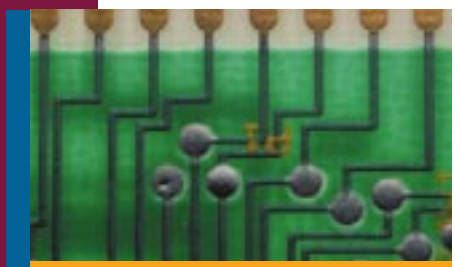

BY PATRICK RENARD

Implementing a Web Server on OS/390: Part III — Writing Common Gateway Interfaces and Installing Java Virtual Machine

This article presents programming techniques to write Common Gateway Interfaces (CGIs) on OS/390 to enhance your server with dynamically built web pages. It will also describe how to install and to use Java Virtual Machine (JVM) on OS/390.

PARTS I and II of this series examined the technical implementation of an ICSS server on OS/390. This article presents programming techniques to write Common Gateway Interfaces (CGIs) on OS/390 to enhance your server with dynamically built web pages. It will also describe how to install and to use Java Virtual Machine (JVM) on OS/390.

USING COMMON GATEWAY INTERFACES

IBM provides a set of gateway interfaces from the web server to back-end transaction and database servers. Using HTML alone, you can only create static web pages. To access data from the web, you need to write CGI programs to dynamically build web pages.

CGI is a standard, supported by almost all web servers, that defines how information is exchanged between a web server and an external program (CGI program). Basically, anytime you want to take input from the browser and generate a response, you can use a CGI program. CGI programs can be written in any language supported by the operating system. So on OS/390 you can use REXX, C/C++ or Java.

Note: The only available ICSS documentation on CGI programming, the *ICSS Web Programming Guide*, can be found on the web at www.ics.raleigh.ibm.com/pub/icsw-pg.htm.

CGI programs must be stored in specific directories located by " Exec" directives in httpd.conf, as shown in Figure 1. In general, CGI programs process the data they receive in three stages:

1. **Parsing:** In this stage, the program takes input data in one of the possible formats and breaks it into individual variables. For example, the following kind of data will be received by sample REXX CGI programs presented in this article:

```
var1=Value2&var2=Option3&var3=Option
      1&var3=Option3
```

Parsing will break this string to individual variables used in the "data manipulation" block of the CGI program. Two methods are available to call CGI programs:

```
POST method: Input data is read
     directly from STDIN file.
GET method: Input data can be
     obtained from QUERY_STRING
     variable and CGIPARSE function.
```

2. **Data manipulation:** In this stage, the CGI program uses parsed data to perform appropriate actions. The "logic" of your program is included in this section.

3. **Response generation:** This is done through the STDOUT file. The CGI program is responsible for writing an output HTML document in the STDOUT file. The response must contain at least

Figure 1: Sample Exec Directive to Locate CGI Programs

```
£ www02 server
£
Exec       /cgi/*       /u/imwebsrv/www02/cgi/*       www02.mzsmvs.mvs.ctrne.fr
Pass       /*           /u/imwebsrv/www02/*           www02.mzsmvs.mvs.ctrne.fr
£
```

Figure 2: Sample HTML Code to Call a REXX CGI Using POST Method

```
<h2> Rexx CGI </h2>
<a href="/cgi/showfile.cmd?back=/html/cgidemo.html%file=/u/imwebsrv/www02/cgi/cgirexx1.cmd
   View Rexx source code.</a>
<form method="post" action="/cgi/cgirexx1.cmd">
   <§————————————————————————————————————————>
   <b> Variable 1 </b> - (Input Field) -
     <input name="VAR1" size=8>
     <br>
   <§————————————————————————————————————————>
   <b> Variable 2 </b> - (Check Box) -
     <input type="checkbox" name="VAR2" value="Value1"> Value1
     <input type="checkbox" name="VAR2" value="Value2"> Value2
<br>
   <§————————————————————————————————————————>
   <b> Variable 3 </b> - (Selectable List / Multiple choices) -
     <select name="VAR3" size=4 multiple>
                      <option selected> Option1
                      <option> Option2
                      <option selected> Option3
     </select>
     <br>
   <§————————————————————————————————————————>
     <input type="submit" value="Execute Rexx CGI">
     <input type="reset" value="Reset all fields">
   <§————————————————————————————————————————>
</form>
```

Figure 3: Sample REXX CGI Using POST Method

```
/* REXX */

/* generate MIME header */

'cgiutils -status 200 -ct text/x-ssi-html'

address mvs 'EXECIO 1 DISKR STDIN (STEM infile.'
stdin = infile.1
parm.='';
do while infile.1 > ''
   parse var infile.1 varname '=' value '&' infile.1
   if parm.varname = '' then  parm.varname = value
                        else  parm.varname = parm.varname ' ' value
end say '<html>'
  say '<body>'
    say '<h1> Processing form with Rexx CGI script. </h1>'
    say '<p>'

    /* get stdin value */

    say 'Value for STDIN = '
    say '<b>'
    say stdin
    say '</b> <br>'
    say '<p>'

    /* get variable 1 value */

    say 'Value for variable 1 = '


say '<b>'
       say parm.var1
    say '</b> <br>'

    /* get variable 2 value */

    say 'Value for variable 2 = '
    say '<b>'
       say parm.var2
    say '</b> <br>'

    /* get variable 3 value */

    say 'Value for variable 3 = '
    say '<b>' say parm.var3
    say '</b> <br>'
  say '</body>'
say '</html>'
```

Figure 4: Sample HTML Code to Call a REXX CGI Using GET Method

```
<§— to use CGIPARSE => method=get ————————————————>
<form method="get" action="/cgi/cgirexx2.cmd">
...
```

one MIME header (Content-Type) followed by a blank line. The blank line separates the headers from the content of the response. The MIME header can be produced manually or using the CGIUTILS command.

REXX is a suitable choice to write CGIs because it is a very easy language to write, can be deployed quickly and can be modified very easily.

WRITING REXX CGIS

REXX is a suitable choice to write CGIs because it is a very easy language to write, can be deployed quickly and can be modified very easily. There is no noticeable speed difference for an end user between a small REXX program and a small C program, since most of the response time delay is due to the network, which far overshadows the machine time execution differences. The following sections highlight two coding techniques of REXX programming.

REXX CGIs Using the POST Method

The HTML sample in Figure 2 shows how to call a REXX CGI to process a form. With the POST method (Figure 3), input data is read from the STDIN input file and parsed manually. Note also that in this sample, the MIME header is produced using CGIUTILS command.

REXX CGIs Using the GET Method

To use the GET method, you only need to change the "method" parameter in the HTML form tag as shown in Figure 4. With the GET method, you can use the CGIPARSE command to automatically parse query strings. Notice in Figure 5 that instead of using the CGIUTILS command, the MIME header is produced manually.

WRITING C/C++ CGIS

The most popular language to write CGIs is C. On OS/390, you can also use this language to write your own CGI programs. C/C++ CGI programs run faster than REXX CGIs but are more complex to write. They have to follow the same logic as other

Figure 5: Sample REXX CGI Using GET Method

```
/* REXX */

/* generate MIME header */

say 'Content-Type: text/x-ssi-html'
say ''

say '<html>'
   say '<body>'
      say '<h1> Processing form with Rexx CGI script. </h1>'
      say '<p>'

      say 'Request method = '
      say '<b><§—£echo var="REQUEST_METHOD" —> </b> <br>'
      say '<p>'
say 'Query string = '
      say '<b><§—£echo var="QUERY_STRING" —> </b> <br>'
      say '<p>'

      /* get variable 1 value */

      say 'Value for variable 1 = '
      say '<b>'
        'cgiparse -value var1'
      say '</b> <br>'

      /* get variable 2 value */

      say 'Value for variable 2 = '
      say '<b>'
      'cgiparse -value var2'

say '</b> <br>'
      say '<p>'

      say 'Value for variable 3 = '
      say '<b>'
      'cgiparse -value var3'
      say '</b> <br>'
      say '<p>'

      /* get number of unique fields in QUERY_STRING */

      say 'Number of unique fields = '
      say '<b>'
      'cgiparse -form -count'
      say '</b> <br>'

   say '</body>'

say '</html>'
```

Figure 6: Batch Job to Compile and Linkedit a C/C++ CGI

```
//*
//STEP01   EXEC PROC=CBCCL
//COMPILE.SYSIN  DD DISP=SHR,DSN=I990557.PDS.C(CGIC)
//LKED.SYSLMOD   DD  PATH='/u/imwebsrv/www02/cgi',
//        PATHMODE=(SIRWXU,SIRGRP,SIXGRP,SIROTH,SIXOTH)
//LKED.SYSIN DD *
 NAME cgic(R)
/*
```

Figure 7: Path Update to Access Java Modules

```
BROWSE — /etc/profile ———————————————— Line 00000000 Col 001 043
******************************** Top of Data ********************************
export PATH=.:/bin:/usr/lpp/java/J1.1.1/bin
export _BPX_SHAREAS=YES
export _BPX_SPAWN_SCRIPT=YES
PS1='$LOGNAME':'$PWD':' >'
export PS1
set -o vi
******************************** Bottom of Data ******************************
```

Figure 8: Check Java Installation

```
I990557:/u/i990557: >java -version
java version "JDK1.1.1 IBM build m111-19970926"
```

programs: They have to read and parse input data from STDIN and write their output result to STDOUT.

A sample CGI written in C/C++ is available for download from the NaSCOM Internet server as filename NOV98002.EX1. The JCL in Figure 6 shows how to compile and linkedit a C CGI program in an OpenEdition HFS.

Java for OS/390 can be used to create applications that might otherwise have been written in C/C++, COBOL, or PL/I.

RUNNING JAVA ON OS/390

Java is an object-oriented environment developed by Sun Microsystems that operates independently of any operating system or microprocessor. Java for OS/390 is an IBM product (program number 5655-A46) that can be ordered for free. IBM's implementation of Java for OS/390 is based on Sun's Java Developers Kit (JDK) for Solaris. IBM's implementation of Java for OS/390 includes these components:

- a compiler
- a debugger
- an optimized Java byte-code interpreter
- just-in-time compiler (JIT)

Java for OS/390 can be used to create applications that might otherwise have been written in C/C++, COBOL, or PL/I. In this case, the JDK can be used to create and test applications to be executed on OS/390. This allows a customer to use the Java skill base for both web and business application development. Java applications are not limited to web usage. Any application that requires platform portability can take advantage of the benefits of Java.

Applets enable web users to deliver more visually compelling web content, including animation. Applets can be dynamically transmitted over a network and run on any system that is enabled for Java. Because applet execution is platform-independent, an applet developed with the OS/390 tools can be executed on any Java-enabled platform such as OS/2 or Windows 95. Applets and applications developed on OS/2, Windows 95, or other platforms can execute on OS/390.

Figure 9: Sample Helloworld Java Program

```
EDIT       /u/i990557/java_src/helloworld.java           Columns 00001 00072
********************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000001 import java.applet.Applet;
000002 import java.awt.Graphics;
000003 public class helloworld extends Applet
000004   {
000005     public void paint(Graphics g)
000006       {
000007         g.drawString("hello World from OS/390...",20,25);
000008       }
000009   }
******************************** Bottom of Data ****************************
```

Figure 10: Compiling Your Java Class

```
I990557:/u/i990557/java_src: >javac
   helloworld.java
```

Figure 11: Export Your Workstation Terminal

```
I990557:/u/i990557/java_src: >export
   DISPLAY=126.16.4.128:0.0
```

Figure 12: Sample HTML Code to Use appletviewer

```
BROWSE — /u/i990557/java_src/java.html ———————— Line 00000000 Col 001 054
******************************** Top of Data ********************************
<html>
 <applet code="helloworld.class" width=500 height=300>
 </applet>
</html>
******************************** Bottom of Data *****************************
```

Figure 13: Starting appletviewer to Run Your Applet

```
I990557:/u/i990557/java_src: >appletviewer java.html
```

Figure 14: httpd.conf Directives for Java CGI

```
£
£ www02 server
£
Exec       /cgi/*       /u/imwebsrv/www02/cgi/*       www02.mzsmvs.mvs.ctrne.fr
Exec       /java/*      /u/imwebsrv/www02/java/*      www02.mzsmvs.mvs.ctrne.fr
Pass       /*           /u/imwebsrv/www02/*           www02.mzsmvs.mvs.ctrne.fr
£
```

Figure 15: httpd.envvars Sample for Java CGI

```
PATH=/bin:.:/usr/lpp/internet/bin:/usr/lpp/java/J1.1.1/bin
SHELL=/bin/sh
TZ=ESTOEDTO
LANG=C
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%L/%N
LIBPATH=/usr/lpp/internet/bin:/usr/lpp/internet/sbin:...:/usr/lpp/java/J1.1.1/lib/mvs/native_t
hreads
CLASSPATH=/usr/lpp/java/J1.1.1/lib/classes.zip:/u/imwebsrv/www02/java
```

Figure 16: HTML Sample Code to Call a Java CGI

```
<h2> Java CGI </h2>
<img src="/gif/java.gif">
<a href="/cgi/showfile.cmd?back=/html/cgidemo.html%file=/u/imwebsrv/www02/java/cgijava.java">
   View Java source code.</a>
<form method="post" action="/java/cgijava.class">
   <b> Variable 1 </b> - (Selectable List / Multiple choices) -
     <select name="var1" size=4 multiple>
                    <option selected> Option1
                    <option> Option2
                    <option selected> Option3
                    <option> Option4
                    <option> Option5
     </select>
     <br>
   <§————————————————————————————————————————————————>
   <b> Variable 2 </b> - TextArea -
<textarea name="var2" rows=3 cols=30>
     </textarea>
     <br>
   <§————————————————————————————————————————————————>
     <input type="submit" value="Execute Java CGI">
     <input type="reset" value="Reset all fields">
     <hr>
</form>
```

Java CGI programs have to follow the same rules as other CGI programs. Input to the program is read from STDIN and output is written to STDOUT.

Installation

Java for OS/390 is installed using SMP/E. Two FMIDs exist: HJVA111 for OS/390 releases 1 and 2 and HJVA11A for OS/390 releases 3 and higher. The product is composed of several MVS datasets and one OpenEdition filesystem (mounted on /usr/lpp/java). To access Java modules, you need to update your PATH. This can be done in your /etc/profile, as shown in Figure 7. You can verify your installation with the shell command in Figure 8.

Creating Applets

The intent of this article is not to provide instructions on Java language but rather to demonstrate how to use it under OS/390. Several good books have been written on this subject. For this demonstration, we will use the "famous" and simple HelloWorld applet. Type the source code shown in Figure 9 in a file named helloworld.java (all Java source files need to end with .java). Then you will need to compile it. Start the compilation with the shell command shown in Figure 10. If the code compiles successfully, you should not get any error messages.

To use Java Windowing Support (AWT), the X/Windows feature of TCP/IP must be installed on the OS/390 host. Your workstation must also have an X/Windows emulator to communicate with the host.

Before starting "appletviewer," you should export your workstation terminal using the code provided in Figure 11. You also need to code an HTML page to call your applet.

To accomplish this, use the code provided in Figure 12. Then you are ready to run your applet using the shell command shown in Figure 13. After a few seconds you should see your applet running on the screen.

Using Java CGI with ICSS

The facility called JGI (Java Gateway Interface) uses a Java Virtual Machine running on OS/390. First, you will need to update httpd.conf to add an Exec directive to locate your Java CGI. See Figure 14. Next, you will need to modify httpd.envvars to add a PATH and a LIBPATH entry to locate Java modules. You should also add CLASSPATH definitions to locate Java classes. See Figure 15.

Java CGI programs have to follow the same rules as other CGI programs. Input to the program is read from STDIN and output is written to STDOUT. To demonstrate the use of a Java CGI, we will use an adaptation in Java of the C sample program provided in the *ICSS Programming Guide*. This program reads data entered in an HTML form and displays it again on your browser. The HTML sample in Figure 16 will show you how to call this Java CGI. The source code for this Java CGI program is available for download from the NaSCOM Internet server as filename NOV98002.EX1.

CONCLUSION

The concluding article will examine how to interconnect Web applications and strategic OS/390 products such as DB2 and CICS. ts

NaSPA member Patrick Renard has been an MVS systems programmer for eight years. His experience includes DB2 Data Sharing and Parallel Sysplex implementation. He can be reached at renard@mygale.org.

