

# AS/400 WEB CGI programming made easy

<http://www.easy400.ibm.it>

*Giovanni B. Perotti IBM Italy*



To run the CGI sample program:

<http://www.easy400.ibm.it/cgidevoit/opinion.mbr>

*To display speaker's notes:*

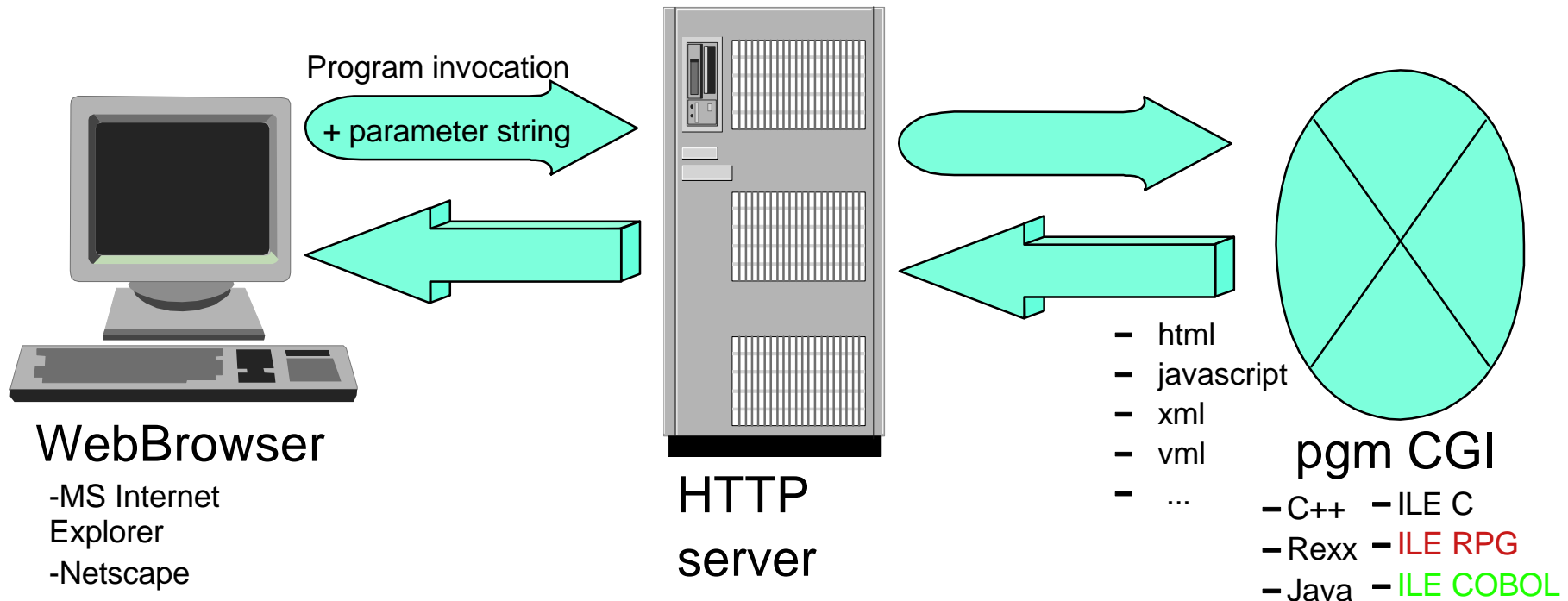
*Page-Open speaker note*

# What's CGI

## Common Gateway Interface

supported in all Internet platforms

- A **standard** of information exchange between the web browser and a server program
- **Transaction oriented** programs
  - standard input
  - standard output
  - One browser request
  - One program response



# Traditional 5250 interactive program

called from a previous program

```
CALL MYLIBLIB/MYPGM PARM('2')
```

housekeeping

PUT screen  
**WAIT**  
GET screen

?

End

process

*Interactive*

# Web CGI program

called from a previous html page

```
http://.../QSYS.LIB/MYLIB.LIB/  
mypgm.pgm?custno=2
```

GET input  
string

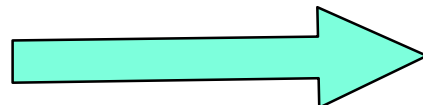
Housekeeping  
&  
process

PUT **html**

no **WAIT**

End

*Transaction oriented*



## CGI "weaknesses"

*Must die* after responding:  
HTTP cannot wait;  
must Seton LR and Return

- always starts from beginning
- memory is cleared

*non-persistent CGI*

## A great memory medicine

Adopt the *forgetful one* technique

- ★ send a memo to the client and ...
- ★ get it back on the next call

## Persistent CGI

- Returns with LR off
- Issues a *handle* to be called back

- always starts from beginning
- mandatory for COMMIT
- same performance
- requires non-persistent CGI programming skill

# INPUT to a CGI

It's a **string** sent from browser along with the *program invocation*

```
http://.../path/mypgm.pgm?cusno=000817&cusnam=Brown
```

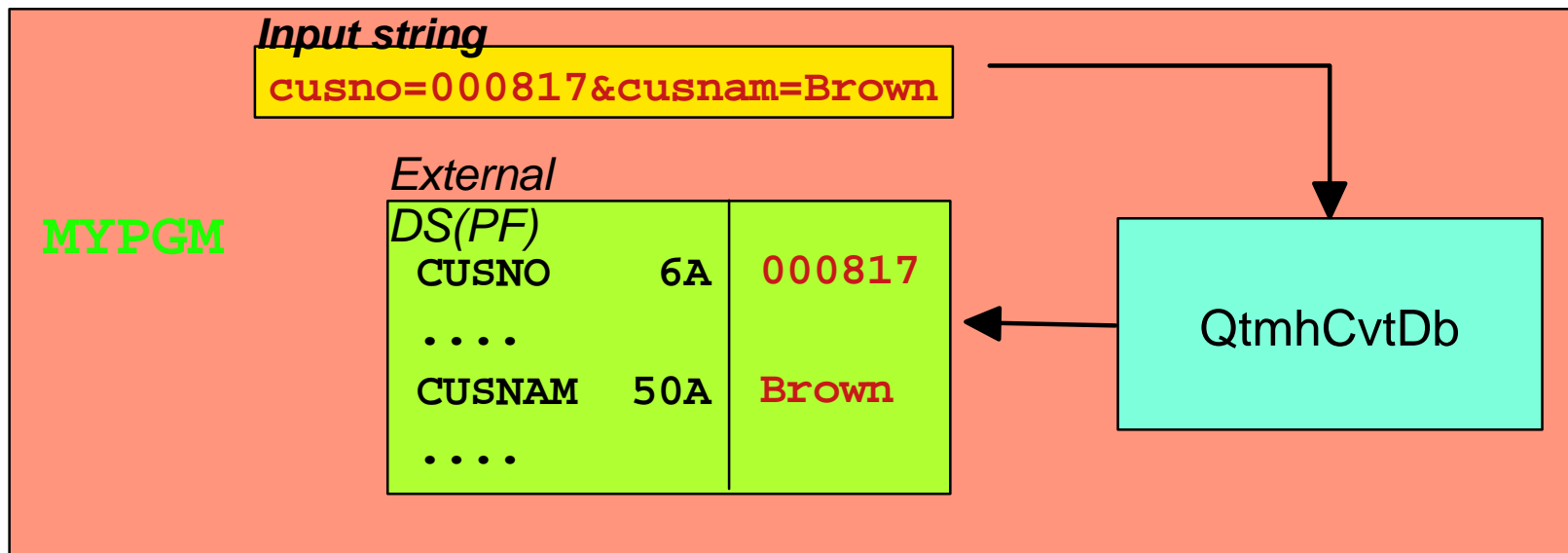
The string works like a parameter list:

the CGI must read it, understand it, and establish its process.

The CGI must;

1-read it thru the appropriate API, `QtmhRdStdIn` or `QtmhGetEnv`

2- *pars* it, i.e. break it into program variables; this may be done thru `QtmhCvtDb` API. This API splits the input into fields of an external DS



# Output from CGI

- It's a string, containing a script which the browser would understand, usually an HTML script
- The output buffer must be sent thru **API `qtmhWrStOut`**

```
Content-Type: text/html

<HTML>
<HEAD>
<TITLE>Order Entry</TITLE>
</HEAD>
<BODY><H1>Order confirmation</H1>
....
</BODY></HTML>
```

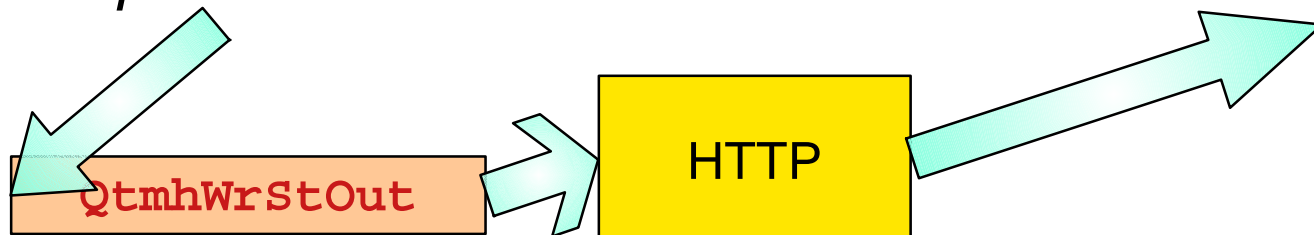
*Output buffer*

Order Entry

Order confirmation

....

*Browser would "mime"*



## The real problems of CGI ...

no externally defined WEB device file!

- ▶ No instructions (EXFMT) to perform I/O, just API's
- ▶ unformatted input string
- ▶ unformatted output string, to contain both text and variables!

Unability to separate presentation dialog from process logic results into cumbersome development, long testing, and high maintenance costs.

*Would you ever develop on AS/400 without externally defined display files?*

*Though development*

*Hell of a maintenance*

*Too expensive!*

## ... solved this way

an ILE RPG **Service Program** developed in IBM Rochester lab

- ✓ allows script (HTML, ...) external definitions
- ✓ supports record formats in externally defined scripts
- ✓ supports variables in record formats
- ✓ simplified interfaces to all HTTP API's

- ★ A script is just a text in a source file member
- ★ Even easier than DDS
- ★ No level check

# External HTML script

```
crtpf mylib/htmlsrc rcdlen(132)
```

```
/$header
```

```
Content-type: text/html
```

```
<HTML><BODY>
```

```
<HEAD><TITLE>External HTML  
script</TITLE></HEAD>
```

```
/$sections
```

<P>An HTML source member can be divided into up to 50 sections (record formats). Each section can be separately issued from the CGI in the proper sequence.

```
/$variables
```

<p>Each section may imbed up to 50 `/%variables%/`. Each variable can be loaded by the CGI with a value up to 500 char.

```
/$end
```

```
</BODY></HTML>
```

50

`/$sections`

in a member

50

`/%variables%/`

in a section

<http://www.easy400/cgi devo/opinion.mbr>

An opinion

I'd like to know your opinion about

  
 (1)

CGI program EXERCISE

- ☞ Read input string
- ☞ Load external HTML script
- Process request
- ☞ Assign values to */%variables%/* in record format */\$answer*
- ☞ Issue record format */\$answer*
- ☞ Send buffer to client  
LR & RETURN

Source file HTMLSRC, member EXERCISE

```
<HTML>
<h1>Just an
opinion</h1>
At /%time%/ my opinion
about
/%subject%/ is:<i>
/%this%/</i><br>
<u>Retry</u>
```

Just an opinion

At 8:56 my opinion about this coffee is: great

[Retry](#)

☞ = *funcions assisted by our service program*

# The functions of our SERVICE PROGRAM

## ★ Basic

1. Get input from client browser
2. Map input string into external data structure
3. Load external HTML script
4. Set HTML variables and issue script record formats (*sections*)

## ★ Often used

1. Override database files
2. Edit numeric field
3. Get environment variables
4. HTML trace & debug

## ★ Other

1. Retrieve page counter
2. Generate random integer
3. Execute CL commnad
4. Create or change environment variable
5. Handle multiple occurencies of input variable

<http://www.easy400.ibm.it>

the site for free downloads of CGI development tools and Web utilities

- ✿ cgi service program
- ✿ demos
- ✿ tutorials, labs, examples
- ✿ utilities
- ✿ packages

... and all the sources,  
so you can do the same  
and even better



Made by  
Giovanni B. Perotti  
IBM Italy