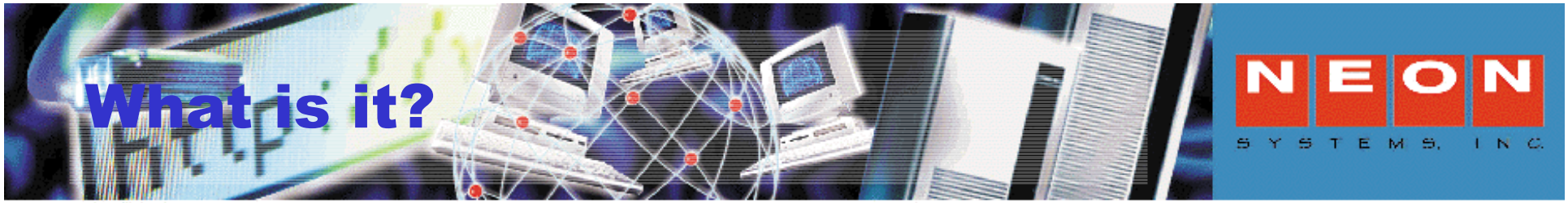
Become a Web Server Power User

Session M30

Michael Ludé -- mlude@neonsys.com

NEON Systems, Incorporated

http://www.neonsys.com



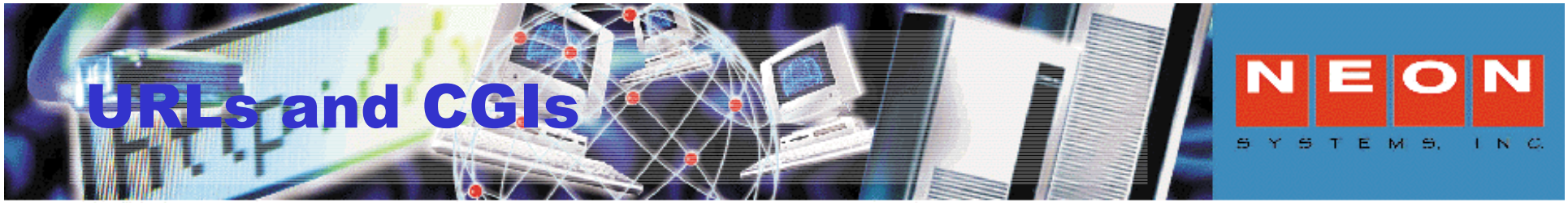
- CGI = Common Gateway Interface
- It's a user-written program running on the server
  - On Shadow, written in REXX with Pipelines
  - Other servers use C or Perl
- You have access to everything on the server
  - Probably want to run an application

- Called as a REXX Pipelines filter
  - Three input streams (two on Webshare)
  - One output stream
- Use Pipes to manage streams:
  - PEEKTO and READTO commands for input
  - OUTPUT command for output
  - SELECT and STREAMSTATE for control

A Trivial CGI

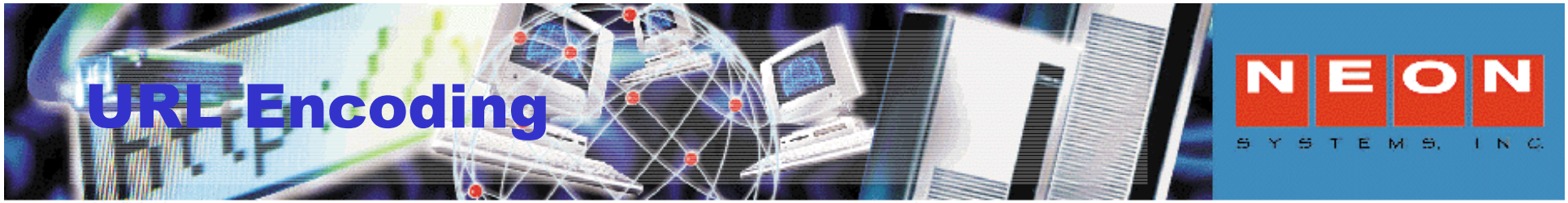
- This displays the incoming parameter list
- Illustrates input and output processing

```
/* REXX */
trace

"OUTPUT" "<html><title>A Simple CGI</title><body>"
"OUTPUT" "<h2>This is a TRIVIAL CGI</h2>"
"OUTPUT" "TRIVIAL CGI A called with parameters:<br>"
do forever
  "PEEKTO parm"
  if rc <> 0 then leave
  "OUTPUT" parm || "<br>"
  “READTO”
end
SaveRC = RC
"OUTPUT" "</body></html>"

exit SaveRC * (SaveRC<>12)
```



URLs and CGIs

- bob.com/calc.cgi?apple&yellow+pencil
  - "?" indicates start of parameter list
  - "&" is the parameter delimiter
  - "+" translates to a space
- .../enroll.cgi?name=santa&addr=north+pole
  - Commonly, "name=value" pairs are specified
- You have access to the entire URL within your CGI



URL Encoding

- Q: How can I use "Tom&Jerry" as a parm?
- A: Encode the "&" in ASCII: %26
  - Use the ASCII hex value of a character
  - See RFC1738 for details
  - "+" encodes to a space
- Browsers commonly do this for you
  - EWENCODE and EWDECODE Shadow utilities
- There are only a few "safe" characters
  - Alphamerics and $-_.,!*'( )

CGI Stream Definitions

- Input:
  - Stream 0: input parameters, already decoded
    - "Stdin" for other servers, not decoded
  - Stream 1: browser header records
    - Also available in CGI variables
  - Stream 2: your thread number
- Output:
  - Everything on output goes to the browser

Example of CGI Input

- company.com/doit.cgi?apple&blue+%232+pencil
- Note that Shadow has decoded the input for you

Stream 0:
apple
blue #2 pencil

Stream 1:
Connection: Keep-Alive
User-Agent: Mozilla/3.01 (Win95; I)
Pragma: no-cache
Host: jupiter.beyond-software.com:83
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*

Stream 2:
/EWEB.THREAD/2

CGI Environment Variables

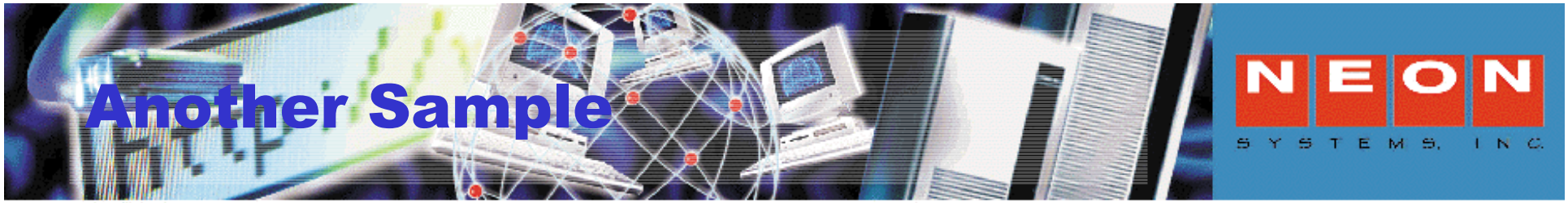
- CGI environment available for your use
  - List of variables defined by IETF
- Use GLOBALV command
  - GLOBALV GET PATH_INFO
- All headers also available
  - Authentication, Expiration, User-Agent, etc.

The CGI Environment

- The CGI specification defines these variables
- Retrieve only those you need
- CMS and REXX environments also available

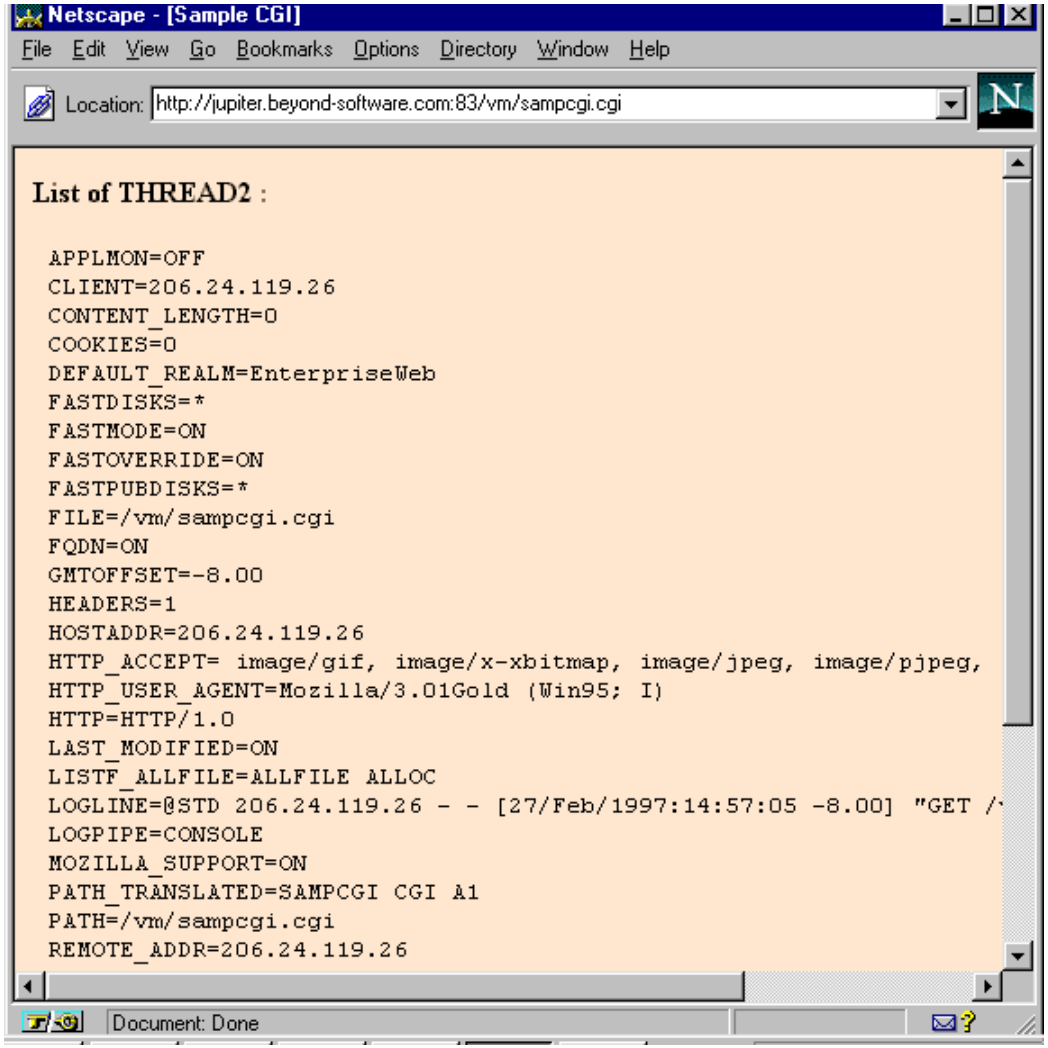
IETF variable name

AUTH_TYPE
CONTENT_LENGTH
CONTENT_TYPE
GATEWAY_INTERFACE
HTTP_*
PATH_INFO
PATH_TRANSLATED
QUERY_STRING
REMOTE_ADDR
REMOTE_HOST
REMOTE_IDENT
REMOTE_USER
REQUEST_METHOD
SCRIPT_NAME
SERVER_NAME
SERVER_PORT
SERVER_PROTOCOL
SERVER_SOFTWARE




```
/*
This CGI displays the environment variables
*/
trace

"OUTPUT" "<html><title>Sample
CGI</title><body>"

"Callpipe *.input.2: | varload"
if rc <> 0 then exit 0
Thread = eweb.thread

"Output <h4>List of THREAD"Thread":</h4>"
"Output <pre>"
"Callpipe" ,
   "CMS GLOBALV SELECT THREAD”thread "LIST |" ,
   "drop 1 |" ,
   "sort |" ,
   "*:"
"Output </pre></html>"
```

Output of SAMPCGI



An HTTP Conversation

Client:
GET /Shadow/index.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.08 [en] (Win98; U ;Nav)
Host: jupiter.neonsys.com:83
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*

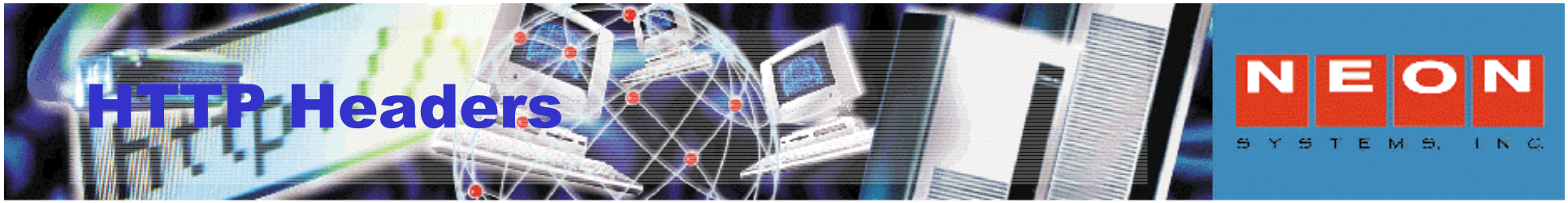
Server:
HTTP/1.0 200 OK
Content-Type: text/html
Server: Shadow_Web_Server_for_VM+SSL/1.1.5 VM_ESA/2.2.4.9903 CMS/15.903...
MIME-Version: 1.0

<html>
<!-- Neon Systems Inc. Shadow Web Server Index -->
<!-- Shadow Web Server 1.5 EWMENU: SHADOW FILELIST *-->
<head><title>Installation Verification Program </title></head>
<body>
<h2>Shadow Web Server for VM</h2>

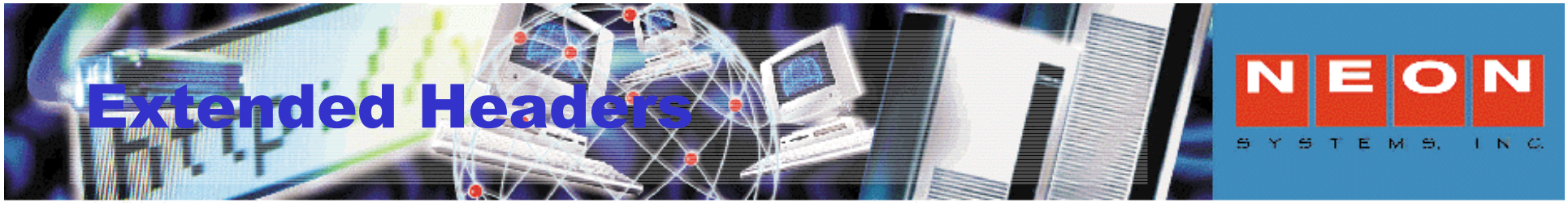
--- SNIP ---

</body></html>

- HTTP defines status codes for Web traffic
  - 100-199: Informational; unused
  - 200-299: Success 200 OK
  - 300-399: Redirection 302, 304
  - 400-499: Client error 401, 403, 404
  - 500-599: Server error 500
  - 600-???: Reserved for future use
- Your CGI can emit these



- Useful server headers (see spec for details)
  - Content-Length: works "thermometer"
    - New EWFSIZE Shadow utility
  - Content-Type: text/plain, image/gif, etc.
  - Date: Date/time of server response
  - Expires: Date/time when data is stale
  - Last-Modified: enables browser cache
  - Location: redirection



Extended Headers

- Headers that start with "X-" are extensions to the specification
- Shadow Web Server extensions:
  - x-eweb: Headers-begin
  - x-eweb: Headers-end
  - x-eweb: Headers-reset
  - x-eweb: Headers-reset-nodefault

This is in bold text

/* A CGI that emits headers */
"Output HTTP/1.0 200 OK"
"Output Content-type: text/plain"
"Output"
"Output This is in bold text"

This is in bold text

/* A CGI that emits headers */
"Output HTTP/1.0 200 OK"
"Output Content-type: text/html"
"Output"
"Output This is in bold text"

More Fun with Headers

NEON SYSTEMS, INC.

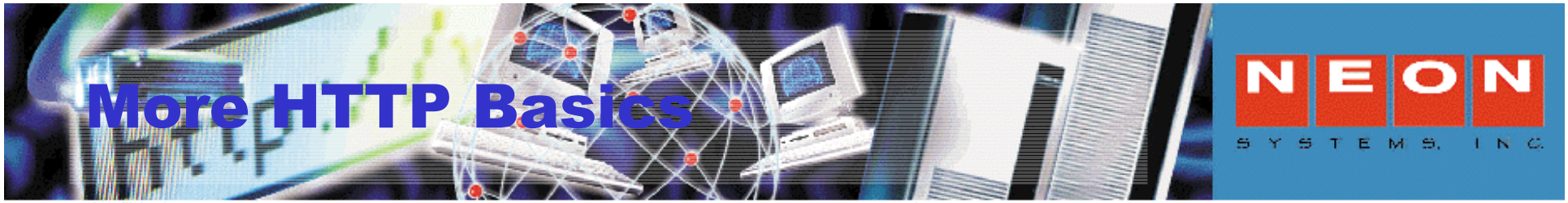
- Automatically launch MS-Excel
  - Use tab-delimited data
- This works with MS-Word and other applications, as well

```
'Output Content-Type: application/msexcel       - binary'
'Output Content-Disposition: inline;filename="mystuff.xls"'
'Output'

"Callpipe (name SendIt)" ,
   "stem TabData. |" ,
   "*.output.0:"
```

EWLayout Utility

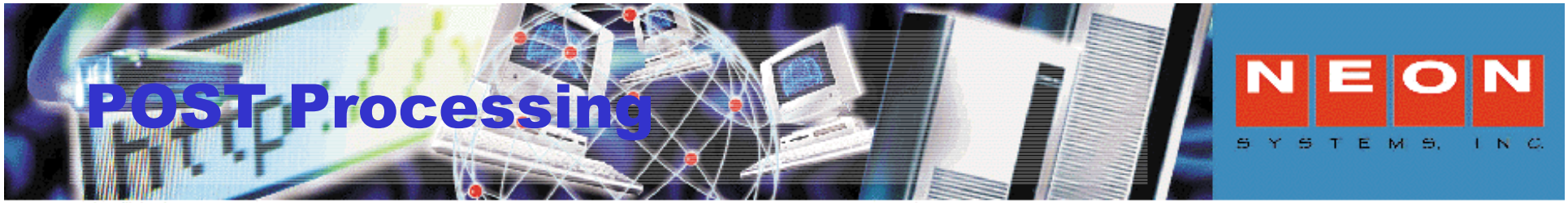
- New for Shadow 1.5; used to “table-ize” data
- Just the thing for adding tabs to SQL output
  - Note: Without the parameter list, EWLAYOUT will generate the <tr> and <td> HTML tags for a table

```
'Output Content-Type: application/msexcel      - binary'
'Output Content-Disposition: inline;filename="mystuff.xls"'
'Output'

'CallPipe',
      '  sqlselect * from employee',
      '| ewlayout Name ”05”x Address ”05”x SSN',
      '| *:'
```



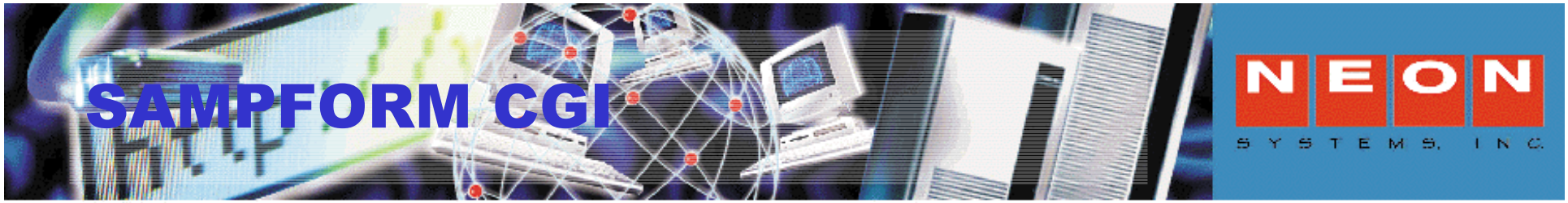
More HTTP Basics

- There are two basic HTTP verbs, which are requests of the client:
  - GET: requests a file from the server
  - POST: send data to the server
- POST must send data to a CGI; CGIs may also be invoked via GET
- All HTML pages, GIFs, etc are called via GET

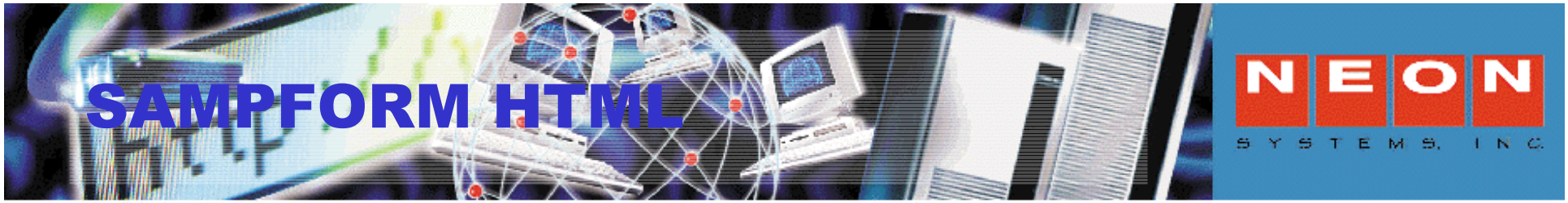


POST Processing

- Use HTML FORM tag to prompt for input
- User data is then POSTed to server
- You must parse for correctness
- Text, hidden, non-displayed fields available
- Drop lists, radio buttons, action buttons too
- EWGET and EWSET utilities provided for processing FORM data

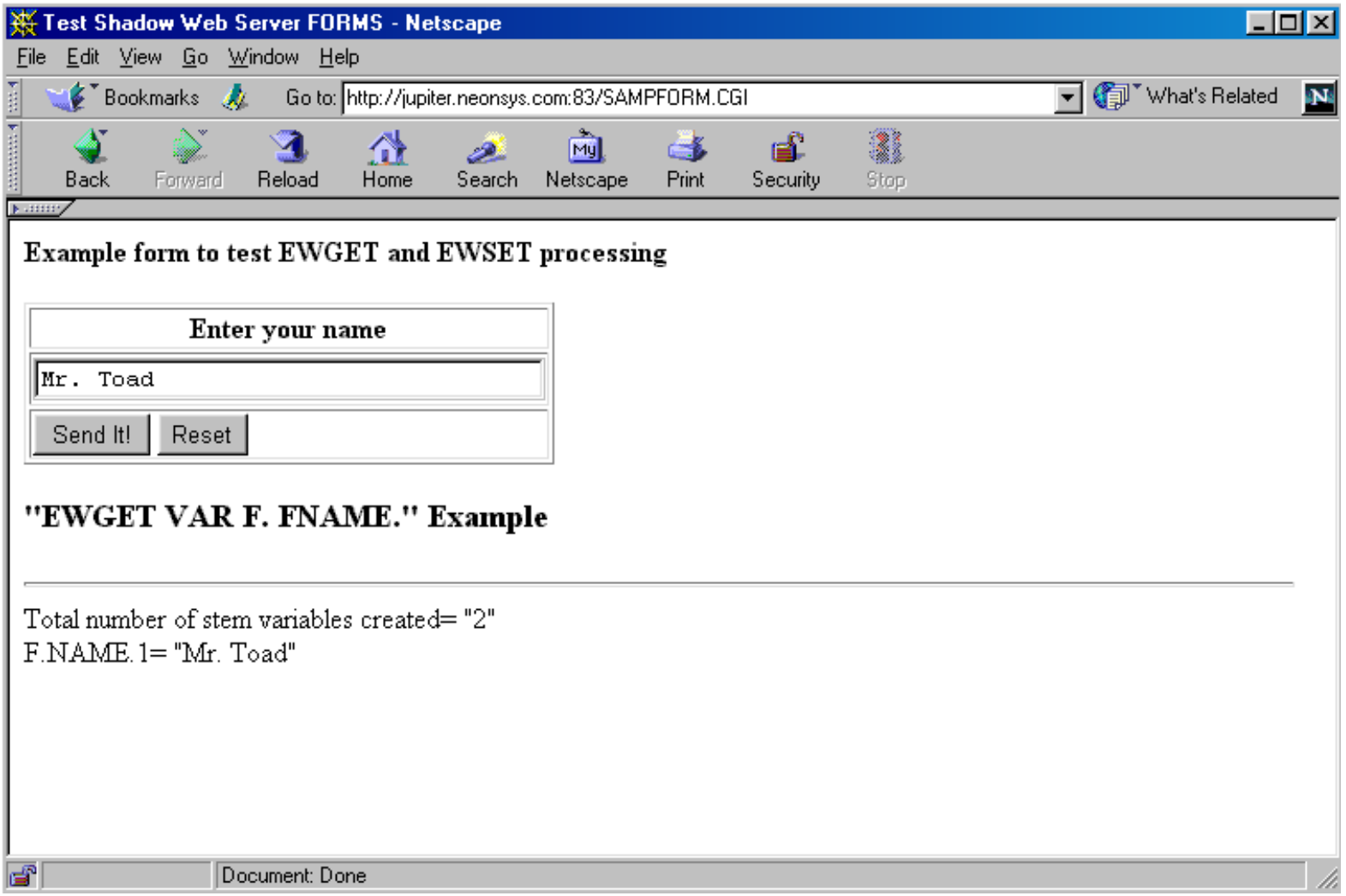


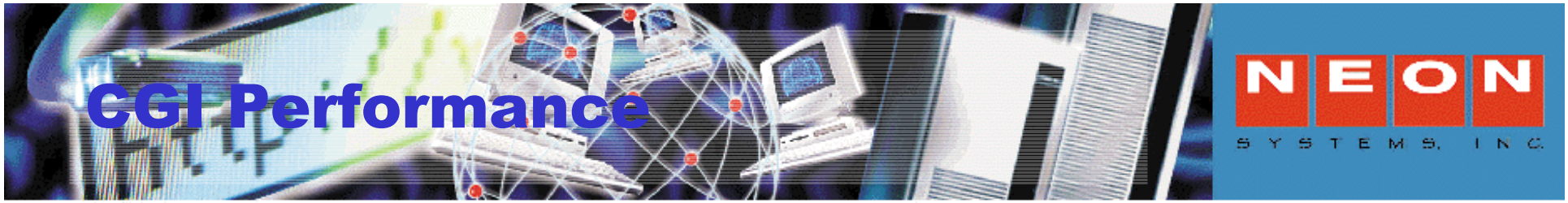

```
/* EWEB example of how retrieve HTML FORM variables. */

'callpipe *: | EWGET VAR F. FNAME.'

/* If no FORM stem variables set then we need to display HTML FORM */
if FNAME.0 = 0 then 'callpipe < SAMPFORM.HTML | *:'

else do
 'CallPipe (endchar ?)',    /* Update form */
    '< SAMPFORM HTML',      /* Form on Primary */
    '| a: ewset',           /* Apply Variables to Form */
    '| *:',                 /* Send out updated Form */
    '?',
    '  stem fname.',        /* Variables on Secondary */
    '| a:'                  /* Route to EWSet */

  output '<H3>"EWGET VAR F. FNAME." Example</h3><hr>'
  output 'Total number of stem variables created= "'FNAME.0'"<br>'
  output 'F.NAME.1= "'F.NAME.1'"' '<br>'
  output '</body></html>'
end

exit
```




```
<!-- Called by SAMPFORM CGI for EWSET/EWGET processing -->
<html><head><title>Test Shadow Web Server FORMS</title></head>
<body>
<h4>Example form to test EWGET and EWSET processing</h4>
<FORM  METHOD="POST" ACTION="SAMPFORM.CGI">
<TABLE BORDER>
<tr>
  <TH>Enter your name</TH>
</tr>
<tr>
  <td><INPUT Type="text"  Name="NAME" size="35"></td>
</tr>
<tr>
  <td><INPUT Type="submit" Name="ACTION" Value="Send It!">
      <INPUT Type="reset"></td>
</tr>
</TABLE>
</form>
</body>
</html>
```

Screen shot of SAMPFORM



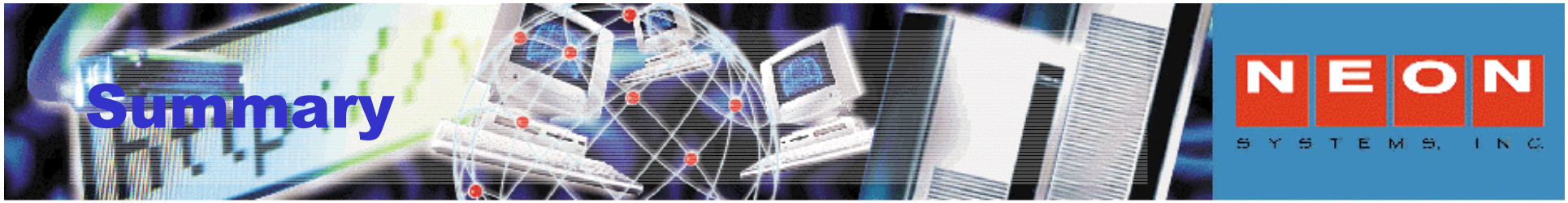


CGI Performance

- Compile your REXX, if possible
- pre-EXECLOAD your CGIs:
  - Add "CGIPROD" to mediamap w/PRELOADED directive
  - Name your production CGIs CGIPROD, and add them to $EWEB PRELOADS
- Mind your PIPELINE performance

Other "Cool" Things to Do

- Do your own password prompts, multipart files, etc.
- Examples ship with Shadow Web Server
  - Publishing system
    - Move files from client to server
  - Guest book
    - Form to "sign in"; sends to e-mail, log, etc.
  - Source included for all examples



- It's easy to do simple things with CGI
- With a little work, extremely complicated pages can be built.
- The REXX/Pipelines environment is a very efficient application development platform
- And, it's fun to play with this stuff!