 1. Caso este parâmetro não seja informado , o Default é a posição 1 da string.
nLength	Numérico	nLength corresponde ao tamanho a ser considerado para a geração do DUMP a partir da posição inicial recebida em nStart. Caso este parâmetro não seja informado, é considerado o tamanho até o final da String.

Retorno

Tipo	Descrição
Caracter	O retorno da função corresponde à uma string Advpl , formatadas em 16 bytes em hexadecimal por linha , mais o separador pipe () , mais os 16 caracteres em Ansi. Os caracteres de controle (código ascii menor que 32) são convertidos para visualização para o caractere underline (_)

Descrição

Através da função HexStrDump(), podemos gerar uma string em Advpl em formato de Dump Hexadecimal a partir da string informada como parâmetro, a partir de uma determinada posição da string, considerando um número de bytes informado.

Caso os parâmetros nPosIni e nTamString não sejam informados, o dump gerado corresponde a string recebida como parâmetro em sua totalidade.

Observação :

Não devemos passar para a função HexStrDump uma string maior que 240 Kb , pois a geração da String de dump é realizada em memória, sendo a string final gerada em média 4,2 vezes maior que a string passada como parâmetro. Caso a string passada como parâmetro seja maior que 240 Kb , a execução será abortada com a ocorrência de erro fatal "string size overflow"

HTMLNOTAGS

Sintaxe

HTMLNOTAGS (< cStrHtml >) --> cStrNoTags

Parâmetros

Argumento	Tipo	Descrição
cStrHtml	Caracter	cStrHtml corresponde a uma String que não pode conter caracteres Html interpretáveis.

Retorno

Tipo	Descrição
Caracter	String original com os caracteres interpretáveis Html < > & " convertidos para caracteres não-interpretáveis.

Descrição

A Função HTMLNOTAGS converte as Tags interpretáveis de uma String HTML para TAGS não interpretáveis. Este recurso é normalmente utilizado quando precisamos montar um input Html com um conteúdo que não pode ser interpretado pelo browser como uma Tag . Esta função apenas converte os caracteres < (menor que) , > (maior que) , & (e comercial) e " (aspas duplas) .

Recomenda-se fortemente que, na montagem do value de um input html , o conteúdo do mesmo seja colocado entre aspas duplas, pois caso o conteúdo do value inicial do campo contenha aspas simples (não convertidas pela função HtmlNoTags) , isto poderá ocasionar perda de dados e erro de sintaxe no formulário Html.

HTTPISWEBEX

Sintaxe

HTTPISWEBEX () --> IIsApWEBEX

Retorno

Tipo	Descrição
Lógico	A função retornará .T. caso o ambiente de execução atual seja uma Working Thread WEBEX , inicializada pela função de infra-estrutura STARTWEBEX.

Descrição

Através da função HttpIsWebEx() é possível identificarmos se o programa atual está sendo executado através de uma Working Thread inicializada utilizando-se as funções de Infra-Estrutura APWEBEX

ISEMAIL

Sintaxe: ISEMAIL (< cEmail >) --> IEmailOk

Parâmetros

Argumento	Tipo	Descrição
cEmail	Caracter	cEmail corresponde a string a ser analisada , contendo um e apenas um endereço de e-mail.

Retorno

Tipo	Descrição
Caracter	Retorna .T. caso a string recebida como parâmetro atenda às definições de nomenclatura válidos para um endereço de e-mail.

Descrição

Utilizada para validar e-mails em Advpl , a função ISEMAIL recebe como parâmetro uma string contendo um e-mail , retornando .T. caso a string esteja em um formato válido respeitando a regra para nomenclatura de endereços de e-mail.

Regra : Um e-mail é considerado válido caso seja iniciado por um caracter , apenas contenha caracteres asc de a a z e 0 a 9 , e os caracteres @ (arroba) , . (ponto) , - (hífen) ou _ (underline) ; e deve conter uma e apenas uma arroba , e no minimo um ponto apos a arroba, intercalado por um caracter.

LOWERACE

Sintaxe

LOWERACE (< cString >) --> cStrLower

Parâmetros

Argumento	Tipo	Descrição
cString	Caracter	cString é a sequência de caracteres a ser convertida para letras minúsculas.

Retorno

Tipo	Descrição
Caracter	String original em letras minúsculas.

Descrição

A Função LOWERACE converte todos os caracteres de uma String para “minúsculo”, semelhante à função LOWER(), porém considera e converte também caracteres acentuados em ANSI.

NTOC

Sintaxe

NTOC (< nNumero > , < nBase > , < nTamStr >) --> cString

Parâmetros

Argumento	Tipo	Descrição
nNumero	Númerico	nNumero é o valor numérico , em base devimal , a ser convertido.
nBase	Númerico	nBase corresponde à base binária a ser utilizada para converter nNumero.
nTamStr	Númerico	nTamStr corresponde ao tamanho da string representando o numero na base desejada .

Retorno

Tipo	Descrição
Caracter	cString corresponde ao numero convertido para a base numérica especificada como parâmetro.

Descrição

A Função NTOC converte um número em notação decimal para um número representado por String utilizando uma base numérica entre 2 e 36 , preenchendo-o com “0” (zeros) à esquerda do tamanho especificado.

Observação : O Parâmetro nBase deve ser especificado com um número entre 2 e 36 . Caso seja passado como parâmetro um número base fora desta faixa, o processamento é abortado com a ocorrência de erro [APWEXERR_0022] INVALID NTOC BASE [X] , onde <X> foi a base passada como argumento.

REDIRPAGE

Sintaxe

REDIRPAGE (< cUrl > , [cTarget] , [nTime]) --> cScript

Parâmetros

Argumento	Tipo	Descrição
cUrl	Caracter	Link para onde o script deve apontar o redirecionamento
cTarget	Caracter	Destino do redirecionamento. Caso não especificado , o default é _self.
nTime	Numérico	Tempo (em segundos) de delay antes do redirecionamento ser executado.

Retorno

Tipo	Descrição
Caracter	Script Html / JavaScript que , ao ser executado no Browser (client) , chama a página/objeto chamado no Link.

Descrição

A função RedirPage é utilizada quando desejamos devolver ao Browser um script que , ao ser executado , redirecionará o Browser à abertura de um link passado como parâmetro.

RETSQLACE

Sintaxe

RETSQLACE (< cStrFind >) --> cStrQuery

Parâmetros

Argumento	Tipo	Descrição
cStrFind	Caracter	cStrFind corresponde à sequência de caracteres a ser procurada , podendo conter caracteres maiúsculos e minúsculos , com ou sem acentuação.

Retorno

Tipo	Descrição
Caracter	cStrQuery corresponde à string de busca a ser utilizada na query, utilizando caracteres em minúsculo.

Descrição

A função RetSqlAce é utilizada para auxiliar de montagem de queries de busca de caracteres acentuados em bases de dados. A função automaticamente trata a string original, removendo os acentos, convertendo todos os caracteres para minúsculas, e trocando todas as vogais e cedilhas da string original por uma sequência de caracteres acentuados em minúsculas para busca posicional.

RETSQCOND

Sintaxe: RETSQLCOND (< cAlias >) --> cSqlWhere

Parâmetros

Argumento	Tipo	Descrição
cAlias	Caracter	Lista contendo um ou mais aliases, separados por vírgula, a serem considerados para a montagem da expressão de validação.

Retorno

Tipo	Descrição
Caracter	Corresponde à expressão SQL para filtrar os dados através da cláusula WHERE

Descrição

Utilizamos a função RetSqlCond() como auxiliar na montagem de queries para busca de dados em tabelas em conformidade com o padrão adotado pelo ERP Microsiga e o Makira Hypersite, utilizando aliases de 3 caracteres.

A função retorna, a partir dos aliases passados como parâmetro, as expressões de filtro de dados para considerar a filial atual (xFilial) de acordo com o modo do arquivo (X2_MODO), e para sempre desconsiderar registros deletados.

Observações :

- Esta função foi mantida apenas por compatibilidade, pois a ordem de comparação de campos na cláusula WHERE de uma query deve procurar seguir a ordem dos campos dos indexadores do banco para efeitos de performance. Para

ganharmos performance nas Querys , devemos ao invés de utilizar a função RetSqlCond() , utilizar como primeira cláusula WHERE o retorno da função RetSqlFil() (comparação dos campos _FILIAL , os primeiros do(s) índice(s) do ERP) , que retorna apenas as comparações de Filial , e por último a função RetSqlDel, que retorna o script para verificação dos campos deletados (que é o último campo das chaves de índice do ERP , utilizando TopConnect).

- Devemos também atentar ao fato que a função RetSqlCond() retorna os campos para a comparação utilizando o prefixo da expressão SQL com o Alias reduzido (3 letras) das tabelas informadas, de modo que estes alias devem ser especificados na cláusula FROM , na abertura da Query, quando utilizamos a função RetSqlName para retornar o nome físico das Tabelas no Banco de Dados. A função RqtSqlTab() já retorna os nomes físicos das tabelas juntamente com os alias para este fim .

RETSQDEL

Sintaxe: RETSQLDEL (< cAliases >) --> cSqlWhere

Parâmetros

Argumento	Tipo	Descrição
cAliases	Caracter	Lista contendo um ou mais aliases , separados por vírgula , a serem considerados para a montagem da expressão de validação.

Retorno

Tipo	Descrição
Caracter	Corresponde à expressão SQL para filtrar os dados através da cláusula WHERE

Descrição

Utilizamos a função RetSqlDel() como auxiliar na montagem de querys para busca de dados em tabelas em conformidade com o padrão adotado pelo ERP Microsiga, utilizando aliases de 3 caracteres. A função retorna , a partir dos aliases passados como parâmetro , as expressões de filtro de dados para considerar o campo D_E_L_E_T_ da(s) tabela(s) passada(s) como parâmetro.

Observações :

- A ordem de comparação de campos na cláusula WHERE de uma query deve procurar seguir a ordem dos campos dos indexadores do banco para efeitos de performance . Para ganharmos performance nas Querys , devemos utilizar a função RetSqlDel() na montagem das ultimas consistências da cláusula WHERE de uma Query.
- Devemos também atentar ao fato que a função RetSqlDel() retorna os campos para a comparação utilizando o prefixo da expressão SQL com o Alias reduzido (3 letras) das tabelas informadas, de modo que estes alias devem ser especificados na cláusula FROM , na abertura da Query, quando utilizamos a função RetSqlName para retornar o nome físico das Tabelas no Banco de Dados. A função RqtSqlTab() já retorna os nomes físicos das tabelas juntamente com os alias para este fim .

RETSQFIL

Sintaxe

RETSQFIL (< cAliases > , [cCompFil]) --> cSQIWhere

Parâmetros

Argumento	Tipo	Descrição
cAliases	Caracter	Lista contendo um ou mais aliases , separados por vírgula , a serem considerados para a montagem da expressão de validação. Através de cCompFil é possível especificar uma Filial FIXA a ser comparada com os campos FILIAL do(s) alias(es) passados no parâmetro cAliases. Caso não informado, os campos _FILIAL da(s) tabela(s) passadas como parâmetro em cAliases serão comparados com o retorno da função xFilial() de cada alias, respectivamente.
cCompFil	Caracter	

Retorno

Tipo	Descrição
Caracter	Corresponde à expressão SQL para filtrar os dados através da cláusula WHERE

Descrição

Utilizamos a função RetSqlFil() como auxiliar na montagem de queries para busca de dados em tabelas em conformidade com o padrão adotado pelo ERP Microsiga, utilizando aliases de 3 caracteres.

A função retorna , a partir dos aliases passados como parâmetro , as expressões de filtro de dados para considerar o campo filial atual (xFilial) de acordo com o modo de abertura do arquivo no ERP (X2_MODO) .

Observação :

- Devemos atentar ao fato que a função RetSqlFil() retorna os campos para a comparação utilizando o prefixo da expressão SQL com o Alias reduzido (3 letras) das tabelas informadas, de modo que estes alias devem ser especificados na cláusula FROM , na abertura da Query, quando utilizamos a função RetSqlName para retornar o nome físico das Tabelas no Banco de Dados. A função RqtSqlTab() já retorna os nomes físicos das tabelas juntamente com os alias para este fim .

RETSQLTAB

Sintaxe

RETSQLTAB (< cAliasList >) --> cStrQuery

Parâmetros

Argumento	Tipo	Descrição
cAliasList	Caracter	String contendo um ou mais alias , separados por vírgula , a terem seus nomes físicos determinados.

Retorno

Tipo	Descrição
Caracter	String contendo nomes físicos e alias identificadores dos aliases recebidos como parâmetro.

Descrição

Utilizamos a função RetSqlTab() como auxiliar na montagem de query's quando trabalhamos com o padrão de Tabelas ERP Microsiga, que utilizam nomenclatura de alias com 3 Caracteres.

A função recebe como parâmetro um ou mais alias, separados por vírgula, de tabelas que desejam ser utilizadas na query, e retorna os nomes físicos das tabelas e seus respectivos alias para serem inseridos na query.

SEPARA

Sintaxe: SEPARA (< cString > , < cToken > , < IEmpty >) --> aTokens

Parâmetros

Argumento	Tipo	Descrição
cString	Caracter	String com a sequência de caracteres a ser "parseada"
cToken	Caracter	cToken corresponde à string a ser utilizada como separador para delimitar as informações.
IEmpty	Caracter	IEmpty identifica se caso um intervalo vazio entre tokens deve ser retornado como um elemento do array. Caso não especificado , o Default é .T.

Retorno

Tipo	Descrição
Caracter	Array de uma dimensão contendo os elementos parseados pela rotina levando-se em conta o separador passado como parametro.

Descrição

Através da função SEPARA(), pode-se parsear uma string de elementos a partir de um determinado separador, sendo retornado um Array com os elementos identificados na String.

Exemplo :

alInfo	:=	Separa('1,2,,4',',',f.)	//	Resulta	{'1','2','4'}
alInfo := Separa('1,2,,4',',',t.) // Resulta {'1','2','4'}					

Observação :

Para realizar a análise de uma string, cujo delimitador tenha apenas 1 byte, e as ocorrências de dois separadores juntos sejam ignoradas na geração do array, a função separa() utiliza a função StrTokArr(), função escrita em C no Protheus Server, mais rápida para este processamento. Apenas existe a necessidade de utilizarmos a função Separa() caso as ocorrências de dois separadores juntas devam ser consideradas no array de resultado e/ou a string utilizada como separador possua mais que 1 byte de tamanho.

UNESCAPE

Sintaxe

UNESCAPE (< cString >) --> cUnEscaped

Parâmetros

Argumento	Tipo	Descrição
cString	Caracter	cString é a string a ter os caracteres em formato escape convertidos.

Retorno

Tipo	Descrição
Caracter	cUnEscaped corresponde à string recebida como parâmetro, com os caracteres originalmente em notação escape (%HH) convertidos para ASCII

Descrição

Realiza a operação inversa à função Escape(), convertendo os caracteres especiais em notação %HH em caracteres ASCII.

Observação : Apenas serão convertidos os caracteres originalmente tratados pela função Escape()

UPPERACE

Sintaxe

UPPERACE (< cString >) --> cStrUpper

Parâmetros

Argumento	Tipo	Descrição
cString	Caracter	String a ser convertida. Pode conter também acentos

Retorno

Tipo	Descrição
Caracter	Retorna a string original com todas as letras maiúsculas , inclusive as letras acentuadas.

Descrição

A Função UPPERACE converte todos os caracteres de uma String para “maiúsculo” , semelhante à função UPPER() , porém considera e converte também caracteres acentuados em OEM e ANSI

UPSTRTRAN

Sintaxe: UPSTRTRAN (< cString > , < cSearch > , [cReplace] , [nStart] , [nCount]) --> cNewString

Parâmetros

Argumento	Tipo	Descrição
cString	Caracter	cString corresponde à sequência de caracteres ou campo memo a ser pesquisado.
cSearch	Caracter	Corresponde à sequência de caracteres a ser procurada em cString
cReplace	Caracter	cReplace corresponde à sequência de caracteres que deve substituir a string cSearch. Caso não seja especificado, as ocorrências de cSearch em cString serão substituídas por uma string nula ("")
nStart	Numérico	nStart corresponde ao número sequencial da primeira ocorrência de cSearch em cString a ser substituída por cReplace. Se este argumento for omitido , o default é 1 (um) . Caso seja passado um numero menor que 1, a função retornará uma string em branco ("")
nCount	Numérico	nCount corresponde ao número máximo de trocas que deverá ser realizada pela função . Caso este argumento não seja especificado , o default é substituir todas as coorências encontradas.

Retorno

Tipo	Descrição
Caracter	A função UPSTRTRAN retorna uma nova string, com as ocorrências especificadas de cSearch trocadas para cReplace, conforme parametrização.

Descrição

Similar à função Strtran(), porém realiza a busca da ocorrência da string considerando letras maiúsculas e minúsculas. A função Strtran() é case-sensitive, e a função UpStrtran() não.

VALTOSQL

Sintaxe: VALTOSQL (< xExpressao >) --> cQryExpr

Parâmetros

Argumento	Tipo	Descrição
xExpressao	(Qualquer)	Valor Advpl a ser convertido para utilização em Query. Pode ser dos tipos "C" Caracter , "N" Numérico e "D" Data.

Retorno

Tipo	Descrição
Caracter	Expressão a ser acrescentada na Query.

Descrição

A Função VALTOSQL() é utilizada como auxiliar na montagem de Query's , convertendo um conteúdo variável Advpl para a string correspondente a ser acrescentada na Query.

Podemos passar como parâmetro uma Expressão do tipo "C" Caracter , "D" Data ou "N" Numérica.

- A expressão Catacter será colocada entre aspas simples, sendo removidas as aspas simples contidas na mesma , caso existam .
- Uma expressão numérica será simplesmente convertida para caracter , com aproximação de 2 casas decimais.
- Uma expressão Data será convertida para formato ANSI (AAAAMMDD) , entre aspas simples.

VARINFO

Sintaxe: VARINFO (< cld > , < xVar > , [nMargem] , [lHtml] , [lEcho]) --> cVarInfo

Parâmetros

Argumento	Tipo	Descrição
cld	Caracter	cld corresponde a um nome atribuído à variável para análise. Internamente , apenas é utilizado para prefixar o retorno das informações da VarInfo.
xVar	(Qualquer)	Variável de qualquer tipo a ser examinada
nMargem	Numérico	Corresponde à margem esquerda inicial de espaços da String de retorno , multiplicado por 5. Default = 0
lHtml	Lógico	Identifica se a String de retorno será montada em formato Html (.T. / Default) ou ASCII (.F.)
lEcho	Lógico	Define se o Echo do retorno deve ser enviado ao console do Protheus Server , caso habilitado. (Default = .T.)

Retorno

Tipo	Descrição
Caracter	String contendo o "Dump" da análise da variável. Caso lHtml seja .T. , retorna String em formato HTML , , senão retorna string ASCII com quebras CRLF.

Descrição

A Função VARINFO() gera um texto ASCII e/ou Html , com possibilidade de ECHO para o Console do Protheus Server (caso habilitado) , com as informações sobre o conteúdo de uma variável de memória Advpl , de qualquer tipo .

Cada tipo de variável possui um tratamento para conversão em String :

- CodeBlock : É exibido apenas o tipo da mesma (B)
- Array : Todos os níveis e elementos do mesmo são explorados recursivamente.
- Objeto : No caso de um Objeto de XML e/ou Web Services, são exploradas todas as suas propriedades recursivamente.
- (demais tipos) : São convertidos para String através da função AllToChar

Observação : O segundo parâmetro (xVar) deve ser uma variável Advpl que deve existir no escopo de variáveis. Caso a variável não exista, o processamento é abortado com a ocorrência de erro "Variablçe does not exist" . Para saber se uma determinada variável existe no escopo da execução da função atual, deve ser utilizada a função Advpl TYPE(), onde passamos a variável a ter seu tipo determinado como string (entre aspas) .

WEBINFO

Sintaxe: WEBINFO () --> cHtmlInfo

Retorno

Tipo	Descrição
Caracter	cHTMLInfo corresponde à string HTML contendo as informações da requisição HTTP.

Descrição

A função WebInfo() foi desenvolvida para ser chamada através de uma requisição http , via link .apl ou .apw , e ela identifica todos os parâmetros recebidos via uma requisição http: Parâmetros via get , post , o header HTTP, os Cookies, o content-type , Legth , Content-disposition , SoapRaction (ação SOAP para requisições de WebServices) , e OtherContent (caso o conteúdo postado não seja um text/html)

Esta função retorna uma página Html com todas estas informações, e é utilizada no desenvolvimento de projetos. quando temos a necessidade prática de recuperarmos todas as informações provenientes de uma requisição HTTP. Adicionalmente , a função WebInfo

Por exemplo, com o Protheus configurado para atender requisições de links .apl via HTTP , chame a função WebInfo.apl através do link :

<http://localhost/webinfo.apl?param1=teste¶m2=outroteste>

Será exibido no Web Browser uma tela semelhante à tela abaixo :

```
__aCookies -> ARRAY ( 0 ) [...]
__aPostParms -> ARRAY ( 0 ) [...]
__nProclId -> N ( 10 ) [1169539314]
__aProcParms -> ARRAY ( 0 ) [...]
__httpPage -> C ( 0 ) []

__HttpHeader      ->      ARRAY      (      8      )      [...]
  __HttpHeader[1]  ->      C      (      25      )      [GET      /webinfo.apl      HTTP/1.1]
  __HttpHeader[2]  -> C ( 172 ) [Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*]
  __HttpHeader[3]  ->      C      (      22      )      [Accept-Language:      pt-br]
  __HttpHeader[4]  ->      C      (      30      )      [Accept-Encoding:      gzip,      deflate]
  __HttpHeader[5]  -> C ( 61 ) [If-Modified-Since: Wed, 10 Dec 2003 12:24:29 GMT; length=1003]
  __HttpHeader[6]  -> C ( 81 ) [User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)]
  __HttpHeader[7]  ->      C      (      13      )      [Host:      automan]
  __HttpHeader[8]  -> C ( 22 ) [Connection: Keep-Alive]

HttpRCtType() -> C ( 0 ) []
HttpRCtLen() -> N ( 10 ) [ -1]
HttpRCtDisp() -> C ( 0 ) []
```

```
SoapRAction() -> C ( 0 ) []
```

```
HttpOtherContent() -> C ( 0 ) []
```

Caso a mesma requisição seja realizada através de link .apw , utilizando-se a tecnologia WEBEX , deverá ser exibida uma tela semelhante à tela abaixo :

```
aHeaders          ->          ARRAY          (          9)          [...]
    aHeaders[1]    ->          C          (          25)          [GET          /webinfo.apw          HTTP/1.1]
    aHeaders[2] -> C ( 172) [Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-
flash,          application/vnd.ms-excel,          application/vnd.ms-powerpoint,          application/msword,          */*]
    aHeaders[3]    ->          C          (          22)          [Accept-Language:          pt-br]
    aHeaders[4]    ->          C          (          30)          [Accept-Encoding:          gzip,          deflate]
    aHeaders[5] -> C ( 61) [If-Modified-Since: Tue, 09 Dec 2003 21:23:03 GMT; length=1480]
    aHeaders[6] -> C ( 81) [User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)]
    aHeaders[7]    ->          C          (          21)          [Host:          apwebex.automan]
    aHeaders[8]    ->          C          (          22)          [Connection:          Keep-Alive]
    aHeaders[9]    ->          C          (          41)          [Cookie:          SESSIONID=1071153371;          AP5PROCID=0]
httpCookies        ->          ARRAY          (          2)          [...]
    httpCookies[1] ->          C          (          9)          [SESSIONID]
    httpCookies[2] -> C ( 9) [AP5PROCID]
```

```
SESSIONID -> C ( 10) [1071153371]
```

```
AP5PROCID -> C ( 1) [0]
```

```
httpPost -> ARRAY ( 0) [...]
```

```
httpGet -> ARRAY ( 0) [...]
```

```
HttpRCtType() -> C ( 0) []
```

```
HttpRCtLen() -> N ( 10) [ -1]
```

```
HttpRCtDisp() -> C ( 0) []
```

```
SoapRAction() -> C ( 0) []
```

```
HttpOtherContent() -> C ( 0) []
```

STARTWEBEX

Sintaxe

STARTWEBEX ([NIL]) --> ISucess

Parâmetros

Argumento	Tipo	Descrição
NIL	(NULO)	Este ponto de entrada não recebe parâmetros.

Retorno

Tipo	Descrição
Lógico	ISucess corresponde ao status de montagem de ambiente. Caso o ambiente tenha sido montado com sucesso, o ponto de entrada deve retornar .T. , caso contrário .F. . Uma vez retornado .F. , o Protheus irá eliminar esta Working Thread da memória.

Descrição

Este ponto de entrada é executado na inicialização de cada Working Thread, quando utilizada a configuração para a Lib APWEBEX.

Através dele, devemos iniciar o ambiente necessário ao atendimento das requisições de processamento via Browser , através de links .apw, tais como a abertura de dicionários e conexão com o Banco de Dados.

Grupos Relacionados

Principal / A Tecnologia Protheus / Programação Advpl para WEB / Infraestrutura APWEBEX / Pontos de Entrada

CONNECTWEBEX

Sintaxe

CONNECTWEBEX (< cFnLink >) --> cHtmlVId

Parâmetros

Argumento	Tipo	Descrição
cFnLink	Caráter	Função chamada através do Link . Por exemplo , um link no browser chamando <code>http://localhost/w_teste.apw?Opc=03</code> , seria recebido neste parâmetro a string "W_TESTE"

Retorno

Tipo	Descrição
Caracter	Caso retornada uma string em branco , a execução da função originalmente chamada no link .apw prossegue normalmente. Caso contrário , a string retornada é devolvida ao Browser solicitante , e a função chamada através do link não é executada.

Descrição

Este ponto de entrada é executado imediatamente antes do processamento de uma requisição realizada através de um browser para processamento de uma função Advpl , através de um link .apw , permitindo realizar uma pré-validação antes de cada processamento solicitado através do Browser.

RESETWEBEX

Sintaxe

RESETWEBEX (< cFnLick >) --> cHtmlAdd

Parâmetros

Argumento	Tipo	Descrição
cFnLick	Caracter	cFnLink corresponde à função Advpl que foi chamada e processada imediatamente antes da chamada deste ponto de entrada.

Retorno

Tipo	Descrição
Caracter	Este ponto de entrada DEVE retornar uma string , podendo ser inclusive uma string vazia. A String retornada será acrescentada ao Código HTML a ser retornado ao Browser

Descrição

Este ponto de entrada é executado imediatamente após o processamento de uma requisição de processamento Advpl através de uma Web Browser utilizando as configurações e Lib APWEBEX.

Ele permite que seja executado um processamento adicional após o processamento de cada requisição .apw , e ainda permite um retorno de HTML adicional ao browser.

Vale a pena lembrar que este ponto não será executado em caso de erro fatal no ponto de entrada U_CONNECTWEBEX ou na execução da função principal chamada através do Link.

FINISHWEBEX

Sintaxe

FINISHWEBEX () --> NIL

Parâmetros

Argumento	Tipo	Descrição
Retorno		

Tipo	Descrição
(NULO)	O Retorno deste ponto de entrada não é utilizado.

Descrição

Este ponto de entrada é executado quando da finalização (Fechamento) de uma Working Thread APWEBEX. Não recebe parâmetros , e não requer retorno. Ele permite que seja executado um procedimento qualquer no momento da saída de uma Working Thread, seja por timeout ou por tempo total de permanência no ar.

ENDSESSION

Sintaxe

ENDSESSION (< cSessionId >) --> NIL

Parâmetros

Argumento	Tipo	Descrição
cSessionId	Caracter	cSessionId corresponde à string identificadora das sessions deste usuário.

Retorno

Tipo	Descrição
(NULO)	O retorno deste ponto de entrada deve ser nulo

Descrição

Através deste ponto de entrada , podemos executar uma rotina Advpl quando da finalização das sessions de um usuário por timeout de inatividade. O Retorno deste ponto de entrada não é utilizado, devendo ser nulo (NIL).

Apenas devemos compilar este ponto de entrada no Projeto caso realmente exista a necessidade de ser executado um processamento específico relacionado à finalização das sessions de um usuário. Vale a pena ressaltar também que este ponto de entrada apenas é chamado na finalização das sessions por tempo de inatividade. Caso seja utilizada a função `httpreesession()` para limpar da memória as sessions do usuário atual em uma Working Thread, este ponto de entrada não será chamado.

WEBEXERROR

Sintaxe

WEBEXERROR (< oErrorObj > , < cErrorLog > , < cErrorHtml >) --> cMsgHtml

Parâmetros

Argumento	Tipo	Descrição
oErrorObj	Objeto	Objeto do Erro Advpl.
cErrorLog	Caracter	Mensagem ASCII que será gravada no arquivo error.log
cErrorHtml	Caracter	Mensagem HTML original da rotina de tratamento de Erro

Retorno

Tipo	Descrição
Caracter	Retorno opcional. Caso retornado NIL ou string vazia , será retornado ao usuário o HTML de erro gerado pela rotina de tratamento standard. Caso o ponto de entrada retorne uma String HTML , ela será mostrada ao usuário no lugar do HTML gerado pela rotina de tratamento de erro.

Descrição

Este ponto de entrada será chamado no caso de uma ocorrência de erro fatal Advpl durante a execução de uma Working Thread em ambiente / Lib APWEBEX, permitindo a montagem de uma mensagem de erro HTML customizada a ser devolvida ao usuário.

Este ponto de entrada recebe como parâmetros o objeto do erro , a descrição ASCII completa do erro gravada no error.log , e o HTML default montado pela da rotina de tratamento de erro que será devolvido ao usuário. Através da utilização deste ponto de entrada , é possível gerar um HTML diferenciado conforme a necessidade, para mostrar a ocorrência de erro e/ou maiores instruções ao usuário.

OBSERVAÇÕES

- Independentemente do retorno deste ponto de entrada , a Working Thread que apresentou ocorrência de erro será derrubada após o retorno do HTML para o Browser , e o arquivo error.log será gerado normalmente . Caso este ponto de entrada retorne uma string em branco , será mostrada ao usuário a mensagem de erro HTML default gerada pela rotina de tratamento de erro.
- Este ponto de entrada será chamado apenas caso a ocorrência de erro esteja relacionada com uma chamada de função via link .apw, aplicando-se apenas à função .apw chamada e aos pontos de entrada U_CONNECTWEBEX e U_RESETWEBEX. Em caso de ocorrências de erro no start da Thread (U_STARTWEBEX), na finalização da Thread (U_FINISHWEBEX) e na finalização de sessions de usuário por timeout (U_ENDSESSION), o ponto de entrada U_WEBEXERROR não será chamado .

Recomenda-se fortemente que , na montagem da função deste ponto de entrada , não seja utilizado nenhum recurso Advpl que dependa de ambiente , disco , base de dados ou Session , limitando-se apenas à customizar uma mensagem de ocorrência de erro ao usuário .

Caso seja reproduzida alguma ocorrência de erro neste ponto de entrada, isto fará a aplicação (Protheus Server) enviar ao Browser um HTML gerado pela rotina de tratamento de erro default do Protheus.

