

Hosting a Web Site on z/OS

- one person's experience

- Background
- Environment
- The HTTP server
- The Configuration File
- What Directory to Start In
- What Page To Display
- Directives for Individual Users
- File Type Support
- CGI Support
- Putting Pages on Your Site
 - ◆ A Sample XHTML Page
 - ◆ Getting X/HTML Files To Your Web Directory
- Summary

Background

This paper is the result of my personal saga to install a web site on our small z/OS system

- ◆ I am not an authoratative source, not even a full-fledged systems programmer

So this is the story of what has worked for me

- ◆ If these techniques work for you, or let you uncover techniques that work for you, that's terrific

- ◆ But there is no guarantee this document is definitive, and it may even contain some misunderstandings

To begin, I can't even remember when I discovered you could host a web site under z/OS (it was OS/390 in those days, I recall)

- ◆ But ever since I found out you could do this - and without having to purchase an expensive product like WebSphere, I was determined to figure it out

➤ And I have

Environment

- ❑ **You need to have a lot of environment and infrastructure built - some by you, some for you:**
 - ◆ **Network design and configuration, inside your firewall and outside the firewall**
 - ✗ Connections and IP addresses chosen, arranged for, paid for
 - ✗ For outside work, a domain name registered that points to the IP address of your z/OS system
 - ✗ For inside work, a system name or internal IP address established for your z/OS system
 - ✗ TCP/IP configuration files must be set up to reflect all this to the system
 - ◆ **z/OS has to have z/OS UNIX services configured and active**
 - ◆ **Users have to be defined to your security package**
 - ✗ They must have an Open Edition (OMVS) segment defined
 - ✗ They may have a TSO segment defined
 - ◆ **Now we get to the HTTP server, which is what will serve our web pages up ...**

The HTTP Server

- ❑ Your HTTP Server may be brought up when your system is IPL'ed
 - ◆ We have the command "s httpd1" in our VTAM startup parmlib member

- ❑ Or it can be started with an operator command
 - ◆ Explicitly issuing the command "s httpd1" has the same effect as including the command in your startup files

- ❑ In either case, the HTTPD1 proc is invoked; this proc invokes the program IMWHTTPD passing the parm string (in our case):

```
ENVAR("_CEE_ENVFILE=/web/httpd.envars")/ -vv -r /web/httpd.conf -B
```

- ✗ The ENVAR tells the server where to find the environment variables to establish

- ✗ The -vv flag indicates second level tracing should be used and the trace should go to stderr

- ✗ The -r flag points to the configuration file (also called the rules file in the documentation)

- ✗ The -B stands for "bounce", which, according to the docs, avoids "TIMED_WAIT delay" when the server terminates

- ◆ Many other options are available to set at start up, and many can be modified by operator commands after the server is running

The Configuration File

- ❑ **The file we call /web/httpd.conf, the configuration file, defines most behaviors of the server**
 - ◆ **IBM supplies a sample / starter file that is exhaustively commented (and this is supplemented by the HTTP doc)**
 - ◆ **Most of the options are either defaulted or commented out, so you simply modify or uncomment or add additional entries in the same style as in the sample**
 - ◆ **The following pages discuss the most central issues ...**

What Directory to Start In

- When someone requests your site, it is probably through a URI of the form: `http://www.your_domain_name.com`

- The domain name you have registered is passed to a Domain Name Server (DNS) on the Internet to find your actual IP address, and that address is contacted
 - ◆ This request goes to your server, which has been waiting for just such a request

- The configuration file includes what are called "Pass rules" that say:

If the request is for this location, send it to this directory

- ◆ The locations use characters and wild cards to establish templates

✗ For example, the default Pass rules included these lines:

```
Pass    /Docs/*    /usr/lpp/internet/server_root/Docs/*
.
.
.
Pass    /*       /web/pub/*
```

✗ Which directs requests to `www.your_domain_name.com/Docs` to the directory `/usr/lpp/internet/server_root/Docs`

✗ And requests to just `www.your_domain_name.com` to the directory `/web/pub`

What Page to Display

- ❑ Another section in your configuration file, called the "Welcome directives", specifies what pages to look for if no specific page is requested

- ◆ That is, if your server gets a request for *your_domain_name.com/Docs/report-5.html*, the server looks for */usr/lpp/internet/server_root/Docs/report-5.html*
- ◆ But if you get a request just for *your_domain_name.com/Docs*, which page in that directory should be served?

- ❑ The syntax of a Welcome directive is the word Welcome, some whitespace, and a (case sensitive) page name

- ◆ Usually, you supply a list of page names, and the server looks in the selected directory until it finds a page with a name from the list (order is important)
- ◆ For example, our Welcome directives are

Welcome	Welcome.html
Welcome	welcome.html
Welcome	index.html
Welcome	Frntpage.html

- ◆ So if a request comes in for *your_domain_name.com* and no page name, the server will look in */web/pub* for the four file names above
 - ✗ The first one it finds will be chosen
- ◆ Notice how this lets you change the default page by simply providing a page with a name higher in the priority list into a target directory!

Directives for Individual Users

- ❑ So far, what we have looked at is site-wide page selection and serving

- ❑ Often you will want to have separate places for developers to build sites, either for development or production
 - ◆ These places could be tied to developer id's, or you could establish a project id

- ❑ For example, my id is "scomsto" and I would like to be able to designate a location where I can create and test web pages

- ❑ This is done through the "UserDir" directive, with the syntax of the word UserDir, some whitespace, and the name of a sub-directory to use by default

For example

```
UserDir    public_html
```

- ◆ This means that for any id we create, the directory designated as the home directory for that id can use a sub-directory named public_html for holding HTML pages

X Let's explore this a little more ...

Directives for Individual Users, 2

☐ There may be only one UserDir directive, and the value supplied by IBM is "public", but we found many installations seem to prefer "public_html", so we decided to go with that

☐ Now, if someone wants to see my HTML files, they send a URI like this:

`http://www.domain_name.com/~scomsto`

◆ This will map to my home directory and the public_html sub-directory, in my case:

`/u/scomsto/public_html`

◆ And then the server will use the pages in the Welcome directives, looking for a page with the same name as those in that list

◆ Of course, a requestor may request a specific page, such as

`http://www.domain_name.com/~scomsto/report-5.html`

X and the server will send the file

`/u/scomsto/public_html/report-5.html`

File Type Support

- ❑ The configuration file has a section where file name suffixes imply information about content and encoding, so the server knows how to handle them

- ❑ The syntax is: AddType, some whitespace, the suffix, the mime type, the encoding, a weighting factor (0.0 - 1.0), and an optional comment
 - ◆ You can get a good feel for this by looking at your starter file, or checking out the docs

 - ◆ We made two changes to the supplied rules:

Add a suffix for javascript files:

```
AddType .js text/javascript ebcdic 0.5 # javascript
```

Changed the supplied encoding for cascading style sheets

```
AddType .css text/css ebcdic 0.5 # Cascading SS
```

- ✗ The supplied encoding for cascading style sheets is 8bit,
 - which means the server will not translate this when the file is served

 - So the file had better be, essentially, ascii, which means you can't edit it with the ISPF editor

- ✗ This bad default is established due to historical reasons

CGI Support

- ❑ A CGI is a Common Gateway Interface program or script
 - ◆ CGIs are used to process requests from the client
 - ◆ CGIs are run on the server and usually dynamically build HTML pages and pass them back to the client
- ❑ You explicitly open up support for CGIs to be run from specific directories using the "Exec" directive (coded in the same place as your Pass rules)

- ◆ The default includes:

```
Exec /cgi-bin/* /usr/lpp/internet/server_root/cgi-bin/*
```

- ✗ which says requests to run CGIs in directory cgi-bin will be satisfied out of directory
/usr/lpp/internet/server_root/cgi-bin

- ◆ We added CGI support for our ids by this kind of mechanism

```
Exec /SCOMSTO/* /u/scomsto/CGI/*
```

- ✗ So any request for a CGI to be run from SCOMSTO will be run from /u/scomsto/CGI

CGI Support, 2

- ❑ Just so you know, a CGI request typically appears in form tags in HTML or XHTML

For example

```
<form action="/SCOMSTO/cgi_vars.rex" method="get" >  
.  
.  
.  
</form>
```

- ◆ In our situation, then, the CGI REXX program named `cgi_vars.rex` had better be found in `/u/scomsto/CGI`
- ❑ CGI's may be written in just about any language

Putting Pages on Your Site

- ❑ Now, to put pages on your website, you need to get HTML or XHTML (X/HTML for short) files into your web directory in the HFS (we'll assume */u/your_id/public_html* for purposes of discussion)
 - ◆ This requires two major skills
 - X Knowledge of X/HTML
 - X Ability to get pages into your directory

- ❑ Of course, we provide training in both these areas, but here we give you a fast leg up so you can experiment:
 - ◆ A sample XHTML page
 - ◆ Necessary commands to get a file into your directory

A Sample XHTML Page

□ Below is a minimal XHTML page that you can type (or perhaps cut and paste) into an editor (see next page for a couple of choices)

- ◆ It simply displays a start up page, so it should have a file name that appears in the list of file names in your Welcome directives discussed earlier, such as Welcome.html or index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<title>Our Home Web Site</title>
<body>
<h1>Welcome to ____'s home page</h1>
<p>This is the starting point for our great Web adventure!</p>
</body>
</html>
```

Getting X/HTML Files To Your Web Directory

☐ There are four general approaches to accomplishing this

- ◆ 1. Use the ISPF editor to create your XHTML file as a member in a PDS or PDSE

X Then from ISPF option 6 command line, issue this command:

```
oput libry(member) '/u/your_id/public_html/index.html'
```

- ◆ 2. From ISPF option 6, issue the OMVS command to get into the z/OS UNIX shell; from there issue these commands ...

```
cd /u/your_id/public_html  
oedit index.html
```

you can now issue classic ISPF edit commands, end with F3

```
chmod 644 index.html  
exit
```

- ◆ 3. From a workstation, telnet into your z/OS system and use an editor such as vi to create your file - too complex to discuss here
- ◆ 4. Create your file on your workstation then upload it to your z/OS system - too complex to discuss here

Summary

So to host a web site on your z/OS system

X Find out what host name or IP address will reach the host from a browser (may use different values from inside the corporate firewall and outside) - talk with responsible systems person

➤ For example: OurHost.com

X Choose an ID (an individual user or an ID chosen just for this purpose) and build a security entry that includes an OMVS segment - request of appropriate systems person

➤ For example: OURWEB

X Modify your configuration file to include UserDir, Welcome, Pass, Exec, and AddType directives as needed or anticipated - may need to have done by systems person

X Build the sub-directory specified in your UserDir below the web ID's home directory, for example /u/ourweb/public_html or /u/your_id/public_html

X Start coding X/HTML files in the web directory (good to include a page with a name in the Welcome directives list)

X Test by pointing your browser to
<http://www.OurHost.com/~ourweb/>
or
http://www.OurHost.com/~your_id/

Summary, 2

- ❑ For training in working with z/OS UNIX and coding web pages under z/OS, check out The Trainer's Friend, Inc. website at

<http://www.trainersfriend.com>

- ◆ Follow the links to Topics, then to UNIX System Services

X This page contains information about our most relevant courses for this area, and it is being updated constantly

X Collectively, the courses listed include these topics (and more)

- z/OS UNIX concepts and commands
 - Including editors **oedit**, **ed**, and **sed** (and, optionally, **vi**)
- Accessing z/OS UNIX using the TSO/ISPF **omvs** command (and, optionally, **telnet**)
- HTML and XHTML standards
- Substantial subsets from the standards for Cascading Style Sheets (CSS), the Document Object Model (DOM), and ECMAScript (basis for Javascript)
- Shell script writing
- Coding, compiling, binding, and running programs written in COBOL, C, PL/I, and Assembler, all under the shell
- Running shell scripts and programs in batch
- Generating X/HTML pages using shell commands and scripts