
Internet Software Technologies

CGI

IMCNE

A.A. 2008/09

Gabriele Cecchetti

Introduction

(1/2)

- Many important uses of Web require interaction between the individual user and the server, to customize the results (e.g. search a database)
- Also, it's often desirable to customize the response from the Web server.

- Web servers can do all these things and more by using auxiliary programs that run on the server platform and allow the server to generate customized responses
- These programs are called **gateways** or **scripts**
- Scripts may connect the Web or to other services and may collect information from the user through interactive forms.

- Basic idea: Some documents are really programs!
- When a client browser request the URL of something that is actually a program, the server executes the requested program and return the results (not the program itself!) to the client.
- The result should appear to be an HTML document so the browser and the user may never realize that the program was executed

- These programs – scripts – can do many things:
 1. Scripts are used to access information from non Web sources (as db)
 2. Scripts allow interaction between the user and the server
 3. Scripts can construct custom documents dynamically at the time they are requested

- Interaction between the user and the server is usually built by using special type of HTML document that contains a fill-in form (`<FORM>`)
- Input collected by forms is sent to Web server that route this information to Web Scripts which often acts as a gateway between Web and another online service, translating Web language (HTTP) to service language
- Output received by these services is adapted for Web language and sent to user's browser

What is a Script ?

- **A Web script is a program that can be executed by the Web server in response to Web requests**
- Any program can be a Web Script, even if is written using any language!

7

Basic Goal of a Script

- It's to communicate with a browser (2 way)
 - Web server will act as a conduit to pass information between browser and the script
 - *Script does not see httpd daemon!* Even if, it receives input from httpd daemon and send output within this enviroment, it's thinking to communicate to browser

8

(Typical) Role of httpd daemon

1. The daemon must determine that the request is really supposed to be a program rather than a document
2. The daemon must locate the program and determine if it is permissible to execute it
3. The daemon must start the script program and ensure that the input from the client will be passed to the script
4. The daemon must read the output from the script and pass it back to the client
5. The daemon must send an error message back to the client if something goes wrong with the script program. The daemon must also close the network connection properly when the script completes

9

What files are executable scripts ?

- The execution of scripts is always controlled by Web server, non the browser
- The server software will execute scripts only according to specific rules; usually:
- Scripts have to be located in a particular directory and/or having a particular extension (.cgi)
- Scripts have to be system execute permission set.

10

Making the script run: problem...

- HTTPD server starts the script program and passes along the data. Making sure the server and the script will work correctly together can be a problem, though, since they are usually written by complete different programmers...
- This problem is solved by..... **CGI !**

11

Making the script run: the Common Gateway Interface (CGI)

- CGI is a standard for how scripts are to be called and how the data is passed between HTTPD server and the script
- The CGI perform the same role for the server and the script as HTTP does for the server and the client: as long as both the httpd program and the script program follow the CGI rules, everything will work.

12

CGI-scripts

- Web scripts are often called CGI-scripts
- The script directory is often called `cgi-bin`

13

Scripts depend on Web server system

- A scripts that works well on one type of type of Web server system my not work on another type of server system.
- The details of CGI rules depend on the type of operating systems the server run on. Example
 - Unix Systems: scripts executes as a process and it receives input from `stdin` and env. var. and the results are passed back by `stdout`
 - Windows NT/XP/... : scripts executes ad an application and the data is passed through temporary files

14

Alternative Interfaces to CGI

- ISAPI
- NSAPI
- OLE
- ...
- All script interfaces do the same thing, though: they define how the script will be started and how data will be passed to and from a script

15

A user's view of a script

(1/2)

- Consider an example: “to read system uptime”
 - Client would request
`GET /scripts/uptime_script HTTP/1.0`
 - httpd server would execute `uptime_script` and return the result to the client:
`1:29pm up 21 days, 4:35, 5 users, load
average: 0.00, 0.09, 0.00`

16

- The `uptime_script` does 2 things:
 1. It prints information describing information to come (the meta-information)
 2. It prints the results of running program `/usr/bin/uptime`

1) Meta-information of a scripts

- The essential meta-information is the `Content-type` of the document returned (it may be: `text/plain`, or `text/html`, or any other MIME type supported by the server)
- Other meta-information are HTTP directives to be added to the normal HTTP directives produced by server response

2) script_uptime

```
#!/bin/sh
echo "Content-type: text/plain"
echo
if [ -x /usr/bin/uptime ]; then
    /usr/bin/uptime
else
    echo "Cannot find uptime command on
    this system"
    exit 1
fi
```

19

Examination of the script

- Each time the script is used, 3 programs will run on the server: **httpd**, **/bin/sh**, **/usr/bin/uptime**
- The meta-information is separated by results by a blank line (**\n**)

20

1. **HTTPD program waits for a new request** to arrive from a client somewhere on the net

2. **A request arrives from a client**

For example because a user has activated an anchor inside an HTML document:

```
<A HREF="http://myserver./scripts/uptime_script">
```

next, the browser has set-up a 2 way connection towards myserver.retis and sends http request

```
GET /scripts/uptime_scripts HTTP/1.0
```

- 3. The server parses the request:** httpd decode the request according to the HTTP protocol to determine what it should do;
 - Then it checks its configuration and discovers that `/scripts/uptime_script` is supposed to be executed as script
 - The now knows also that it have to use version HTTP 1.0 to return the response to client
 - The results are to be sent back over the connection that the request came from, so there is no need for the server to find the client on the net

- 4. Read other information from Network.** HTTPD now reads the rest of requests, as needed (ex. browser description and capabilities)

```
User-agent: Mozilla for Windows 5.0
Accept: text/plain
Accept: text/html
Accept: */*
```

5. Do the method requested. If no error has been detected, httpd server fulfills the request.

- ❑ Locate the script and confirm that it is executable
- ❑ The server must create the correct execution environment (env. var., `stdin`, `stdout`)
- ❑ The script is executed
- ❑ When script is finished the httpd program sends the response to the client

If Error has been detected, status code is $\neq 200$

6. Close File; Close Network. When the output of the script is completely sent the script terminates and closes all connections to the httpd program (request is completed)

7. Go to step 1.

- ❑ The HTTPD server is now ready for another request from somewhere on the network.
- ❑ The server repeats these steps over and over

Costs of using scripts: many programs running on server

- Scripts require the Web server to do a lot more work than just retrieve a document
- Running scripts means that at least twice as many programs must run on the server, and if the script does any significant computation, execute other programs, or communicates with other server, this can add up to a very large burden on the server
- If the server process more than one request at a time, it might run many scripts at the same time magnifying the problems

Costs of using scripts: NPH-scripts

- HTTPD server not only relays the information from the script to the client, it checks to make sure the proper http headers are sent
- Scanning the result to see if the headers are present or not slows the relay of the data
- For this reason some servers allow NPH-scripts (“No Parse Headers”)

29

Interpreted scripts vs. executable scripts

- Interpreted scripts (using Shell, Perl, ...) need an external interpreter to be executed: this is an additional cost (more memory, slower...)

30

When problems occur

- At best, the user will get an error message
- At worst, the whole web server could hang or crash
- Scripts must end (successfully or not) in a reasonable time: because user goes away and they may waste resources

Scripts and forms

- Scripts may use special HTML documents called forms to gather information from the user.

- Web forms are a special kind of HTML document defined by the rules of HTML.
- Actually they are regular Web documents that have place for the user to respond.

- HTML form contains:
 - The **METHOD** by which the user input is to be sent
 - The **ACTION** which specifies a URL to which the user input is to be sent (a script ?!)
 - A **SUBMIT** button, to send the completed form

- User input may be obtained by one or more of the following items:
 - **Input text boxes** for the user to type in
 - **Checkboxes** for the user to select
 - **Radio buttons** for the user to select
 - Range for the user to **select** along

How arguments are passed to Web scripts: GET method (1/2)

- Browser constructs GET request from the action specified:

GET

```
http://myserver/scripts/something?name  
1=value1&name2=value2 HTTP/1.0
```

How arguments are passed to Web scripts: GET method (2/2)

- Server sets `REQUEST_METHOD="GET"`
- Server copies what follows '?' in environment variable `QUERY_STRING`
- Script has to decode this variable to use the arguments passed

37

How arguments are passed to Web scripts: POST method (1/2)

- Browser constructs POST request from the action specified:

`POST http://myserver/scripts/something
HTTP/1.0`

- The request's body contains the parameters to be passed:
`name1=value1&name2=value2`

38

How arguments are passed to Web scripts: POST method (2/2)

- Server sets `REQUEST_METHOD="POST"`
- Server sets `CONTENT_LENGTH=length_of_body_request`
- Server puts the body request on `stdin` of script
- Script has to decode `stdin` until `CONTENT_LENGTH` characters to use the arguments passed

39

Cost of using Forms

- Using forms requires the web server to do more work because the browser must contact the server twice

40