Course Outline

Writing z/OS CGIs in Assembler

Course Summary

Objectives

By the end of this course, students will be able to:

- Code, assemble, bind, debug, deploy, and maintain CGIs for the z/OS environment, written in Assembler language
- Handle GET and POST requests: analyze and take action, as appropriate
- Parse and decode a QUERY_STRING value for GET
- Gather in the stdin data for POST
- Save a file as is or translated to EBCDIC on the mainframe, for POST
- Produce responses that are dynamically created HTML pages or redirection to existing pages
- Access environment variables
- Access DB2 data (optional: depends if DB2 installed and lab set up done)
- Access VSAM KSDS data by primary key or alternate index
- Put out HTML encoded in UTF-16, to provide a truly international aspect to your website
- Submit jobs to the batch from a CGI (optional; may not be appropriate in all environments).

Topics

- General Program Structure and Techniques
- Basic Processing
- Handling GET Requests
- The Data Connection - Part I: The Story
- The Data Connection - Part III: Working With DB2 Data
- Hidden Controls and cookies
- POST Requests
- Handling Files Sent by POST
- Working With Unicode Data
- Submitting jobs from a CGI
- Wrap up

Prerequisite

Experience in coding / maintaining Assembler programs; current knowledge of Assembler, such as might be obtained from attending course C500: "z/OS Assembler for Applications Programmers". Familiarity with z/OS UNIX such as might be obtained from attending U510: "Introduction to z/OS UNIX" and U520: "Developing Applications for z/OS UNIX". Familiarity with developing a website using the IBM z/OS HTTP server, such as might be obtained with attending U518: "You and z/OS and The World Wide Web".

Duration

Two Days

Course Outline

Writing z/OS CGIs in Assembler

Course Outline

I. *General Program Structure and Techniques*
- A. General program structure
- B. Redirect using printf
- C. Redirect using bpx1wrt
- D. Watching for errors
- E. Deploying your CGI
- Computer Exercise: Setting up for labs: n

II. *Basic Processing*
- A. Emitting Headers
- B. Emitting HTML
- C. Accessing environment variables
- D. Displaying environment variables
- E. Stylesheets and CGIs
- Computer Exercise: Writing out HTML pages b

III. *Handling GET Requests*
- A. Some scenarios
- B. Parsing QUERY_STRING content
- C. Decoding QUERY_STRING content
- Computer Exercise: Handling incoming data b

IV. *The Data Connection - Part I: The Story*
- A. Working With Data on the Server
- B. The Data Connection - Part II: Working With VSAM Data
- C. Working with VSAM files
- Computer Exercise: Working with VSAM data

V. *The Data Connection - Part III: Working With DB2 Data*
- A. Working with DB2 data
- Computer Exercise: Working wth DB2 Data (optional)

VI. *Hidden Controls and cookies*
- A. Session continuity
- B. Hidden controls
- C. Cookies
- D. Modifying the previous CGI [to emit data]
- E. Designing the invoked CGI [to catch data]
- F. Coding the invoked CGI [to catch data]
- Computer Exercise: The Persistence of Memory .

VII. *POST Requests*
- A. Finding needed storage size
- B. Allocating storage
- C. The CGIGETBF Routine
- D. Reading from stdin
- E. Breaking Apart Headers and Data
- F. Our Sample POST CGI Logic
- G. The TCAPSTB CGI code
- Computer Exercise: Handling POST Processing

VIII. *Handling Files Sent by POST*
- A. File Handling
- Computer Exercise: Saving and Linking to

IX. *Working With Unicode Data*
- A. The Role of Unicode
- B. CGIs and Unicode
- Computer Exercise: Working With Unidcode

X. *Submitting jobs from a CGI*
- A. Set up
- B. Logic
- Computer Exercise: Submitting a job (optional)

XI. *Wrap up*