The Trainer's Friend, Inc.

6790 E. Cedar Ave., Suite 201
Denver, Colorado 80224
U.S.A.
E-mail:
Internet:

Telephone: (303) 355-2752

trainers@trainersfriend.com
www.trainersfriend.com

Coding AJAX Applications - supporting files

Set Up Instructions

Thanks for ordering the Coding AJAX Applications paper. I hope you find it interesting and helpful.

If you are interested in actually running the main demo discussed in this paper on your system, I have created a file that provides all the pieces you need to do this. Just follow these instructions...

Attached is a single file for you to use in testing / demo-ing our files that illustrate:

*** Using AJAX coding techniques**

*** Hosting a website on z/OS**

*** Using COBOL for a CGI to handle AJAX**

The steps to take are:

1. Upload the attached file (AJAX.FILE) as binary (if you need to pre-allocate, use these attributes: PS, FB 80 / 3120, 1 cyl pri, 1 cyl sec)

2. Under ISPF 6, unload the file:

==> receive inda(ajax.file)

when you get these messages:

INMR901I Dataset xxxxx.xxxxx.xxxx from xxxxx on xxxx
INMR906A Enter restore parameters or 'DELETE' or 'END' +

reply by pressing 'Enter' to accept the default data set name, which is 'AJAX.LIBRARY', prefixed by your TSO id as the *hlq* (only works if your TSO PROFILE indicates PREFIX(hlq)); or key in

dsn('*fully_qualified_desired_data_set_name***')**

You now have a PDS with these members:

#AJVCOB - a job to compile and bind the COBOL program AJVCOB

AJAXKSDS - a job to create <hlq>**.AJAX.KSDS from** <hlq>**.AJAX.INPUTA**

AJAXLOAD - an unloaded PDS load library

AJAXPAX - a pax'ed member containing all the needed HFS files

AJAXPDSE - a PDSE version of AJAXLOAD

INPUTA - an unloaded sequential file

3. Prepare the target library for compiles and binds:

If your installation supports PDSEs, restore AJAXPDSE, resulting in <hlq>**.AJAX.PDSE:**

==> receive inda(ajax.library(ajaxpdse))

If your installation does not support PDSEs, restore AJAXLOAD, resulting in <hlq>**.AJAX.LOADLIB:**

==> receive inda(ajax.library(ajaxload))

4. Prepare your VSAM file

a. restore INPUTA, resulting in <hlq>**.ajax.inputa:**

==> receive inda(ajax.library(inputa))

b. edit member AJAXKSDS, c all '+++++++' to your high level qualifier and run the job, resulting in <hlq>**.AJAX.KSDS**

5. Change and compile the COBOL CGI

get into edit of <*hlq*>.AJAX.LIBRARY(#AJVCOB)

*** Insert a valid JOB statement at the top**

*** If your high level qualifier is not your tso id, change the first SET statement to be your high level qualifier**

*** Check all the other SET statements to match the data set names in your shop**

*** If your shop does not support PDSEs, change the set of TARGET to be SET TARGET=LOADLIB**

*** Change the occurrence of scomsto to be your high level qualifier (do a find on scomsto); note that the result should be lower case**

*** Change the filename in the program (do a find on 'CHANGE BELOW') to allocate your KSDS file**

and submit the job

6. Prepare the z/OS UNIX pieces:

a. You should be running one of the free HTTP servers from IBM, and in the configuration file (probably httpd.conf in some directory) be sure the following are in place:

UserDir public_html - If not "public_html", jot down the value

AddType .js text/javascript ebcdic 0.5 # javascript
AddType .css text/css ebcdic 0.5 # Cascading SS

(If not already set up this way)

Exec /*YOURID*/* /u/*yourid*/CGI/*

(establish a place where you can put CGI programs that won't interfere with other work)

6. Prepare the z/OS UNIX pieces, continued:

Do the following steps under omvs:

b. Some user (yourself?) should be selected as the id under which the pages will be served, probably in /u/*yourid*/public_html

Note: you do not need any special privileges
Note: if this directory does not exist, you should create it:

```
cd /u/yourid
mkdir public_html
```

c. Create your CGI directory, if it does not already exist:

```
mkdir CGI
```

d. Check permissions of both public_html and CGI: they should be 755

```
chmod 755 public_html
chmod 755 CGI
```

e. Unload the pax file - get into your html directory and issue a pax command:

```
cd  /u/yourid/public_html
pax -rvf "//'<hlq>.ajax.library(ajaxpax)'"
```

(for example: pax -rvf "//stnt003.ajax.library(ajaxpax)' ")

this establishes all the files you need in /u/*yourid*/public_html

f. oedit ajaxvcob.html and change SCOMSTO to YOURID **(in upper case):**

oedit ajaxvcob.html

make the change

then End **(F3)**

g. exit omvs

7. Prepare the COBOL CGI:

put the executable from your earlier job into your CGI directory from ISPF 6, something like:

===> oput ajax.pdse(ajvcob) '/u/*yourid*/CGI/ajvcob'

(note: the first time, you will have to get into omvs and get into your CGI directory and issue "chmod 755 ajvcob" after that, the permissions will remain even if you replace the program)

(note, if you are not using the pdse, use the loadlib:

===> oput ajax.loadlib(ajvcob) '/u/*yourid*/CGI/ajvcob'

8. Ready to test / demo...

Point your browser to http://*system_id*/~*yourid*/ajaxvcob.html

where "*system_id*" is the IP address or domain name for the z/OS system you've been working on

Similarly, test ajaxpic.html, newajaxxml.html, and ajaxlog.html

(with similar changes as above to each)

All the pieces you need should be in your public_html or CGI library.

Note: you will have to move logger.php to your CGI directory.