UNIT-1

CGI AND HANDLING COOKIES

Ruby is a general-purpose language; it can't properly be called a *web language* at all. Even so, web applications and web tools in general are among the most common uses of Ruby.

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.

Please spend few minutes with CGI Programming Tutorial for more detail on CGI Programming.

Writing CGI Scripts

The most basic Ruby CGI script looks like this −

```
#!/usr/bin/ruby

puts "HTTP/1.0 200 OK"
puts "Content-type: text/html\n\n"
puts "<html><body>This is a test</body></html>"
```

If you call this script *test.cgi* and uploaded it to a Unix-based Web hosting provider with the right permissions, you could use it as a CGI script.

Here when *test.cgi* is requested from a Web browser, the Web server looks for *test.cgi* on the Web site, and then executes it using the Ruby interpreter. The Ruby script returns a basic HTTP header and then returns a basic HTML document.

Using cgi.rb

Ruby comes with a special library called **cgi** that enables more sophisticated interactions than those with the preceding CGI script.

Let's create a basic CGI script that uses cgi −

```
#!/usr/bin/ruby

require 'cgi'
cgi = CGI.new

puts cgi.header
puts "<html><body>This is a test</body></html>"
```

Here, you created a CGI object and used it to print the header line for you.

HTTP protocol is a stateless protocol. But for a business website, it needs to keep session information between different pages.

If the user site registration process needs to jump page, but want to ensure that the information is not lost before filling.

In this case Cookie good to help us solve the problem.

Cookie How does it work?

Almost all the web designers during the design of the site use the Cookie, because they want to give the user browsing the site to provide a more friendly, human culture browsing environment, but also to more accurately collect visitor information.

Writing and reading

Cookies set belongs to the collection of data objects and Request Response object, you need to precede it with the use of Request or Response.

The syntax for the client to send Cookies usually:

When set to a non-existent Cookies settings will be created on the client, if the Cookies already exist, it will be replaced. Since Cookies are sent as part of a client's HTTP transport header information, it is sent to the client code Cookies usually placed before the tag sent to the browser's HTML file.

Cookies If you want to read, you must use the Request object's Cookies collection, its use is: Note that, not only in the server before downloading any data to the browser, the browser and the Server to exchange data Cookies collection Once the browser starts receiving Server downloaded data, Cookies data exchange is stopped, in order to avoid errors, to add response.Buffer = True in the previous program.

Attribute collection

- **1.Expires attribute:** This attribute is used to Cookies set a deadline within the time limit as long as the open web page can call saved Cookies, Cookies If after this period will be automatically deleted. Such as: setting Cookies is valid until April 1, 2004, when it will be automatically deleted. If Cookies are not setting a validity period, its life cycle from the start to open the browser, close the browser to the end of the life cycle will end after each run, next run will start again.
- **2.Domain attribute:** This attribute defines the uniqueness of Cookies data transmission. If only when transferring certain Cookies to _blank "> Sohu home page, you can use the following code:

- **3.Path property:** Defines Cookies are issued only to the specified path request, if the Path property is not set, the default path applications.
- **4.Secure attribute:** Specifies Cookies can be read by users.
- **5, Name = Value: Cookies** are key-value pairs to set and retrieve.

Ruby processing Cookies

You can create an object called cookie and store text messages, send the information to the browser, call CGI.out set cookie header:

```
#!/usr/bin/ruby

require "cgi"
cgi = CGI.new("html4")
cookie = CGI::Cookie.new('name' => 'mycookie',
                         'value' => 'Zara Ali',
                         'expires' => Time.now + 3600)
cgi.out('cookie' => cookie) do
   cgi.head + cgi.body { "Cookie stored" }
end
```

Then we go back to this page, and look for cookie values as follows:

```
#!/usr/bin/ruby

require "cgi"
cgi = CGI.new("html4")
cookie = cgi.cookies['mycookie']
cgi.out('cookie' => cookie) do
   cgi.head + cgi.body { cookie[0] }
end
```

CGI :: Cookie object contains an instance of the following parameters:

parameter	description
name	It specifies the name of the cookie.

value	The predetermined value of the cookie.
expire	Provisions of the cookie.
path	Provisions cookie server path.
domain	Provisions of cookie domain.
secure	Specifies whether connections to transfer cookie over a secure HTTPS.