CGI scripts using Ruby Language

Solanki Gautam V. (04305027)

Department of Computer Science

November 27, 2004

Outline of Presentation

What is Ruby Language ?

Ruby is a pure object-oriented scripting language, developed by Yukihiro Matsumoto in Japan. It was primarily designed to handle text processing and systems management tasks. Not only can you write your own SMTP server, FTP daemon, or Web server in Ruby, but you can also use Ruby for more usual tasks such as CGI programming or as a replacement for PHP.

A simple Ruby Program

```
#!/usr/bin/ruby
$var = "Ruby Languages\n"
print "CS701 Assignment\n"
print $var
```

What is Common Gateway Interface (CGI)?

CGI is not a language. It's a simple protocol that can be used to communicate between Web forms and your program. A CGI script can be written in any language that can read STDIN, write to STDOUT, and read environment variables, i.e. virtually any programming language, including C, Perl, or even shell scripting.

Structure of a CGI Script

Here's the typical sequence of steps for a CGI script:

1. Read the user's form input.
   When the user submits the form, your script receives the form data as a set of name-value pairs. The names are what you defined in the INPUT tags (or SELECT or TEXTAREA tags), and the values are whatever the user typed in or selected.
2. Do what you want with the data.

Structure of a CGI Script

Here's the typical sequence of steps for a CGI script:

1. Read the user's form input.
   When the user submits the form, your script receives the form data as a set of name-value pairs. The names are what you defined in the INPUT tags (or SELECT or TEXTAREA tags), and the values are whatever the user typed in or selected.
2. Do what you want with the data.

Structure of a CGI Script(Cont.)

1. Write the HTML response to STDOUT. First, write the line Content-type: text/html
   plus another blank line, to STDOUT. After that, write your HTML response page to STDOUT, and it will be sent to the user when your script is done. That's all there is to it.

Structure of a CGI Script(Cont.)

1. Write the HTML response to STDOUT. First, write the line
   Content-type: text/html
   plus another blank line, to STDOUT. After that, write your HTML response page to STDOUT, and it will be sent to the user when your script is done. That's all there is to it.

Structure of a CGI Script(Cont.)

1. Write the HTML response to STDOUT. First, write the line
   Content-type: text/html
   plus another blank line, to STDOUT. After that, write your HTML response page to STDOUT, and it will be sent to the user when your script is done. That's all there is to it.

Why to use Ruby for writing CGI scripts ?

Reasons

Because it has a rich set of libraries. There is support for threads, sockets, limited object persistence, CGI programs, server-side executables, DB files, and more. There is some support for Tk, with more on the way.

Writing CGI Scripts using Ruby....

To have a Ruby script generate HTML output, all you need is

```
#!/usr/bin/env ruby
print "HTTP/1.0 200 OK"
print "Content-type: text/html"
print "<html><body>CS-701 ASSIGNMENT</body></html>"
```

Example 1

Creating Forms and HTML using Ruby

Program

```
require "cgi"
cgi = CGI.new("html3")  # add HTML generation methods
cgi.out{
  cgi.html{
    cgi.head{ "\n"+cgi.title{"This Is a Test"} } +
    cgi.body{ "\n"+
      cgi.form{"\n"+
        cgi.hr +
        cgi.h1 { "A Form: " } + "\n"+
        cgi.textarea("get_text") +"\n"+ cgi.br + cgi.submi
    }
  }
}
```

Output of Example 1

```
produces:

Content-Type: text/html
Content-Length: 302

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><HTM
<TITLE>This Is a Test</TITLE></HEAD><BODY>
<FORM METHOD="post" ENCTYPE="application/x-www-form-urlenc
<HR><H1>A Form: </H1>
<TEXTAREA NAME="get_text" ROWS="10" COLS="70"></TEXTAREA>
<BR><INPUT TYPE="submit"></FORM></BODY></HTML>
```

Example 2

Cookies
You can store all kinds of interesting stuff on an unsuspecting surfer's machine using cookies. You can create a named cookie object and store a number of values in it. To send it down to the browser, set a "cookie" header in the call to CGI

```
#out.
require "cgi"
cookie = CGI::Cookie.new("rubyweb", "CustID=123", "Part=ABC
cgi = CGI.new("html3")
cgi.out( "cookie" => [cookie] ){
  cgi.html{
    "\nHTML content here"
  }
```

Output of Example 2

produces:

```
Content-Type: text/html
Content-Length: 86
Set-Cookie: rubyweb=CustID\%3D123&Part\%3DABC; path=
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><HTM
HTML content here</HTML>
```

Example 3

Cookies by themselves still need a bit of work to be useful. What we really want is a session: a persistent state for some Web surfer. Sessions are handled with CGI::Session , which uses cookies but provides a higher-level abstraction.

```
require "cgi"
require "cgi/session"
cgi = CGI.new("html3")
sess = CGI::Session.new( cgi, "session_key" => "rubyweb",
        "session_id"  => "9650", "new_session" => true, "p
sess["CustID"] = 123
sess["Part"] = "ABC"
```

Example 3(Cont.)

```
cgi.out{
  cgi.html{"\nHTML content here"}
}
```

This will send a cookie to the user named “rubyweb” with a value of 9650. It will also create a disk file in TMP/web-session.9650 with the key, value pairs for CustID and Part.

Reciveing set of saved session data

When the user returns, all you need is a parameter to indicate the session id. In this example, that would be rubyweb=9650. With that value in the parameters, you'll be able to retrieve the full set of saved session data.

```
require "cgi"
require "cgi/session"


cgi = CGI.new("html3")
sess = CGI::Session.new( cgi, "session_key" => "rubyweb",
                               "prefix" => "web-session.")
```

Reciveing set of saved session data(Cont.)

```
cgi.out{
  cgi.html{
    "\nCustomer #{sess['CustID']} orders an #{sess['Part']]
}
```

When the user returns, all you need is a parameter to indicate the session id. In this example, that would be rubyweb=9650. With that value in the parameters, you'll be able to retrieve the full set of saved session data.

References

- http://www.rubycentral.com/book/web.html
- http://www.rubycentral.com/articles/cgi.html
- http://www-106.ibm.com/developerworks/linux/library/l-ruby1.html
- http://www.jmarshall.com/easy/cgi/