

Desvendando o CGI com o FLAGSHIP

Através do FlagShip, que utiliza linguagem igual à do Clipper, é possível desenvolver os mais diversos produtos para a Internet

JORGE E. HIME SOMERS
jhsomers@somers.com.br

Vamos falar um pouco sobre como colocar um site na Internet, interagindo com as suas bases de dados do FlagShip. Praticamente todos que têm acesso à Internet já visitaram algum site de procura ou um banco comercial, e em alguns deles puderam observar que o endereço é muito longo, como em: cgi.netscape.com/cgi-bin/rlcgi.cgi?URL=www.somers.com.br. Nesse exemplo está

nítido que o servidor da Netscape está executando um programa que receberá o endereço www.somers.com.br. Já no caso de bancos comerciais como o Itaú, poderemos ver um endereço como bankline.itau.com.br/GRIPNET/montTela.exe, e nesse caso é nítido que o servidor do Itaú está executando um programa externo ao servidor Web.

Esses dois exemplos escolhidos aleatoriamente têm em comum apenas o fato de estarem executando um programa externo, mas o método utilizado difere em muitas coisas. No primeiro exemplo, os dados são enviados na mesma linha do endereço e o dado enviado para o programa `rlcgi.cgi` é o URL, com o valor de www.somers.com.br. Já no caso do Itaú os dados são enviados de forma a não aparecerem na linha de endereço. Então como os dados são enviados?

Se você pedir para ver o fonte, verá que cada imagem tem um formulário específico que tem um campo chamado ID que não aparece na tela, que tem um dado diferente. Cada vez que você executa um novo login, esse dado muda,

TABELA 1

```
CONTENT_LENGTH=232 → Quantidade de dados de formulário contendo??? contendo nome dos campos
CONTENT_TYPE=application/x-www-form-urlencoded → O tipo de dado
DOCUMENT_ROOT=/home/httpd/html → Diretório raiz físico do domínio
HTTP_COOKIE=MEUCOOKIE=VALOR DO COOKIE → Todos os cookies para o endereço separados por ;
HTTP_PRAGMA=no-cache → Se a página ficará arquivada no cache???cachê? do cliente
HTTP_REFERER=*http → //linux/cgi-bin/fsweb.cgi De onde veio a requisição, sites de procura etc..
HTTP_USER_AGENT=Mozilla/4.7 [en] (Win98; U) → Nome do browser do cliente, neste caso foi utilizado Netscape 4.7
QUERY_STRING=variable=jorge → Quais os dados que foram enviados pela linha de comando
REMOTE_ADDR=200.0.0.2 → IP do cliente
REQUEST_METHOD=POST → Tipo de requisição
REQUEST_URI=/cgi-bin/fsweb.cgi?variable=jorge → Linha de comando sem o domínio e com as variáveis
SCRIPT_FILENAME=/home/httpd/cgi-bin/fsweb.cgi → Path completo do CGI
SCRIPT_NAME=/cgi-bin/fsweb.cgi → Path a partir do diretório raiz do domínio sem variáveis
SERVER_ADMIN=webmaster@somers.com.br → E-mail do administrador
SERVER_NAME=linux.somers → Nome do servidor
SERVER_PORT=80 → Porta 80 = HTTP, 443 = HTTPS
```

por motivos de segurança, e é assim que o Itaú controla o acesso às contas correntes. Esta é uma forma segura e muito simples de se implementar segurança e flexibilidade em uma tela com várias opções.

Uma dúvida freqüente, quanto ao programa CGI, é se ele fica “rodando” no servidor. Não fica, ele é executado a cada requisição de cada cliente, e o programa não “permanece rodando”. Vejamos então como funciona:

1. Você pode iniciar o seu CGI de duas formas:

- Por um HTML, com um formulário a ser enviado ao servidor
 - Diretamente no CGI, caso em que precisa prever que pode haver uma requisição sem dados de formulário
2. Uma vez enviado o dado para o CGI o programa é iniciado e captura as variáveis, se houver.
3. É gerado um HTML resposta, que existe apenas na tela do cliente.

Fisicamente o HTML não existe, ou seja, não é gravado arquivo temporário algum, o qual contém um outro formulário para que o usuário possa continuar a navegar pelo programa.

4. O programa é encerrado

E esse processo se repete a cada requisição de cada cliente.

Como você pode ver, o programa CGI não fica parado esperando alguma resposta de teclado. Em suma, o CGI é um programa que não interage diretamente com o cliente, mas sim com um servidor Web, e todas as suas saídas são direcionadas para o servidor Web, e é possível enviar dados para impressoras (locais ou redirecionadas pelo servidor), arquivos em disco, etc.

Então muito cuidado se você costuma utilizar a seguinte estrutura:

```
while .t.  
  @ 10,10 say "nome" get nome  
  read  
  if !empty(nome)  
    exit  
  else  
    alert("erro nome não preenchido")  
  loop  
endif  
enddo
```

Pois essa estrutura não pode ser utilizada em um CGI.

Como é que o programa recebe os dados do formulário?

Agora que entendemos as formas de se enviar os dados para o programa, devemos entender mais alguns fatos sobre como o dado chega no programa. Quando o cliente clicar sobre uma imagem ou botão, os dados são enviados para o servidor Web que se encarregará de disponibilizar todos os campos e seus valores, e mais alguns dados que poderemos utilizar de várias formas, por exemplo: como verificador de cliente, no caso de utilizarmos LPs (Linhas Privativas), para verificar se estamos utilizando um servidor seguro etc.

Você pode ver na tabela 1 alguns dos dados de minha intranet no meu servidor Web juntamente com uma explicação resumida.

Se você está usando um servidor seguro (HTTPS) verá também vários dados iniciados por SSL. Pode-se ver alguns desses dados na tabela 2

Os dados do formulário?

Para simplificar a explicação, esses dados são lidos diretamente do servidor Web pelo FS_WEB.

Então para facilitar o acesso a esses dados eu criei uma variável somente para o FS_WEB, que se chama Content_String, uma alusão às variáveis Content_Length e Query_String, que contém todos os dados do formulário enviado. Os dados contidos nessa variável são de difícil manuseio, pois todos os caracteres que não estão entre a/z, 0/9 ou A/Z, são substituídos pelo seu código hexadecimal, processo que é conhecido como “escape”, e cada campo é separado por & e cada conteúdo de variável inicia com =, como podemos ver abaixo:

```
CONTENT_STRING=Escondido=2345&Nome=&Endereco=  
&LongTexto=Area+ definida%3A+6+linhas+30+colunas+  
com+wrap+autom%Etico+m%Elximo+de+200+  
caracteres&cTipoDanca_1=BSEG&cTipoDanca_2=  
RTER&TESTE1=LISTA2&TESTE2=MLISTA1&TESTE3=  
LISTA3&ticavel=cLing1& Check=DBASE
```

Com a biblioteca FS_WEB o processo de “unescape” (reversão de um “escaped string”), e de recuperação de dados, se torna totalmente transparente e muito simples como podemos ver na próxima página:

TABELA 2

SSL_CIPHER_ALGKEYSIZE=128 Æ Tamanho da chave de criptografia do servidor

SSL_CIPHER_EXPORT=false Æ Se o browser do cliente é para exportação apenas, entende-se 40 bits apenas

SSL_CIPHER_USEKEYSIZE=128 Æ Neste caso estou usando um Netscape com chave de 128 bits.

cgi connect():Transfere todos os dados para o executável do FlagShip getField "op" at cOp:Transfere o valor do campo op do formulário para a variável cOp.

E para simplificar mais ainda, os nomes dos campos do formulário não são "case sensitive", ou seja, tanto faz se estão em caixa alta ou baixa, logo não se pode ter dois campos no formulário com o mesmo nome. Exemplo:

Nome, nome.

Fato que seria totalmente aceitável se utilizássemos JavaScript, pois para oJavaScript, nome e Nome são dois dados totalmente diferentes.

Finalização de tags

O FS_WEB ajuda muito na simplificação do uso dos comando de HTML, normalmente chamados de Tags, pois a linguagem utilizada é bastante simplificada, mas a maioria dos recursos está contemplada.

Exemplo: se desejamos apresentar um texto em itálico e negrito com letras vermelhas escrevemos:

```
SAY TEXT "BOM DIA" NEGRITO ITALICO COLOR NS_RED  
ou
```

```
SAY TEXT "BOM DIA" BOLD ITALIC COLOR NS_RED  
a ordem dos atributos não tem importância alguma, e o  
resultado na tela será:
```

BOM DIA

E o FS_WEB enviou:

```
<FONT COLOR="#FF0000"><B><I>BOM DIA</I></B></FONT>
```

Como você pode ver não foi necessário se preocupar com a finalização de cada um dos atributos, pois neste caso o FS_WEB faz isto por você. Já em outros casos isto se torna muito mais fácil pois é utilizada a sintaxe Begin / End veja abaixo:

```
BEGIN TABLE BORDER 0 ALIGN CENTER WIDTH "100%"  
BGCOLOR NS_BLUE  
CELLPADDING 0 CELLSPACE 0  
  BEGIN ROW  
    BEGIN DATA ALIGN CENTER  
      MENU LINK "MENU" MENU "Main" ONCLICK  
    END DATA  
  END ROW  
END TABLE
```

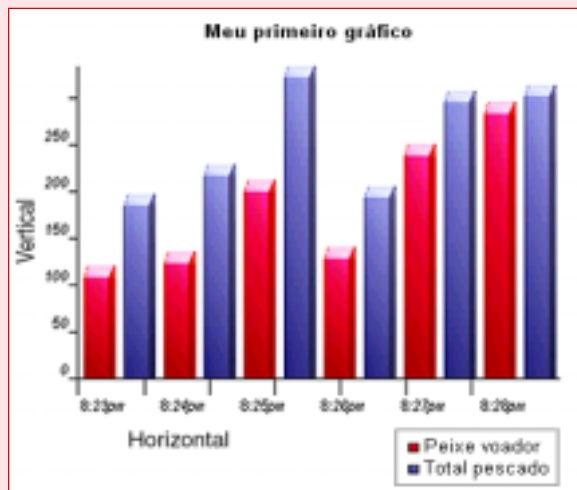
Geração de gráficos

Também a geração de gráficos tornou-se muito simples com o uso do FS_WEB, pois nele incorporei uma rotina que faz todos os cálculos quase automaticamente. No

exemplo abaixo, apenas gráficos de barras verticais estão contemplados.

```
BEGIN GRAPH g1 size 300,200  
  GRAPH DATA 124,138,216,143,256,302  
  GRAPH DATA 201,234,340,210,314,320  
  GRAPH ATRIBUTES STACKED  
  GRAPH ATRIBUTES SCALE "50"  
  GRAPH ATRIBUTES TITLE "Meu primeiro gráfico"  
  GRAPH ATRIBUTES XLABEL "Horizontal"  
  GRAPH ATRIBUTES YLABEL "VERTICAL"  
  GRAPH ATRIBUTES TIME "20:23"  
  GRAPH ATRIBUTES LEGEND "Peixe voador",  
  "Total pescado"  
END GRAPH
```

Este código resultará no seguinte gráfico:



É possível utilizar um código JavaScript?

Sim, é possível e muito simples, desde que você tenha algum conhecimento dessa linguagem. No exemplo abaixo, você pode ver como fazer para que, ao se alterar um dado do formulário, ele seja procurado na base de dados e seja emitido um alerta no caso de já estar cadastrado, tudo isto feito em uma segunda janela, que não tem menu, barra de navegação etc. Veja tabela 3.

Na linha Get name Size 30 Maxlen 30 Onchange testname() é gerado um código HTML. Ao ser feita alguma alteração nesse campo, será executada a função testname(), que está logo acima.

A função testname() não abrirá uma nova janela se o nome estiver vazio, senão será criada uma nova janela (nova=window.open(...)) e nela será criado um novo formulário que será enviado para o CGI automaticamente

TABELA 3

```

say SCRIPT ;
“function testname() {“+CRLF+“ var retval = false “+CRLF+
“if (document.adddata.name.value.length=0) {“+CRLF+“retval = false “+CRLF+“}“+CRLF+;
“else { “+CRLF+“document.adddata.name.focus()“+CRLF+;
“nova=window.open(‘’, ‘erro’, ‘toolbar=0, location=0, directories=0, status=1, menubar=0, scrollbars=1, ;
resizable=1, width=350, height=100, dependent=no, screenX=440, screenY=20’)“+CRLF+;
“ nova.document.write(‘<body>’)+CRLF+;
“ nova.document.write(‘<form name=teste method=post action=/cgi-bin/cad.sh>’)+CRLF+;
“ nova.document.write(‘<input type=hidden name=op value=showname>’)+CRLF+;
“ nova.document.write(‘<input type=hidden name=name>’)+CRLF+;
“ nova.document.write(‘</form>’)+CRLF+;
“ nova.document.teste.name.value=document.adddata.name.value”+ CRLF+;
“ nova.document.write(‘<script>’)+CRLF+;
“ nova.document.write(‘document.teste.submit()’)+CRLF+;
“ nova.document.write(‘</script>’)+CRLF+;
“ nova.document.write(‘</body>’)+CRLF+;
“ }“+CRLF+;
“return retval”+CRLF+;
“}“+CRLF
begin form “adddata” action “/cgi-bin/cad.sh” hidden “op” value “adddata”
GET name SIZE 30 MAXLEN 30 ONCHANGE testname()

```

document.teste.submit()), somente então o CGI irá processar a verificação da requisição.

Todo esse processo pode parecer muito lento, mas apesar de muitos de nós não termos um link de mais de 56kbps, isto fica muito rápido, desde que:

1. As telas resultantes não contenham mais de trezentos registros com quatro ou cinco colunas.
2. Não existam tabelas muito grandes.

Outro fato que torna a velocidade de resposta muito boa, é que no caso do Linux, a resposta gerada pelo CGI é enviada diretamente para o cliente, ao contrário de outros sistemas operacionais que geram um arquivo intermediário para depois enviá-lo ao cliente.

Veja no site o código completo desse exemplo no arquivo FSWEBCAD.PRG.

Outras perguntas comuns que envolvem transações via Internet:

É possível utilizar o FS_WEB para e-commerce?

Sim, certamente que sim, mas lembre-se de que para realizar transações com número de cartões de crédito é necessário dispor de um servidor com o que costumamos chamar de “site seguro”, o que independe do FS_WEB e tem a ver só com seu servidor Web. Para se ter o que melhor nível de segurança, é aconselhável utilizar o módulo ssl do apache (mod_ssl), que pode ser encontrado em www.modssl.org, o

que permite a instalação de um site com até 128 bits de criptografia, com uma chave a ser comprada de qualquer provedor de chave de criptografia como VeriSign e muitos outros. Não utilize a chave de demonstração, ela não é realmente segura, ela imita a segurança de 128 bits.

Chave 128bit para o Netscape?

Para que o seu Netscape tenha um maior nível de criptografia utilize o programa Fortify (www.fortify.net). Ele eleva seu Netscape para um nível de segurança maior. Não é ilegal mesmo para os que não moram nos Estados Unidos, mas é sempre bom verificar se sua empresa permite que você utilize uma chave de 128 bits. Se você está utilizando o Netscape que veio com o Linux-Conectiva 4.0 não é necessário utilizar o Fortify pois já está atualizado. Antes de baixar o programa, verifique se o seu Netscape já não está fortificado?! É só utilizar o menu do Netscape em Help → About se aparecer na tela algo como: This version supports U.S. Security with RSA Public Key Cryptography, MD2, MD5, RC2-CBC, RC4, DES-CBC, DES-EDE3-CBC .

O Netscape já está fortificado?

Caso você tenha alguma dúvida, dica ou comentário não deixe de nos enviar para que esta coluna traga sempre novidades para todos.

Veja este programa sendo executado na Internet em www.somers.com.br – em FS_WEB → Demonstração 2. 