

# **ADVPL III – Rotinas Web**

**Julho/2010**

## Conteúdo

Capítulo 01 – Introdução .....	3
Objetivos do Treinamento .....	3
Metas do Treinamento .....	3
Conteúdo.....	3
Capítulo 02 – Manipulação de E-mail .....	4
Parâmetros de Email .....	4
Passos para envio de email.....	4
Exercícios email.....	5
Capítulo 03 – Manipulação de FTP .....	7
Funções de FTP.....	7
Capítulo 04 – Workflow.....	10
Configuração.....	10
Criação do HTML .....	14
Criação de Processos em ADVPL.....	15
Classe TWFPPROCESS .....	15
Capítulo 05 – WEB Service .....	20
Configuração.....	20
Exercício .....	21
Capítulo 06 – ADVPL ASP .....	22
Características do ADVPL ASP – Arquivos .APH.....	22
Desenvolvimento de Funções .APL.....	24

## Capítulo 01 – Introdução

- Objetivos do treinamento;
- Metas do treinamento;
- Conteúdo;

### Objetivos do Treinamento

Conhecer e empregar adequada e eficazmente os conceitos e funcionalidades do Protheus – ROTINAS DA WEB.

### Metas do Treinamento

Habilitar o aluno do curso de ADVPL III – Rotinas da WEB, a obter o conhecimento sobre a linguagem AdvPL – Advanced Protheus Language, voltada para a WEB e utilizada na programação do Protheus da Microsiga, um dos softwares de gestão mais utilizados em toda a América Latina. O conhecimento do uso desta linguagem permite a elaboração de customizações e o desenvolvimento de novas soluções.

### Conteúdo

#### Email

- Função para conectar-se ao servidor
- Função para envio de mensagem
- Função para desconectar-se do servidor

#### • FTP

- Manipulação FTP
- Download
- Upload

#### • Workflow

- Configuração dos parâmetros
- Configuração do Envio e Retorno
- Configuração do JOB no Servidor
- Configuração do JOB no Servidor
- Código Fonte para iniciar o JOB
- Schedule de Relatórios
- Schedule de Rotinas

#### • ADVPLASP

- Monitorando páginas simples
- Trabalhando entre sessões
- Usando cookies
- Usando Método Post e Get
- Montando páginas dinâmicas
- ADVPL ASP com Base de Dados

## Capítulo 02 – Manipulação de E-mail

A linguagem ADVPL permite o envio e o recebimento de mensagens através de e-mails. Veremos a seguir as funções para manipulação de e-mails e parâmetros configurados previamente que podemos também utilização no envio de emails.

### Parâmetros de Email

Parâmetro	Descrição	conteúdo
MV_RELACNT	Conta a ser utilizada no envio de E-Mail para os relatórios	workflow@advpl.com.br
MV_RELAPSW	Senha para autenticação no servidor de e-mail	escola1010
MV_RELAUTH	Determina se o Servidor necessita de Autenticação.	.T.
MV_RELFROM	E-mail utilizado no campo FROM no envio de relatórios por e-mail	workflow@advpl.com.br
MV_RELPSW	Senha da Conta de E-Mail para envio de relatórios	escola1010
MV_RELSERV	Nome do Servidor de Envio de E-mail utilizado nos relatórios	smtp.advpl.com.br
MV_WFSMTP	Nome do Servidor SMTP(envio de mensagem	smtp.advpl.com.br
MV_WFACC	Conta de Acesso(ex:informática)	workflow@advpl.com.br
MV_WFPASSW	Senha da Conta do Workflow	escola1010

Exemplo de utilização:

```
Private cSrv := GetMV("MV_RELSERV")  
Private cMail := GetMV("MV_RELACNT")  
Private cPass := GetMV("MV_RELPSW")  
Private lAuth := GetMv("MV_RELAUTH")  
Private cMailA := GetMV("MV_RELACNT")  
Private cPswA := GetMV("MV_RELAPSW")
```

Ou

```
Private cSrv := GetMV("MV_WFSMTP")  
Private cMail := GetMV("MV_WFACC")  
Private cPass := GetMV("MV_WFPASSW")  
Private lAuth := GetMv("MV_RELAUTH")  
Private cMailA := GetMV("MV_RELACNT")  
Private cPswA := GetMV("MV_RELAPSW")
```

### Passos para envio de email

- Conectar ao Servidor SMTP  
CONNECT SMTP SERVER cSrv ACCOUNT cMail PASSWORD cPass RESULT IResul
- Verificar se necessita de autenticação

IOk := MailAuth(cEmail,cPass)

- Verificar mensagens da conexão  
GET MAIL ERROR cError
- Enviar o email  
SEND MAIL FROM **cDe** TO **cPara** SUBJECT **cAssunto** BODY **cMsg** CC **cCC** CCO **cCCO**;  
ATTACHMENT **cAnexo** RESULT **ISend**
- Desconectar do Servidor  
DISCONNECT SMTP SERVER

Exemplo:

```
cServer := GETMV("MV_WFSMTP")
cAccount := GETMV("MV_WFACC")
cPassword := GETMV("MV_WFPASSW")
cFrom := GETMV("MV_RELFROM")
cDest := "cris@advpl.com.br"
```

```
CONNECT SMTP SERVER cServer ACCOUNT cAccount PASSWORD cPassword Result IConectou
```

```
IAuth := GetMv("MV_RELAUTH")
```

```
If IAuth
```

```
IOk := MailAuth( cAccount ,GetMV("MV_RELAPSW"))
```

```
If !IOk
```

```
IOk := QAGetMail()
```

```
EndIf
```

```
EndIf
```

```
cAssunto := "Teste de corpo de email"
```

```
SEND MAIL FROM cFrom TO cDest SUBJECT "Teste Assunto";  
BODY cAssunto ATTACHMENT "\\system\error.log" RESULT IEnviado
```

```
If !Enviado
```

```
GET MAIL ERROR _cMensagem
```

```
Alert(_cMensagem)
```

```
Endif
```

```
DISCONNECT SMTP SERVER Result IDisConectou
```

## Exercícios email

1. Fazer um programa que envie para você mesmo um email para cada cliente com as seguintes informações:

**Codigo:** 999999

**Nome:** XXXXXXXXXXXXXXXX

**Endereço:** XXXXXXXXXXXXXXXX

**Telefone:** 99 9999 9999

No Assunto coloque o Nome reduzido do cliente

2. Fazer um programa que envie para você mesmo um único email com todos os fornecedores com as seguintes informações:

<b>Codigo</b>	<b>Loja</b>	<b>Nome</b>	<b>Endereço</b>
999999	99	XXXXXXXXXX	XXXXXXXXXXXXXXXXXX
999999	99	XXXXXXXXXX	XXXXXXXXXXXXXXXXXX
999999	99	XXXXXXXXXX	XXXXXXXXXXXXXXXXXX

No Assunto coloque o Nome reduzido do fornecedor

## Capítulo 03 – Manipulação de FTP

A linguagem ADVPL permite o download e upload de arquivos através de FTP e também manipulação de arquivos no FTP, como alteração, exclusão, renomear arquivos e trocar de diretórios dentro do FTP.

### Funções de FTP

- **FTPConnect** - Conecta a um FTP

Sintaxe:

lRet := FTPConnect ( cServer, nPorta, cUser, cPass )

Argumento	Obrig	Tipo	Descrição
cServer	Sim	C	Servidor de FTP
nPorta	Sim	N	Porta de FTP, Default 21
cUser	Sim	C	Nome de usuário de FTP
cPass	Sim	C	Senha de FTP
lRet		L	.T. ou .F.

**Exemplo:**

lRet := FTPConnect("ftp.advpl.com.br", 21, "aluno", "123456")

- **FTPDirectory** - Retorna conteúdo do diretório corrente do FTP

Sintaxe:

aRet := FTPDirectory ( cMascara , cAttr )

Argumento	Obrig	Tipo	Descrição
cMascara	Sim	C	Máscara dos arquivos a retornar, ex. "*.*)"
cAttr	Não	C	Atributo dos arquivos a retornar
aRet		A	array com arquivos que obedecem os argumentos

**Exemplo:**

aRet := FTPDirectory("\*.\*)", "R")

- **FTPUpload** - Sobe arquivos para o FTP

Sintaxe:

lRet := FTPUpload( cFileLocal, cFileFTP )

Argumento	Obrig	Tipo	Descrição
cFileLocal	Sim	C	Caminho completo no server, o caminho deve estar dentro do rootpath.
cFileFTP	Sim	C	Nome do arquivo no FTP
lRet		A	T. ou .F.

**Exemplo:**

lRet := FTPUpload("\\system\\error.log", "error.log" )

- **FTPDownload** - Baixa arquivos do FTP

Sintaxe:

lRet := FTPDownload ( cFileLocal, cFileFTP )

Argumento	Obrig	Tipo	Descrição
cFileLocal	Sim	C	Caminho completo no server, o caminho deve estar dentro do rootpath.
cFileFTP	Sim	C	Nome do arquivo no FTP
lRet		A	T. ou .F.

**Exemplo:**

lRet := FTPDownload("\system\error.log", "error.log" )

- **FTPErase** - Apaga arquivo no FTP no diretório corrente

Sintaxe:

lRet := FTPErase( cFileFTP )

Argumento	Obrig	Tipo	Descrição
cFileFTP	Sim	C	Nome do arquivo no FTP que se quer apagar
lRet		A	T. ou .F.

**Exemplo:**

lRet := FTPErase("error.log")

- **FTPRenameFile** - Renomeia arquivo no FTP

Sintaxe:

lRet := FTPRenameFile(cFileOri, cFileNew)

Argumento	Obrig	Tipo	Descrição
cFileOri	Sim	C	Nome do Arquivo original
cFileNew	Sim	C	Nome novo para o arquivo
lRet		A	T. ou .F.

**Exemplo:**

lRet := FTPRenameFile("sigafin.xnu", "sigafin.old")

- **FTPDirChange** - Muda diretório no FTP

Sintaxe:

lRet := FTPDirChange( cDirFTP )

Argumento	Obrig	Tipo	Descrição
cDirFTP	Sim	C	Caminho escolhido
lRet		A	T. ou .F.

**Exemplo:**

lRet := : FTPDirChange("\tmp")

- **FTPGetCurDir** - Retorna o nome do diretório corrente do FTP

Sintaxe:

cRet := FTPGetCurDir( )

- **FTPDisconnect** - Desconecta o FTP, retorna .T. ou .F.



Sintaxe:

lRet := FTPDisconnect( )

### **Exemplo de um programa com FTP**

```
# include 'protheus.ch'
User Function SendFtp()
Local cServer := 'ftp.totvstec.com.br'
Local nPorta := 21
Local cUser := 'aluno01'
Local cPass := 'escola1010'
Local n:= 0
Local aRetDir := {}
cpyt2s('c:\login.htm', '\system\')

If FTPConnect( cServer, nPorta, cUser, cPass )
  Conout('FTP conectado!')
  Conout('Diretorio FTP',FTPGetCurDir())
  Conout('FTPDireChange',FTPDireChange('alunos'))
  Conout('FTPDireChange',FTPDireChange('aluno01'))
  Conout('Diretorio FTP',FTPGetCurDir())
  FTPUpload( "\system\error.log", "error.log")
  If FTPRenameFile("error.log", "teste.old")
    Conout('Arquivo Renomeado! xls para old')
  Else
    Conout('Falha Renomear!')
  EndIf
  If FTPErase("teste.old")
    Conout('Exclusão Ok! ' + "teste.old")
  Else
    Conout('Falha Exclusão!' + "teste.old")
  EndIf

  aDirFTP := FTPDIRECTORY( "", )
  aDirLoc := DIRECTORY( "\system\*.xnu", )

  For I:=1 to Len(aDirLoc)
    If FTPUpload( "\system\" + aDirLoc[I][1], aDirLoc[I][1] )
      Conout('UpLoad Ok! ' + aDirLoc[I][1])
    Else
      Conout('Falha UpLoad!' + aDirLoc[I][1])
    EndIf
  Next
  For I:= 1 to LEn(aDirFTP)
    If FTPDownload( '\system\' + aDirFTP[I][1], aDirFTP[I][1] )
      Conout(funname()+' - Download Ok! ' + aDirFTP[I][1])
    Else
      Conout('Falha DownLoad ' + aDirFTP[I][1])
    EndIf
  Next
  FTPDisconnect()
Else
  Conout('Falha Conexao!')
EndIf

Return .T.
```

## Capítulo 04 – Workflow

É a automação de processos de negócio, utilizando-se o email como forma de comunicação nos processos críticos da organização, evitando gargalos e deixando o sistema mais ágil e seguro. Essa comunicação é feita a partir de um processo ativado por alguma rotina processada pelo usuário.

### Benefícios:

- Aumento da eficiência eliminando passos desnecessários.
- Melhor gerenciamento dos processos.
- Melhora nos serviços ao cliente.
- Flexibilidade.
- 1. Forte rastreabilidade dos processos: pode-se saber a atual situação de qualquer processo através da consulta da rastreabilidade.
- 2. Sistema Protheus e torna ativas, e a informação e a ação necessária a ser tomada é levada ao usuário e não o contrário.
- 3. Controle de *time-out* dos processos.
- 4. Qualquer processo existente pode ser automatizado através da construção de um Workflow.

As ações a serem tomadas pelas pessoas em cobrança podem ser realizadas através de quatro meios, evitando que um processo tenha seu ciclo interrompido por falta de comunicação, são eles:

- e-mail: os processos são respondidos através de um cliente de e-mail homologado.
- *browser* de internet: através do Internet Explorer.
- Protheus: através do recurso de *Messenger* do Protheus.
- Pontos de entrada: através de pontos de entradas existentes no sistema.

## Configuração

A configuração do Workflow é feita no módulo configurador (Ambiente - Workflow – Contas de Email), e em (Ambiente - Workflow - Parâmetros WF)

Os parâmetros são atualizados no cadastro de parâmetros (SX6)

Correio			
Opção	Item	Parâmetro	Descrição
Caixa de correio	Conta	MV_WFMLBO X	Caixa de correio a ser utilizada pelo workflow para o envio e recebimento de mensagens.
	Envia figura do html como anexo da mensagem	MV_WFIMAG E	Recurso ainda não disponível.
Composição da mensagem	Envia html no corpo da mensagem	MV_WFHTML	Selecionando esta opção, o html passara a fazer parte do corpo da mensagem. O contrário, irá como anexo.
	Envio automático	MV_WFSNDA U	Após a criação do processo, o workflow enviará a mensagem imediatamente.

	Usa Java Script	MV_WFJAVAS	No html, o Workflow adicionará rotinas semi-prontas em Java script.
--	-----------------	------------	---

Processos			
Opção	Item	Parâmetro	Descrição
Execução de retornos	Execução(ões) de retorno(s) simultâneos de processos.	MV_WFMAXJB	Define a quantidade de processos de retorno que poderão ser executados por vez.
	Reativar processos automaticamente	MV_WFREACT	Caso esta opção seja selecionada, ocorrerá erro na execução de retornos. O Workflow reativará o processo imediatamente para ser executado de novo. Caso contrário, será reativado somente quando o <b>Scheduler</b> for reiniciado.
Tratamento de erros	Usar TRANSAÇÃO na execução de funções de "RETORNO" e "TIMEOUTs"	MV_WFTRANS	Habilita o recurso de transação com a finalidade de conservar a integridade dos dados em caso de falha de execução.
Notificação			
Opção	Item	Parâmetro	Descrição
E-mail do administrador	Endereço	MV_WFADMIN	Endereço eletrônico do administrador do sistema. Separe cada email por ponto e vírgula (;) <i>Exemplo: user1@advpl.com.br; user2@advpl.com.br</i>
Enviar notificação	Quando ocorrer erro ao executar funções "Retorno" e Timeout".	MV_WFNF001	Notificar por e-mail a lista de endereços dos administradores sobre o erro ocorrido.
	Ao reativar processos pendentes.	MV_WFNF002	Notificar por e-mail a lista de endereços dos administradores no momento em que forem reativados os processos que ocorreram erro.
	Ao receber mensagens não reconhecidas.	MV_WFNF003	Notificar por e-mail a lista de endereços dos administradores sobre as mensagens não reconhecidas pelo Workflow.

Messenger			
Opção	Item	Parâmetro	Descrição
Browser Internet	Caminho	MV_WFBROWS	Arquivo executável do browser Internet que deverá estar no <i>path</i> da estação.
	Servidor	MV_WFBRWSR	IP ou Nome de PIPE do servidor Protheus para uso do serviço http. Adicione ":" + a

<b>Diretório HTTP</b>	Caminho	MV_WFDHTTP	porta, caso seja diferente do padrão. Diretório de trabalho do serviço http. Verifique o identificador " <b>Path=</b> " na seção " <b>[HTTP]</b> " do arquivo <b>mp8Srv.ini</b> para obter o diretório de trabalho.
<b>Habilitar</b>	Habilitar Messenger automaticamente . (próximo login)	MV_WFMESSE	O <i>messenger</i> será executado automaticamente no próximo <i>login</i> de qualquer ambiente Protheus.

Após toda essa configuração você deve configurar o Inicializador do server com as seguintes seções:  
[ONSTART]

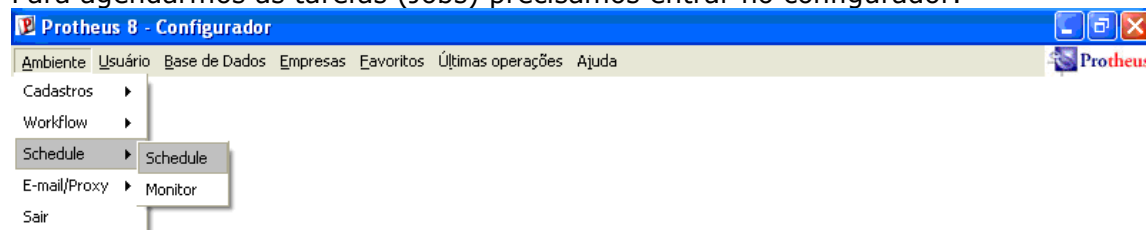
Jobs=WORKFLOW

[WORKFLOW]

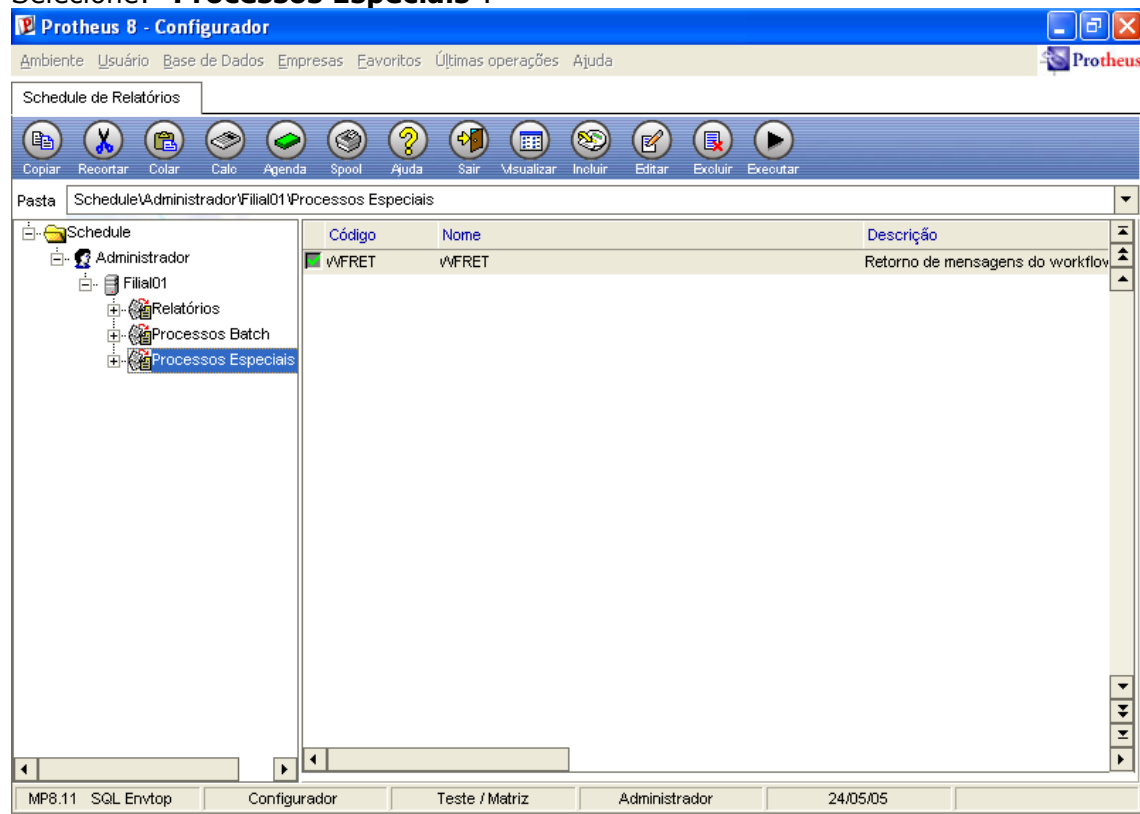
Main=WFONSTART

Environment=Enviromment

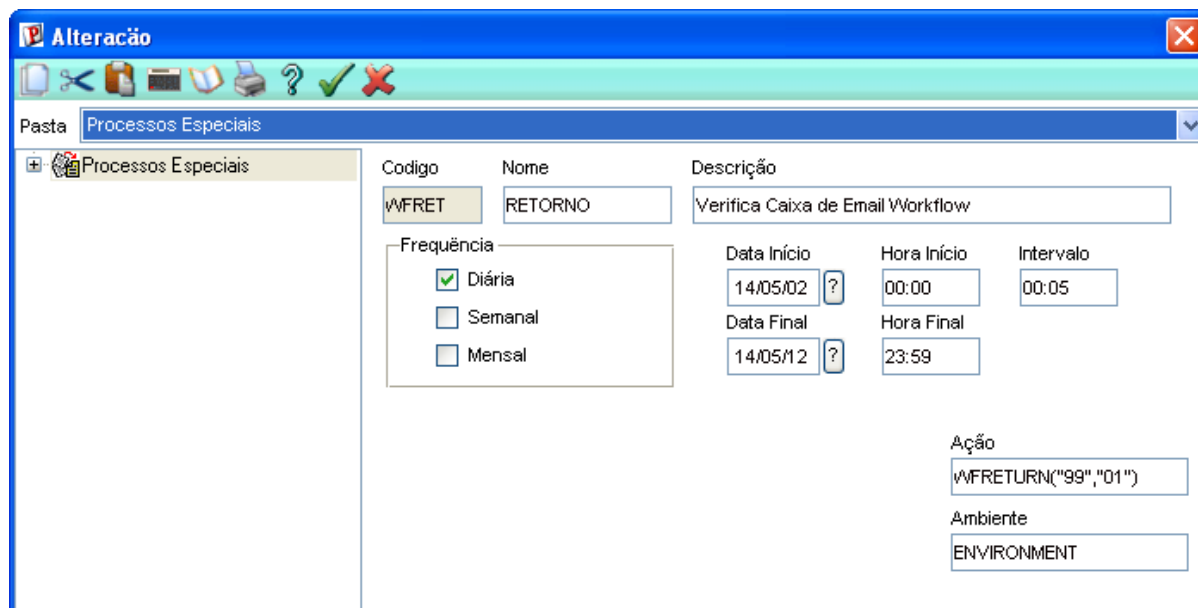
Para agendarmos as tarefas (Jobs) precisamos entrar no configurador.



Selecione: "**Processos Especiais**".



## Clique em Incluir



**Alteração**

Pasta: Processos Especiais

Processos Especiais

Código: WFRET Nome: RETORNO Descrição: Verifica Caixa de Email Workflow

Frequência:

☒ Diária  
☐ Semanal  
☐ Mensal

Data Início: 14/05/02 ? Hora Início: 00:00 Intervalo: 00:05

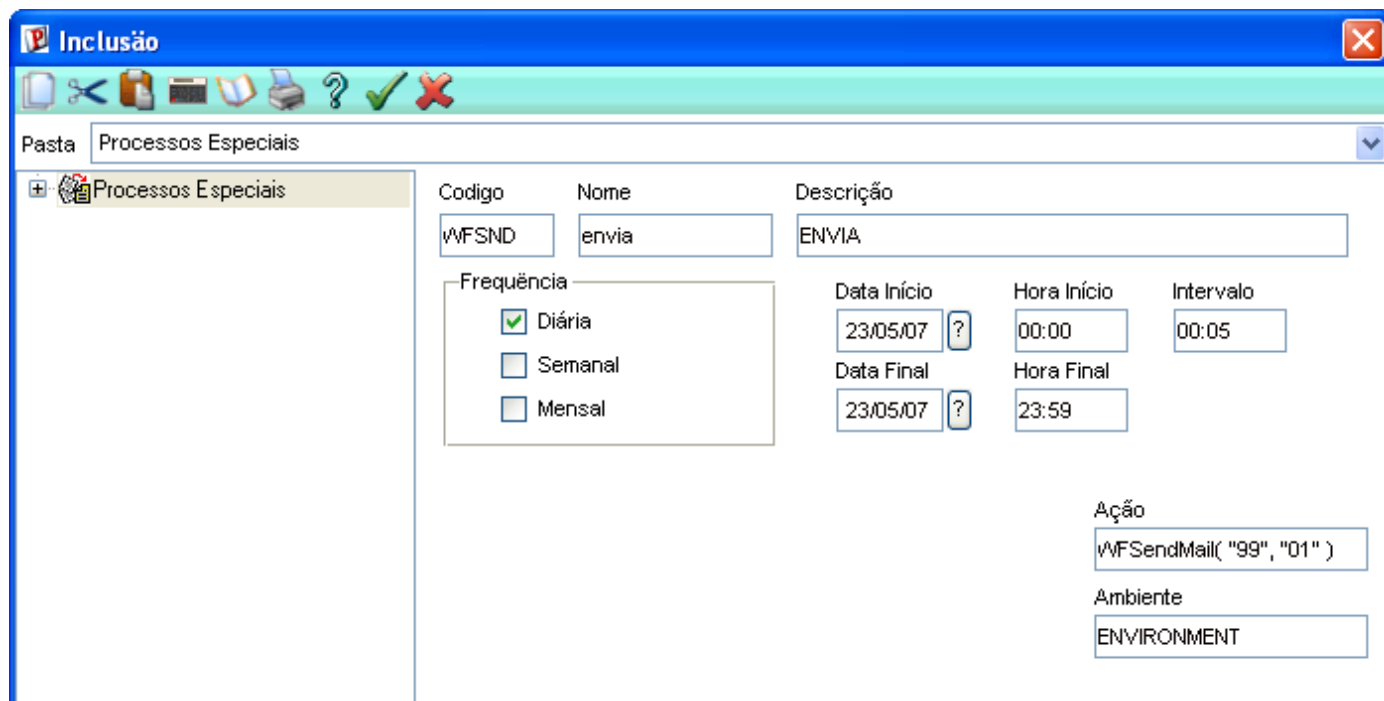
Data Final: 14/05/12 ? Hora Final: 23:59

Ação: WFRETURN("99", "01")

Ambiente: ENVIRONMENT

Scheduler – Processos Especiais		
Agrupamento	Campos	Descrição
Identificação	Código	Código único identificador do <i>job</i> .
	Nome	Nome simplificado.
	Descrição	Descrição do <i>job</i> .
Frequência	Diária	Executa diariamente entre os horários inicial e final a cada intervalo de tempo.
	Semanal	Executa nos dias selecionados da semana (domingo, segunda, terça, quarta, quinta, sexta e sábado), entre os horários inicial e final a cada intervalo de tempo.
	Mensal	Executa nos meses selecionados do ano (janeiro a dezembro), entre os horários inicial e final a cada intervalo de tempo.
Período Inicial	Data Início	Data inicial que passará a vigorar a execução.
	Hora Início	Horário inicial de execução.
Período Final	Data final	Dia limite de execução.
	Hora final	Horário limite de execução.
Período	Intervalo	Intervalo de tempo a cada execução entre o horário inicial e final.
Job	Ação	Função de usuário ou função interna do Protheus a ser executado. Os parâmetros passados para esta função deverão ser sempre tratados como um <i>array</i> .
Environment	Ambiente	Ambiente em que o <i>job</i> será executado.

É aconselhável configurar o recebimento automático de tempos em tempos caso haja algum problema de retardo de email ou gargalo de fluxo de email.



## Criação do HTML

Você deve criar um formulário em html para enviar no email pelo workflow, veja exemplo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
<p><font size="2" face="Verdana, Arial">Prezado Aprovador!</font></p>
<p><font size="2" face="Verdana, Arial">Ap&ocirc;e;s o lan&ccedil;amento abaixo,
o saldo de <strong>!NOME!</strong> ficou negativo em <strong>R$ !SALDO!</strong>
.</font></p>
<font size="2" face="Verdana, Arial">Nr. Transa&ccedil;&atilde;o:
<strong>!NUMERO!</strong></font>
<br><font size="2" face="Verdana, Arial">Item: <strong>!ITEM!</strong></font>
<br><font size="2" face="Verdana, Arial">Data: <strong>!DATA!</strong></font>
<br><font size="2" face="Verdana, Arial">Hist&ocirc;e;rico:
<strong>!HIST!</strong></font>
<br><font size="2" face="Verdana, Arial">Valor: <strong>!VALOR!</strong></font>
```

```
<form name="form1" method="post" action="mailto:%WFMailTo%">
  <p><font size="2" face="Verdana, Arial"><input type="radio" name="%APROVA%"
value="Sim">Aprova este lan&ccedil;amento</font></p>
  <p><font size="2" face="Verdana, Arial"><input type="radio" name="%APROVA%"
value="Nã&tilde;o aprova</font></p>
  <p><font size="2" face="Verdana, Arial"><input type="submit" name="Submit"
value="Enviar"></font> </p>
</form>
</body>
</html>
```

A rotina de workflow substitui todas as palavras encontradas nos formulários html que estiverem entre "%" ou "!". Essa palavra-chave terá seu conteúdo modificado por outra informação durante a execução do processo.

## Criação de Processos em ADVPL

Um processo de Workflow poderá ser iniciado a partir de uma destas opções:

- pontos de entrada;
- item de menu de qualquer ambiente;
- *job* agendado através do Scheduler;
- manualmente através do remote.

Na maior parte, o Workflow baseia-se, no uso de classes referenciadas aos objetos, que se tornam a ponte de acesso para uso de seus métodos e propriedades. A principal classe do Workflow é a "**TWFProcess**", o que veremos a seguir (é necessário conhecimento de programação orientada a objetos).

## Classe TWFPROCESS

A classe TWFProcess é responsável pela criação e gerenciamento do processo. Como toda classe, a TWFProcess é dividida em métodos e propriedades. Veremos alguns dos principais métodos e propriedades que iremos usar para criação de um processo.

### MÉTODOS

- **NEW** - O método New() é responsável pela criação e inicialização da classe WFProcess.

Sintaxe

:New(<cCodProc>,<cDescr>,<cProcID>)

Argumento	Obrig	Tipo	Descrição
cCodProc	Sim	C	Recebe o código do processo usado em "Cadastro de Processos".
cDescr	Sim	C	Recebe a descrição do processo que está sendo criado no momento. Se não for informado, será usada a descrição contida no cadastro de processo, localizada através do parâmetro anterior cCodProc



cProcID	Não	C	Recebe o ID do processo criado anteriormente. Normalmente, é utilizado para reconstruir um processo anterior, dando seqüência a ele
---------	-----	---	---

**Exemplo:**

```
oP := TWFFProcess():New("PEDCOM","Aprovação do Pedido de Compras")
```

- **NEWTASK** - Responsável em criar a seqüência de tarefas a ser executada e identificar o html que será utilizado.

Sintaxe

```
:NewTask(<cDescr>,<cArqHtml>,<lCopiar>)
```

Argumento	Obrig	Tipo	Descrição
cDescr	Sim	C	Recebe a descrição da tarefa.
cArqHtml	Sim	C	Recebe o caminho e o nome do arquivo html que fará uso no processo.
lCopiar	Não	L	Copia os campos utilizados na tarefa anterior para a tarefa que está sendo criada.

**Exemplo:**

```
oP:NewTask("Criando Aprovação", "\\Workflow\WFW120p.htm" )
```

- **ATTACHFILE** - Responsável pelos arquivos anexos à mensagem, que deverão estar dentro do rootpath do Protheus.

Sintaxe

```
:AttachFile(<cArquivo>)
```

Argumento	Obrig	Tipo	Descrição
cArquivo	Sim	C	Caminho e nome do arquivo a ser anexo à mensagem

**Exemplo:**

```
oP:AttachFile("\\Workflow\teste.txt")
```

- **START** - Responsável por construir o processo, gravar nas tabelas do Workflow e enviar mensagem aos destinatários. O retorno é uma chave composta por **ProcessID+TaskID** composto por 17 algarismos hexadecimais.

Sintaxe

```
:Start(<cHtmlCopiarPara>) -> cProcessKey
```

Argumento	Obrig	Tipo	Descrição
cHtmlCopiarPara	Sim	C	Caminho em que o Workflow deverá realizar uma cópia do html final.

**Exemplo:**

```
cID := oP:Start("\\Workflow\copia")
if file( "\\Workflow\copia\" + cID)
    conout("Arquivo copiado com sucesso.")
endif
```



**FINISH** - Este método é responsável por finalizar o processo. Após a finalização, ele não estará mais disponível para execuções do tipo retorno e timeout.

Sintaxe  
:Finish()

**Exemplo:**

oP:Finish()

- **TRACK** - Responsável por incluir as descrições dos passos seguidos pelo fluxo do processo e apresentá-los na consulta da rastreabilidade.

Sintaxe

:Track(<cCodStatus>,<cDescr>,<cUsuario>)

Argumento	Obrig	Tipo	Descrição
cCodStatus	Sim	C	código do status do processo.
cDescr	Sim	C	descrição do passo ocorrido.
cUsuario	Sim	C	nome do usuário a que se destinou a tarefa.

**Exemplo:**

oP:Track("100200", "Enviando o pedido para aprovação", "AprovadorA")

**PROPRIEDADES**

**:cTo, :cCC e :cBCC**

Estas propriedades definem o endereço dos destinatários. Para informar mais que um destinatário, basta incluir um ponto-e-vírgula ";" entre eles. Se for informada uma palavra qualquer que não seja um endereço de e-mail válido, o Workflow considera que é um diretório e deverá gerar o HTML. Pode-se mesclar os tipos.

**Exemplo:**

oP:cTo := "aluno1@advpl.com.br;aluno2@advpl.com.br"

oP:cCC := "aluno3@advpl.com.br;Aluno"

**oP:cBCC := "aluno4@advpl.com.br"**

**:cSubject**

Esta propriedade define o assunto da mensagem.

**Exemplo:**

oP:cSubject := "Aprovado do pedido de compras no. 1028"

**:cBody**

Esta propriedade armazenará um texto que permanecerá no corpo da mensagem. Caso seja utilizado, o html irá como anexo da mensagem.

**Exemplo:**

oP:cBody := "Testando..."

**:bReturn**

Esta propriedade contém o nome da função que será executada no momento em que o Workflow receber a mensagem de resposta de um dos destinatários via e-mail ou serviço http.

**Exemplo:**

oP:bReturn := "U\_Retorno"

**:bTimeout**

Esta propriedade recebe um *array* de *timeouts* contendo nomes das funções e tempo de espera. Caso o tempo seja alcançado, serão executadas as funções mencionadas no 1º item do *array*. Poderão ser especificados mais do que um *array* de *timeouts*.

Sintaxe:

```
{ { <cFuncao>, <nDias>, <nHoras>, <nMinutos> }, { ... } }
```

**Exemplo:**

```
oP:bTimeOut := { { "TimeOut1", 0, 5, 30 } }
```

ou

```
oP:bTimeOut := { { "TimeOut1", 0, 5, 30 }, { "TimeOut2", 1, 10, 0 } }
```

### **:fProcessID**

Esta propriedade fornece o número ID do processo.

**Exemplo:**

```
cProcID := oP:fProcessID
```

### **:fTaskID**

Esta propriedade fornece o número ID da tarefa criada para um determinado processo, através do método **:NewTask()**.

**Exemplo:**

```
oP:NewTask( "100100", "\\Workflow\WFW120p.htm" )
```

```
cTaskID := oP:fTaskID
```

### **:oHTML**

Esta propriedade é responsável pelo tratamento das palavras-chaves no html mencionado no método **:NewTask()**. Esse objeto é uma referência da classe TWFHtml.

### **CLASSE TWFHTML**

A classe TWFhtml é responsável pela leitura do HTML de retorno e de envio para tratamento das variáveis. Veremos alguns dos principais métodos e propriedades que iremos usar.

### **MÉTODOS**

- **VALBYNAME** - Este método tem como objetivo atribuir ou obter um valor a uma "macro" existente no html. Deverá ser usado somente no momento em que estiver enviando workflow ou na função de timeouts, em que o uso é necessário por motivo de não haver recebido resposta.

**:oHTML:ValByName(<cMacro>,<uConteudo>)**

Argumento	Obrig	Tipo	Descrição
cMacro	Sim	C	nome da macro (palavra-chave) encontrada no html e identificada entre os símbolos "%" e "!".
uConteudo	Sim	C	valor a ser atribuído à macro.

**Exemplo:**

**Assinalando um valor:**

```
oP:oHtml:ValByName( "Nome", "Aluno1" )
```

**Usando na função de timeout:**

```
cNome := oP:oHtml:ValByName( "Nome" )
```

**Atribuindo um valor a uma tabela:**

```
AAdd( oP:oHtml:ValByName( "produto.codigo" ), SB1->SB1_COD )
```

- **RETBYNAME** - O método `retbyname()` é responsável pelo tratamento das variáveis de retorno do workflow. O método somente deverá ser usado na **função de retorno**.

Sintaxe:

`:oHTML:RetByName( cMacro )`

Argumento	Obrig	Tipo	Descrição
cMacro	Sim	C	nome da macro (palavra-chave) encontrada no html e identificada entre os símbolos "%" e "!" no formulário de retorno.

**Exemplo:**

`cNome := oP:oHtml:RetByName( "Nome" )`

**Em uma tabela:**

`aCodigo := oP:oHtml:RetByName( "produto.Codigo" )`

## Capítulo 05 – WEB Service

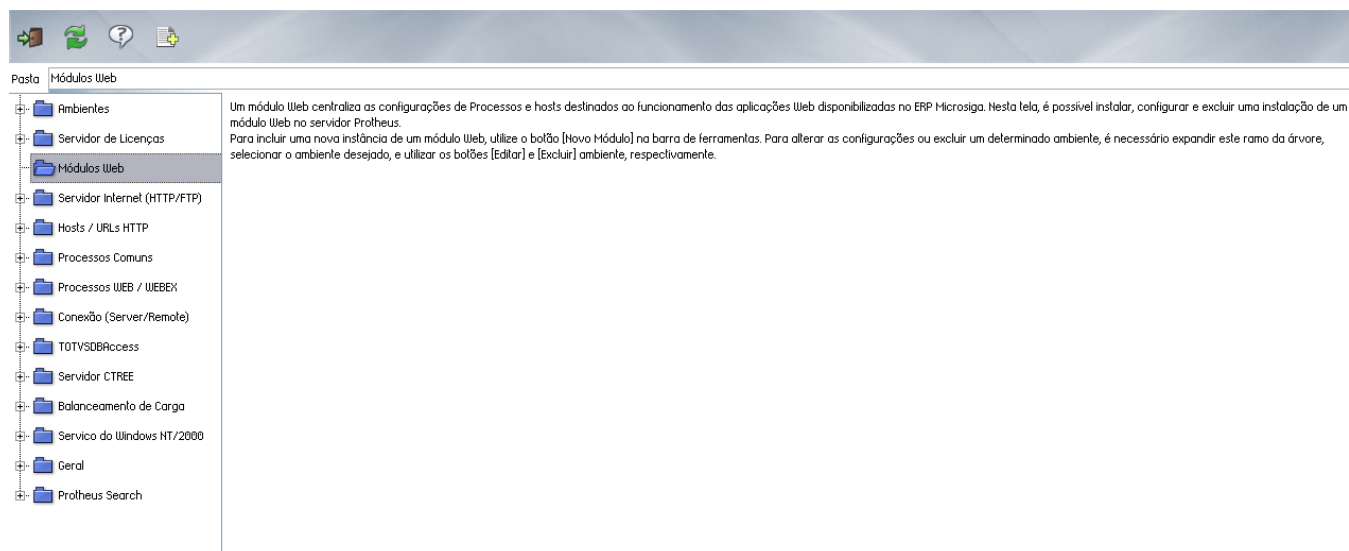
Quando um Web Service é criado e disponibilizado, junto dele também é disponibilizada a definição do serviço, seus argumentos, estruturas e retornos (WSDL). Para nos utilizarmos de um Web Service, precisamos montar um programa - Client que seja capaz de montar um "envelope" SOAP com os dados necessários ao processamento do Serviço, realizar a chamada, e tratar o respectivo retorno e exceções.

Embora existam Web Services que podem ser acessados via Http "direto", apenas passando parâmetros via URL, o Client de Web Services do Advanced Protheus tem seu foco e recursos direcionados apenas à serviços que possuam interface de comunicação que realize POST de pacotes de dados XML em formato SOAP.

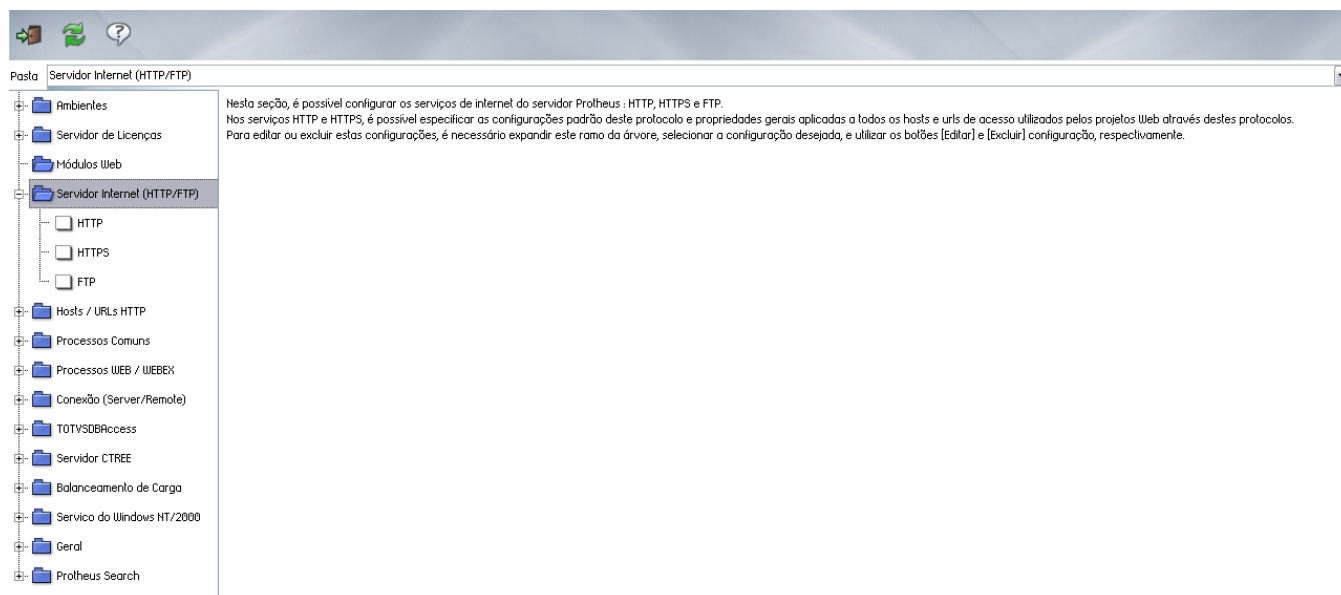
Para a codificação de um webservice, foram criadas em Advpl instruções especiais de declaração de classe, específicas para WebServices, que suportam nomes 'longos' no nome da classe, métodos e propriedades. A utilização destes comandos exige a declaração do `#include 'apwebsrv.ch'` no topo do código-fonte; e exige também a atenção em alguns pontos e particularidades, a iniciar pela nomenclatura do serviço, estruturas, métodos e propriedades.

### Configuração

Tanto WebService quanto ADVPL ASP devem obrigatoriamente ser configurados pelo Wizard, pois são descompactados pacotes necessários para o perfeito funcionamento da rotina.



Configuração HTTP com o Wizard é possível configurar os serviços de Internet do Servidor Protheus: HTTP, HTTPS e FTP



## Exercício

Criar um Web Service de Saldo de Produtos:

-Recebe:Código do Produto (B1\_COD)

-Retorno de acordo com 3 métodos

1. Saldo do Produto (B2\_QATU)
2. Saldo do Produto m Valor (B2\_VATU1)
3. Quantidade Disponível (B2\_QATU-B2\_RESERVA-B2\_QEMP)

Criar um Cliente para esse Web Service para testá-lo usando os 3 métodos.

## Capítulo 06 – ADVPL ASP

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.

### 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\_aph

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 Microsiga, por tratar-se de um Aph de Usuário, equivalente à 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:

```
<html><head><title>ADVPL ASP Demo</title></head>
<body>
<p>Bem vindo ao mundo do ADVPL ASP!</p>
<%
// Soma os 100 primeiros números
Local i, nSoma := 0
For i := 1 To 100
```

```
NSoma += i
Next i
%>
</body>
</html>
```

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:

**http://localhost/H\_WEBDEMO.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 podem 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 Browser:

```
<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 ficam 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 :

1. 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:

#### **CERTO**

```
<% IF !IOk %>
```

```
<% nErro++ %>
```

```
<% Endif %>
```

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

```
<%
```

```
IF !IOk
```

```
nErro++
```

```
Endif
```

```
%>
```

```
IF !IOk
```

```
nErro++
```

```
Endif
```

#### **ERRADO**

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

2. 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.

3. Uma linha qualquer em um arquivo .APH não 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.

### **Desenvolvimento de Funções .APL**

A princípio, todas as funções contidas no repositório podem ser executadas diretamente através de uma requisição HTTP, via link com extensão .apl, ao Protheus Server. Porém, alguns detalhes devem ser considerados:

- Uma função executada no momento do recebimento de uma requisição HTTP é executada como um JOB, ou seja, não contém interface. Por isso, tais funções não podem conter comandos de interface, como criação de janelas ou exibição de helps e mensagens de alerta;



- A única interface possível é a utilizada no client HTTP. Por isso, tais funções devem SEMPRE retornar uma string de caracteres. Após o processamento da função, essa string de retorno será enviada diretamente ao client HTTP e este será o responsável por sua interpretação. Por exemplo, utilizando um Web Browser como client pode-se retornar a string de comandos HTML diretamente. O HTML então será propriamente exibido no Web Browser;

- Qualquer retorno diferente de uma string de caracteres gerará um erro que será enviado à aplicação client HTTP (o erro gerado é ?Invalid Proc Return?);

- O servidor Protheus passa alguns parâmetros para as funções chamadas. Isso significa que ao criar funções para serem utilizadas em resposta às requisições HTTP, deve-se criar o cabeçalho da função com estes parâmetros. Não é obrigatório utilizar os mesmos nomes de parâmetros sugeridos abaixo quando criar diretamente estas funções. Porém, como são esses os nomes utilizados no ADVPL ASP explicado mas a frente, é aconselhável utilizá-los por motivo de padronização:

\_\_aCookies: Este parâmetro recebe um array bidimensional com os Cookies criados na aplicação client HTTP (por exemplo, no Internet Explorer 5). Pode-se utilizá-lo para checar validações mantidas nas máquinas client por exemplo. Para maiores detalhes, consulte a documentação do HTML ou do Web Browser utilizado.

\_\_aPostParms: Este parâmetro recebe um array bidimensional com os campos contidos em um formulário HTML recebido através de um comando POST. Cada item deste array contém um array com o nome do campo e o valor informado. Por exemplo, para um formulário com dois campos para digitação (um chamado nome e o outro chamado endereço), que enviam os dados para a função cadastro.apl através de um POST, a função receberá o array \_\_aPostParms da seguinte forma:

```
{ {?nome?, ?NOME DIGITADO NA PAGINA HTML?},  
  
  {?endereco?, ?ENDERECO DIGITADO NA PAGINA HTML?} }
```

A função pode tratar os dados recebidos neste array para realizar um processamento específico com tais informações. Para campos onde não é possível a entrada de dados e sim a escolha de uma informação pré-definida (como por exemplo um checkbox), o item somente existirá no array caso o campo tenha sido selecionado no formulário HTML (por exemplo, se o checkbox for marcado).

\_\_nProcID: Contém o Handle da Thread de execução daquela função. A utilização deste parâmetro será explicada juntamente com o tópico ADVPL ASP posteriormente;

\_\_aProcParms: Este parâmetro recebe um array bidimensional com os parâmetros enviados na linha de URL do Web Browser. Por exemplo, na execução de uma função via linha de URL do Web Browser como:

`http://servidor/vende.apl?cod=000001&nome=PRODUTO DE TESTE&quant=20`

a função chamada vende receberá o array \_\_aProcParms da seguinte forma:

```
{ {?cod?, ?000001?},  
  
  {?nome?, ?PRODUTO DE TESTE?},  
  
  {?quant?, ?20?} }
```

A função pode tratar estes dados recebidos para realizar processamentos específicos. É muito útil também para criar links de execução diretamente através de um Web Browser.

◦ \_\_cHTTPPage: Esse parâmetro foi criado originariamente para receber o nome da página, arquivo ou função originalmente requisitada para o Protheus Server, porém não foi utilizado e permaneceu por compatibilidade. Caso consultado, ele retorna uma string em branco.

◦ \_\_aHTTPHead: Esse parâmetro recebe um array com os Headers do cabeçalho da requisição HTTP enviados pelo Web Browser.

### Exemplo de função APL

A função a seguir é um bom exemplo para ser executado através de um Web Browser. Ela retorna uma string contendo a página HTML onde está escrita a mensagem ?Hello World? e a lista de parâmetros passados na linha de URL. Para testá-la, crie um arquivo novo no Protheus IDE, cole o código abaixo e salve o arquivo como WEBDEMO.PRW. Após compilar o programa, basta chamar em um Web Browser uma URL como:

```
http://localhost/u_webdemo.apl?cod=000001&desc=DESCRICAO DO PRODUTO&qtd=2
```

### Código da função:

```
#include 'rwmake.ch' User Function
```

```
WebDemo(__aCookies,__aPostParms,__nProcID,__aProcParms,__cHTTPPage) Local cHTML := "Local  
i// Coloca uma mensagem em HTML cHTML += '<p><h1 align='center'>Hello World!!!</h1></p>'  
// Coloca um separador de linha em HTML cHTML += '<hr>' If Len(__aProcParms) = 0 cHTML  
+= '<p>Nenhum parâmetro informado na linha de URL.' Else For i := 1 To Len(__aProcParms)  
cHTML += '<p>Parâmetro: ' + __aProcParms[i,1] + ' - Valor: ' + __aProcParms[i,2] +  
'</p>' Next i Endif Return(cHTML) Importante
```

Para criar as funções que serão utilizadas em chamadas via um Web Browser, ou seja, em qualquer request HTTP, deve-se seguir o procedimento normal de criação de funções no AP5: utilizando o AP5 IDE para a edição e para a compilação.

Note que no caso de funções do usuário (User Function) o nome chamado na URL do Browser também deverá conter o U\_ no começo da função, por exemplo:

```
http://servidor/u_WebDemo.apl
```

## Configuração Mínima

Em tópico à parte é explicada toda a configuração referente à seção http do Protheus Server. A configuração abaixo é a mínima necessária para executar o exemplo acima

[http]

Port=80

Path=(caminho absoluto de disco para arquivos publicados no servidor )

Environment=(nome do environment do Serber que será utilizado para o processamento)

Para testar sua configuração, reinicie o server Protheus e chame via WebBrowser a url :

`http://localhost/time.apl`

Deverá ser mostrada no Browse o horário atual, no formato hh:mm:ss, no Servidor, pois este é o resultado da função Advpl TIME() .

O Protheus atendendo à requisições .apl

Quando solicitado através de um Web Browser um processamento de uma função via link .apl, a função solicitada deve ser responsável por abrir o ambiente necessário ao processamento, conectar com o Banco de Dados caso necessário, realizar o processamento e retornar a String Html ao Web Browser.

Este ambiente criado é fechado imediatamente após o término do processamento, o que exige muito do Servidor da aplicação em se tratando de uma aplicação web com vários usuários efetuando acessos simultâneos. Para atender com mais eficiência às requisições de processamento de um projeto web, foi implementada no Protheus a tecnologia de 'Working Threads', explicada em mais detalhes no tópico 'Desenvolvimento de Funções .apw'.