# CPET 499/ITC 250 Web Systems

November 20, 2014

- CGI (Common Gateway Interface)
  - W3C CGI, http://www.w3.org/CGI/
  - CGI 1.1, 2004, http://www.ietf.org/rfc/rfc3875
- HTTP (HyperText Transfer Protocol) Protocols
  - HTTP/1.1, 1999, http://www.w3.org/Protocols/rfc2616/rfc2616.html
  - HTTP/1.1 RFC 2817: Upgrading to TLS within HTTP/1.1 (Transport Layer Security), May 2000, https://www.ietf.org/rfc/rfc2817.txt
  - HTTP/1.1 RFC 7230: Message Syntax and Routing, June 2014, https://tools.ietf.org/html/rfc7230
  - HTTP/1.1 RFC 7231: Semantics and Content, June 2014, https://tools.ietf.org/html/rfc7231
  - HTTP/1.1 RFC 7232: Conditional Requests, June 2014, https://tools.ietf.org/html/rfc7232
  - HTTP/1.1 RFC 7233: Range Requests, June 2014, https://tools.ietf.org/html/rfc7233
  - HTTP/1.1 RFC 7234: Caching, June 2014, https://tools.ietf.org/html/rfc7234
  - HTTP/1.1 RFC 7235: Authentication, June 2014, https://tools.ietf.org/html/rfc7235
  - HTTP/1.1 RFC 7236: Authentication Scheme Registrations, June 2014, https://tools.ietf.org/html/rfc7236
  - HTTP/1.1 RFC 7237: Method Registrations, June 2014, https://tools.ietf.org/html/rfc7237
- Server Configuration

**Web Servers**

- Web (HTTP and HTTPS) server
  - Answer requests from browsers (the client program)
  - Retrieve the specific files (or execute a CGI script)
  - Return the document or script results
  - Communicate with client via the HTTP protocol

## CGI (Common Gateway Interface)

- W3C CGI, http://www.w3.org/CGI/
- CGI 1.1, 2004, http://www.ietf.org/rfc/rfc3875
  - 1.1. Purpose

    The Common Gateway Interface (CGI) [22] allows an HTTP [1], [4] server and a CGI script to share responsibility for responding to client requests. The client request comprises a Uniform Resource Identifier (URI) [11], a request method and various ancillary information about the request provided by the transport protocol.

    The CGI defines the abstract parameters, known as meta-variables, which describe a client's request. Together with a concrete programmer interface this specifies a platform-independent interface between the script and the HTTP server.

The server is responsible for managing connection, data transfer, transport and network issues related to the client request, whereas the CGI script handles the application issues, such as data access and document processing.

- The Common Gateway Interface (CGI) is a standard for interfacing Web applications with information servers such as HTTP or Web servers in a platform-independent manner.
- Tasks performed by CGI scripts (programs)
  - Query database
  - Perform calculations
  - Solicit and interpret user-supplied data
  - Retrieve requested information
  - Produced customized content
- A CGI program is an executable program that resided in a special directory such as /cgi-bin. It can be written in any language: C/C++, Fortran, Perl, TCL, any UNIX shell, Visual Basic, and AppleScript.

## CGI Environment Variables

**Perl**, https://www.perl.org/

- Perl CGI, http://perldoc.perl.org/CGI.html
  - http://perldoc.perl.org/Env.html
  - %ENV hash/associated array (key/value pair)
  - #! /usr/bin/perl OR
  - $path = $ENV{'PATH'};
  - $pwd = $ENV{'PWD'};
  - $userName = %ENV{'LOGNAME'}

```
#!"C:\xampp\perl\bin\perl.exe"
##
##  printenv -- demo CGI program which just prints its environment
##

print "Content-type: text/plain; charset=iso-8859-1\n\n";
foreach $var (sort(keys(%ENV))) {
        $val = $ENV{$var};
        $val =~ s|\n|\\n|g;
        $val =~ s|"|\\"|g;
        print "${var}=\"${val}\"\n";
}
```

**PHP**, http://php.net/manual/en/reserved.variables.environment.php

- Superglobal variables
  - $_GET
  - $_POST
  - $_ENV
  - $_SERVER, http://php.net/manual/en/reserved.variables.server.php
- Example 1: $_ENV
  <?php echo 'My username is ' . $_ENV["USER"] ?>
- Example 2:
  - $_ENV["HOSTNAME"]

        - $_ENV[COMPUTER NAME”]
    - Example 3: getenv()

```
<?php
$ip = getenv(“REMOTE_ADDRE”);
$ip2 = $_SERVER[‘REMOTE_ADDRE’];
?>
```

A list of environment variable defined by CGI standard is as shown below:

| **Variables** | **Purpose** |
|---|---|
| AUTH_TYPE | If the server supports user authentication, and the script is protects, this is the protocol-specific authentication method used to validate the user. |
| CONTENT_TYPE | It specifies the media type of the data for queries, which have attached information, such as HTTP POST and PUT, this is the content type of the data. |
| CONTENT_LENGTH | The length (number of bytes) of information passed to the script. |
| GATEWAY_INTERFACE | The name and version of the protocol being used by the server to communicate with the script. Format: CGI/revision |
| PATH_INFO | It provides any extra path information, as given in the URL, for accessing this script. The extra information is sent as PATH_INFO to be decoded by the server before it is passed to the CGI script. |
| PATH_TRANSLATED | It gives the absolute file system path for access the script. The server provides a translated version of PATH_INFO, which takes the path and does any virtual-to-physical mapping to it. |
| QUERY_STRING | Any additional information passed to the script after the ? mark in the URL which referenced this script is called the query information. It should not be decoded in any fashion. |
| REMOTE_HOST | It contains a fully qualified domain name of the client computer. If the host name cannot be determined, it should set REMOTE_ADDR to hold the IP address of the host and leave this variable unset. |
| REMOTE_ADDR | The IP address of the remote client computer making the request. |
| REMOTE_IDENT | The client machine's username. Usage of this variable should be limited to logging only. |
| SCRIPT_NAME | A virtual path to the script being executed, used for self-referencing URLs. |
| REMOTE_USER | The name used to authenticate the user for accessing the script. |
| SERVER_SOFTWARE | The name and version of the information server software answering the request (and running the gateway).<br># Format: name/version |
| SERVER_NAME | The server's hostname, DNS alias, or IP address as it would appear in self-referencing URLs. |
| SERVER_PROTOCOL | The name and revision of the information protocol this request came in with. Format: protocol/revision. |
| SERVER_PORT | The port number to which the request was sent. |
| REQUEST_METHOD | The method with which the request was made. For HTTP, this is "GET", "HEAD", "POST", etc. |

| | |
|---|---|
| HTTP_ACCEPT | Gives a comma-separated list of MIME types that the client can accept. |
| HTTP_REFERER | Provides the address of the page where the request originated. |
| HTTP_USER_AGENT | Specifies the name of the client program used to make the request. |

**HTTP Status Codes and Related Implementation**

- HTTP/1.1 – RFC 2616 Status Codes, http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
- HyperText Transfer Protocol (HTTP) Status Code Register, Last Updated – 2014-08-27, http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml
- REST HTTP Status Code, http://www.restapitutorial.com/httpstatuscodes.html
- HTTP Status Code (Windows), http://msdn.microsoft.com/en-us/library/windows/desktop/aa384325(v=vs.85).aspx
- The HTTP status code in IIS 7.0, IIS 7.5, and IIS 8.0, http://support.microsoft.com/kb/943891
- MDN HTTP Response codes, https://developer.mozilla.org/en-US/docs/Web/HTTP/Response_codes
- Twitter HTTP Status Codes, https://dev.twitter.com/overview/api/response-codes
- Android HttpStatus, http://developer.android.com/reference/org/apache/http/HttpStatus.html
- FacebookRequestError, https://developers.facebook.com/docs/reference/android/3.5/class/FacebookRequestError/
- Java Class HttpStatus, https://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/HttpStatus.html
- IBM WebSphere HTTP Status Code

**HTTP/1.1 – RFC 2616 Status Codes, http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html**

**1xx - Informational**

**2xx – Successful**

- 200 OK
- 201 Created
- 202 Accepted
- 203 Non-Authoritative Information
- 204 No Content
- 205 Reset Result
- 206 Partial Content

**3xx – Redirection**

- 300 Multiple Choices
- 301 Move Permanently
- 302 Found
- 303 See Other
- 304 Not Modified

- 305 Use Proxy
- 306 (Unused)
- 307 Temporary Redirect

**Client Errors Code 4xx**

- 400 Bad Request
- 401 Unauthorized
- 402 Payment Required (reserved)
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 407 Proxy Authentication Required
- 408 Request Timeout
- 409 Conflict
- 410 Gone
- 411 Length Required
- 412 Precondition Failed
- 413 Request Entity Too Large
- 414 Request-URI Too Long
- 415 Unsupported Media Type
- 416 Requested Range Not Satisfiable
- 417 Expectation Failed

**Server Error (5xx)**

- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout
- 505 HTTP Version Not Supported
- 506 Redirection failed

## Other References


- Apache HTTP Server Project, http://httpd.apache.org/
- Apache Webmaster Tools & Utilities, https://www.apachelounge.com/viewforum.php?f=8
- Chapter 26, Apache HTTP Server Configuration, https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/System_Administration_Guide/ch-httpdconfig.html
- Understanding Setup in IIS 7, http://www.iis.net/learn/install/installing-iis-7/understanding-setup-in-iis
- Getting Started with the IIS Manager in IIS 7 and IIS 8, http://www.iis.net/learn/get-started/getting-started-with-iis/getting-started-with-the-iis-manager-in-iis-7-and-iis-8

- IBM HTTP Server, http://httpd.apache.org/
- Oracle HTTP Server, https://docs.oracle.com/cd/E21764_01/web.1111/e10144/intro_ohs.htm#HSADM101

Web Services and Cloud Computing

- Standards and Web Services, http://www.ibm.com/developerworks/webservices/standards/
- Amazon Web Services, http://aws.amazon.com/
    - AWS EC2 (Elastic Compute Cloud), Amazon S3 (Simple Storage Service), Commercial RDBMS