Why Dynamic Documents are needed?
- There are many situations when customization of the presented document is needed. Such customization cannot be performed via regular web pages which are static.
- A program is generally needed to customize the documents based on user need expressed by user interest and interaction (such as Daily News).
- Also, there were many other network services available before the Web, which were interactive, and needs to be on the Web (such as online Library Catalog).

What is a Script?
- Scripting is a mechanism by which a web client can request a document from the server to be prepared and served by a third-party program instead of the web server.
- Web Script is this third party program.
- Scripts allow:
  - Access to information from non-web source.
  - Interaction between the user and the server.
  - Script can create custom documents.

What is the Role of HTTP Server?
- The HTTP server acts as a mediator between the client and the script program.
- When a request arrives:
  - Determine that the request is for a program.
  - Locate the program and check access permissions.
  - Start the script.
  - Ensure input from the client reaches to script.
  - Wait for the script to complete and read its output and pass it back to the client.
  - In case of error, notify the client.
  - Close the network connection properly.

What Files are Executables?
- Server, not the browser has the control.
- The server system administrator sets the policy.
- It varies from system to system.
  - In NASA Impd for UNIX, the scripts are always in a special directory (such as /cgi-bin/) and must have an UNIX execution permission on.
  - Sometimes they must have a special extension (such as *.cgi).
  - CERN Impd has a set of rules.
  - HTTPS for Windows NT can execute it from any directory but must have a .exe extension.
Making The Script Run: CGI

- The proper communication between an HTTP server and the Script is ensured by a special protocol called Common Gateway Protocol (CGI).
- As long as both the Server and the Script adhere to these rules proper communication is ensured.
- The actual mechanism for interaction varies from system to system.
  - UNIX uses stdin/stdout and environment variables.
  - MacOS passes data through Apple Events.
  - Windows NT uses a set of temporary files.
- CGI is a collection of "standards"

An Example

- A User wants to know how busy is the UNIX server via Web.
- Unix program `uptime()` can provide the answer.
- We will write a gateway script which will connect the server with `uptime`.

Demonstration

Run the How “busy” Script

See HTML Client Source

See The Source Code of “busy” Script

Server Parses the Request

- Step 3: Server parses the request
  - Server parses the information to decide what to do. It checks the object location and finds out if it is a script!

- Step 4: Server reads other information
  - Now server reads rest of the request. For example:
    - User-agents: Mosaic for X Windows/2.4
    - Accept: text/plain
    - Accept: text/html
    - Accept */*

Some body Clicks..

- Step-1 The httpd program waits for request
- some body clicks
- HTML form:
  - `<A HREF ="http://cgi.mcs.kent.edu/cgi-bin/javed/busy">Show current load on cgi.mcs.kent.edu</A>`
  - The client sends HTTP message to server:
    - GET /cgi-bin/javed/busy HTTP/1.0

Step 5: Do the method requested.

- Step 5: Server does the method requested.
  - It prepares the Unix Environment
  - Sets the appropriate ENV variables
  - Starts a copy of Script
  - Sets STDIN/STDOUT

- Scripts Runs:
  - It reads the ENV variables, if needed. Starts the `uptime`. All output is directed to STDOUT, which is received by HTTPD.

Client can also use POST.
Send the Result

- If Everything goes well:
  - HTTP/1.0 200 Document follows
  - Server: NCSA /1.4
  - Date: Thu, 20 Jul 1998 17:35:00 GMT
  - (then appends what Script has sent)
- Script Appends to it:
  - Content-type: text/plain
  - 1:15am up 7 days
- The response to user looks like this:
  - 1:29pm up 21 days, 4:35, 5 users, load average:0.0,0.09 0.00

If Anything Goes Wrong

- However, if anything goes wrong Server may send:
  - HTTP/1.0 403 Document Not Found
  - Server: NCSA /1.4
  - Date: Thu, 20 Jul 1998 17:35:00 GMT
  - Content-type: text/html
  - Content-length:0
- There are also other ways:
  - Script can append status code and Server notifies Client:
    - HTTP/1.0 500 Server Error
    - Server: NCSA /1.4
    - Date: Thu, 20 Jul 1998 17:35:00 GMT
    - Content-type: text/html
    - Content-length:0
  - Sometimes Script can append its own error code (like the example)
  - Unexpected error message from script can baffle, Server, Browser or the User.

Cost of Using Scripts

- Script Requires HTTP server to do lot more work.
- Many concurrent request can mean serious trouble.
- Generally every Server parses the Script output to check if it correct. This is also expensive.
- How many programs run in this example?
- How many programs run if Scripts are written in C?

Gathering Information From Client

CGI with Forms

Example

- Vicinity Corporation has an Address Locator Database which can find any address in USA.
- Pizza Hut corporation wants to provide its customers an online tool through which any customer can locate the nearest Pizza Hut.
- The locator requires the address of the customer to find nearest Pizza Hut.
- We show a Form based system, where the user can submit his address to Pizza Hut's website. A CGI script redirects the information to Vicinity Corps database and finally the result comes back.

Demonstration

PizzaHut Locator
Technology Behind - HTML Form

- Web Forms are a special kind of HTML documents defined by the rules of HTML.
- They are almost regular pages, except that they have special fields where users can respond.
- Web Browsers must know how to interpret forms.

Special Fields of HTML Form

- The METHOD
  - by which user input has to be sent (must)
- The ACTION
  - which specifies the URL (script) to which data will be sent.
- A set of INPUT objects
  - Through which user types the data.
- A SUBMIT button
  - to initiate the sending.

Some Input Objects

- Text Boxes
  - `<INPUT TYPE="text" NAME="address" SIZE=40 VALUE="default">`
- Check boxes
  - `<INPUT TYPE="checkbox" NAME="valid" VALUE= cream>`
- Radio Buttons
  - `<INPUT TYPE="radio" NAME="valid" VALUE= "yes">`
- Option Selection Boxes
- Reset Button
- Text Area
- Password Box, etc.

How It all works - 1

Step-1 User clicks the forms "submit" button

Step-2 Browser collects all the inputs and creates a message for CGI script in the form of one long string:

```
GMX=1&FAM=pizzahut&AD2=summit+street&AD3=kent,ohio
```

Source Code of PizzaHut Form

Click Here
How It all works -2

- Step-3: Web Browser Invokes the GET HTTP method.
  ```
  GET http://www.vicinity.com/yt.hm/
  ```
- Step-4: HTTP Server receives the method invocation via a socket connection.
- Step-5: HTTP Server parses the GET URL and realizes that it is a script, with data.

GET it or POST it?

- **GET**
  -Retrieve the specified URL
  -Data is appended with Script URL with “?” mark inbetween.
- **POST**
  -Send this data to specified URL
  -Data/file is carried back to the server.
  -A better method if large data have to be carried.

How It all works -3

- Step-6: HTTP server sets up env variables (UNIX)
  -server_name, request_method, path_info, script_name, query_string, content_type, content_length, etc.
- Step-7: HTTP server starts the CGI program.
- Step-8: CGI program reads the environment variables and discovers that it is responding to GET or POST.
- Step-9: IF GET, the data is in query_string. IF POST, CGI program receives the actual message body via STDIN pipe. (it knows content_length!)
  
  Cheer up, we are half way there!

How It all works -4

- Step-10: CGI scripts reads the arguments, and when necessary reformat them in the fashion appropriate for the locator database. Invokes it. Receives the input.
  
  Makes and HTML document.
- Step-11: CGI program returns the output results via STDOUT to HTTP server.
- Step-12: HTTP server receives the results on STDIN and concludes CGI interaction.
- Step-13: HTTP server returns the result to the Web Browser.

Cost of Using Forms & CGI

- 2 contacts are necessary.
- Communication is stateless thus duplicate information have to be sent.
- Parsing needed.
- It is a good idea to run Scripts on a different machine.

CGI and STATE

- HTTP server is stateless.
- But CGI can maintain state, remember which Browser is contacting
- How?
Image MAPS

- Components:
  - An Image.
  - A MAP file associating URLs with image regions.
  - HREF Anchor in the Main document with ISMAP tag.

- Steps:
  - Browser captures the location of the mouse and sends anchor:
    - http://www.server.org/scripts/imagemap/firstfloor
    - http://www.server.org/scripts/imagemap/firstfloor
  - Web Server runs the Imagemap program, which looks into the image map and finds out the right URL and sends it HTTP server.
  - If it is local Web Server returns page otherwise returns the URL with code 302 (Moved temporarily), which makes Browser to automatically contact the new server.

Server Side Include

- Can create a web server signature, customized with current time, creation date etc. meta information.
- Example:
  - Creation Date: &lt;--&lt;#echo var="LAST_MODIFIED"--&gt;
  - will generate output:
  - Creation Date Thu 30 June 1998 22:00:00 GMT
- Very Expensive. Why?

Netscape Push-Pull Technology

- Server-Push:
  - It is based on MIME type multipart/mixed. The first part of the document is sent to Browser as a part of multipart document. Connection is kept open. As each new part is sent Browser replaces the last one with it.

- Client-Pull:
  - A Special tag inside the document tells Client to refresh the document at a specified interval to get a updated version.

- Unique to Netscape only.