Writing z/OS CGIs in Assembler

Course Summary

Description

CGI stands for Common Gateway Interface, and it is the most widely used programming technology for creating dynamic web pages at the server. Students who complete this course will be able to design and code CGI programs in Assembler to run on z/OS used as a web server.

Objectives

After taking this course, students will be able to:

- Code, compile, bind, debug, deploy, and maintain CGIs written in Assembler for the z/OS environment
  - Handle GET and POST requests: analyze and take action, as appropriate, such as
    - Parse and decode a QUERY_STRING value for GET
  - Gather in the stdin data for POST, then
    - Save a file as is or translated to EBCDIC on the mainframe, for POST
- Produce responses that are HTML pages or redirection
- 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 (reentrant, LE-enabled, Assembler CGI)
- Using printf() and bpx1wrt to emit XHTML
- Deploying a CGI
- Handling GET requests
- Accessing environment variables
- Parsing QUERY_STRING
- Decoding QUERY_STRING
- Working with data on the server
- VSAM KSDS access
- DB2 access
- Hidden controls and cookies
- Handling POST requests
- Handling files sent by POST
- Working with Unicode
- Submitting jobs from a CGI

Audience

This course is designed for assembler programmers experienced with working in an OS/390 or z/OS environment who will be designing and coding CGI applications that are to be run on z/OS.

Prerequisites

At the very least, the student entering this course should have an understanding of CGI concepts and design issues, such as might be obtained from attending PT5646 "Introduction to CGIs on z/OS".

Duration

Two days

ProTech protechtraining.com *"Charting the Course ... ... to Your Success!"*

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
- F. Computer Exercise: Setting up for labs

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

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

IV. The Data Connection - Part I: The Story
- A. Working With Data on the Server

V. The Data Connection - Part II: Working With VSAM Data
- A. Working with VSAM files
- B. Computer Exercise: Working with VSAM data

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

VII. 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]
- G. Computer Exercise: The Persistence of Memory

VIII. 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
- H. Computer Exercise: Handling POST Processing

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

X. Working With Unicode Data
- A. The Role of Unicode
- B. CGIs and Unicode
- C. Computer Exercise: Working With Unicode

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