CGI scripts in Python and mod_python

Anna Fałek, Michał Bajer

Common Gateway Interface

Common Gateway Interface (CGI) is a standard environment for web servers to interface with executable programs installed on a server that generate web pages dynamically. Such programs are known as CGI scripts or simply CGIs; they are usually written in a scripting language, but can be written in any programming language.

How static Web pages are served

Each Web server runs HTTP server software, which responds to requests from Web browsers. Generally, the HTTP server has a directory (folder), which is designated as a document collection — files that can be sent to Web browsers connected to this server. For example, if the Web server has the domain name **example.com**, and its document collection is stored at **/usr/local/apache/htdocs** in the local file system, then the Web server will respond to a request for **http://example.com/index.html** by sending out the (pre-written) file at **/usr/local/apache/htdocs/index.html**.

How CGI serve pages

CGI extends this system by allowing the owner of the Web server to designate a directory within the document collection as containing executable scripts (or binary files) instead of pre-written pages; this is known as a CGI directory. For example, **/usr/local/apache/htdocs/cgi-bin** could be designated as a CGI directory on the Web server. If a Web browser requests a URL that points to a file within the CGI directory (e.g., **http://example.com/cgi-bin/printenv.pl**), then, **instead of simply sending that file (/usr/local/apache/htdocs/cgi-bin/printenv.pl) to the Web browser, the HTTP server runs the specified script and passes the output of the script to the Web browser.** That is, anything that the script sends to standard output is passed to the Web client instead of being shown on-screen in a terminal window.

CGI Advantages

- CGI scripts are portable and work on wide variety of web servers and platforms.
- They are language independent. You can write CGI in any language you want, including Python, Perl, PHP, C++, Java, C#, C.
- They can perform both simple and very complex and complicated tasks.

CGI Disadvantages

The fundamental architectural issue with CGI is that each HTTP request requires the server to start a new process. This affects performance in a number of ways:

- It's expensive to start the process, as the OS pages in the program, sets up the process, etc.
- Resources can not be shared across requests, so that any DB connections, etc. have to be set up with each request
- User session state can not be preserved in memory, so it has to be persisted with each request

Configuring Apache to run CGI

In order to get your CGI programs to work properly, you'll need to have Apache configured to permit CGI execution. There are several ways to do this.

First open your Apache conf file:

/etc/httpd/conf/httpd.conf

The ScriptAlias

The ScriptAlias directive tells Apache that a particular directory is set aside for CGI programs. Apache will assume that every file in this directory is a CGI program, and will attempt to execute it, when that particular resource is requested by a client.

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
```

Explicitly using Options to permit CGI execution

You could explicitly use the Options directive, inside your main server configuration file, to specify that CGI execution was permitted in a particular directory:

```
<Directory /usr/local/apache2/htdocs/somedir>
      Options +ExecCGI
</Directory>
AddHandler cgi-script .cgi .py
```

Example CGI script

```python
#!/usr/bin/python3

import cgi, cgitb

# DEBUG MODE ON
cgitb.enable()

# HTTP HEADER PART
print("Content-type:text/html")
print("Expires:Tuesday, 31-Dec-2007 23:12:40 GMT")
print()

print("<h1>Hello World!</h1>")
if 5 < 10:
        print("<p>five is really smaller than ten!!</p>")

# SOME DEBUG INFO
print("<hr />")
cgi.print_environ()
```

Passing POST data to the script

```html
<!DOCTYPE HTML>
<html>

<head>
<meta charset="utf-8">
<title>POST</title>
</head>

<body>
<form action="/cgi-bin/test.py" method="post">
        First Name: <input type="text" name="first_name"><br />
        Last Name: <input type="text" name="last_name" />

        <input type="submit" value="Submit" />
</form>
</body>

</html>
```

Retrieving POST/GET data in the script

```python
#!/usr/bin/python3

import cgi, cgitb
from os import environ

# DEBUG MODE ON
cgitb.enable()

# HTTP HEADER PART
print("Content-type:text/html")
print()

# HTTP BODY
form = cgi.FieldStorage()

first_name = form.getvalue('first_name')
last_name  = form.getvalue('last_name')

print("<h1>Hello " + first_name + " " + last_name + "</h1>")
print("<p>METHOD: " + environ['REQUEST_METHOD'] + "</p>")
```

FieldStorage Class

This class provides naming, typing, files stored on disk, and more. At the top level, it is accessible like a dictionary, whose keys are the field names.

- **getvalue**(self, key, default=None) Dictionary style get() method, including 'value' lookup.
- **getfirst**(self, key, default=None) Return the first value received.
- **getlist**(self, key) Return list of received values.
- **keys**(self) Dictionary style keys() method.
- **has_key**(self, key) Dictionary style has_key() method.

More at: http://epydoc.sourceforge.net/stdlib/cgi.FieldStorage-class.html

Cookies management example

```
#!/usr/bin/python3

import cgi, cgitb
from os import environ

# DEBUG MODE ON
cgitb.enable()

# HTTP HEADER PART
print('Set-Cookie: test="Hello world";')
print("Content-type:text/html")
print()

# HTTP BODY

print("<h1>Cookies</h1>")

if 'HTTP_COOKIE' in  environ:
        for cookie in environ['HTTP_COOKIE'].split(';'):
                (key, value ) = cookie.split('=');
                print("<p>" + key + " ---> " + value + "</p>")
```

CGI.py module

https://docs.python.org/3.4/library/cgi.html

mod_python

General information

Mod_python is an Apache module that embeds the Python interpreter within the server. With mod_python you can write web-based applications in Python that will run many times faster than traditional CGI and will have access to advanced features such as ability to retain database connections and other data between hits and access to Apache internals.

What is Mod_python?

-Apache module that embeds the Pyhton interpreter

-A handler of Apache's request processing phrases, allowing for any phase to be implemented in Python.
-An interface to a subset of the Apache API.
-A collection of tools for developing web applications

Python interpreter

Python code executes directly within the Apache server, eliminating any need to spawn external processes or run additional servers. At the same time, mod_python takes advantage of Apache's highly optimized ability to accept and process incoming requests. The result is a platform that can process requests faster than any other currently available web application framework for Python.

Ability to handle request phases, filters and connections

Mod_python provides the ability to register for any phase and write the processing function in Python. In addition to phases, mod_python also allows for filters to be implemented in Python. Lastly, mod_python allows you to create connection handlers. A connection handler handles the connection from the moment it is received, allowing you to bypass Apache's HTTP handling.

Interface to Apache API

The interface to Apache API can be used to retrieve various kinds of server information and use internal server facilities. Available server information includes typical data available to CGI programs as well as various interesting bits such as number of bytes sent, server document root directory, the phase being processed, the file handler of the file being requested, and more.

Collection of Tools

Early versions of mod_python provided few tools for web development.Luckily, latest version introduces native handling of cookies, including support for cryptographic signing of cookie data (using HMAC), as well as the ability to marshal (serialize) simple objects into cookie values.

Handlers

A "handler" is an internal Apache representation of the action to be performed when a file is called. Generally, files have implicit handlers, based on the file type. Normally, all files are simply served by the server, but certain file types are "handled" separately.

Mod_python handlers

Functionality that takes place in handlers may include authenticating a user, invoking a cgi script, getting the server's status, etc. mod_python allows you to tap into any handler used by Apache. mod_python also provides a few standard handlers to help you with some common tasks.
-Publisher handler
-CGI Handler
-Custom Handlers
-Python Server Pages (PSP)

Publisher Handler

The Publisher handler allows access to functions and variables within a module via URLs.In order to use the publisher handler, you have to something like this to your Apache configuration:

```
<Directory /usr/local/apache2/htdocs/PublisherExample>
     AddHandler mod_python .py
     PythonDebug On
     PythonHandler mod_python.publisher
</Directory>
```

CGI Handler and Custom Handlers

CGI Handler emulates a CGI environment from within mod_python, allowing you to migrate CGI based Python applications to mod_python.

```
SetHandler mod_python
PythonHandler mod_python.cgihandler
```

Using a standard handler may be appropriate in many scenarios, but sometimes you might find it more appropriate to write your own handler.

```
<Directory /usr/local/apache2/htdocs/CustomExample>
     AddHandler mod_python .py
     PythonDebug On
     PythonHandler customexample
</Directory>
```

Python Server Pages

Python Server Pages allow you to inline Python code in HTML (or any other kind of document) as you would if you were using PHP, ASP (Active Server Pages), JSP (Java Server Pages) or something similar.

```
<Directory /usr/local/apache2/htdocs/PSPExample>
    AddHandler mod_python .psp
     PythonHandler mod_python.psp
     PythonDebug On
</Directory>
```

P

SP

```
<html>
<head>
    <title>Python Server Pages
(PSP)</title>
</head>
<body>
<%
import time
%>
Hello world, the time is: <%=time.
strftime("%Y-%m-%d, %H:%M:%S")%>
</body>
</html>
```

Publisher Handler

```
from time import strftime,
localtime

def publisher_example(req):
     req.content_type =
'text/html'
     time_str = strftime("%a %b %d
%H:%M:%S %Y", localtime())
     message = "<h1>Hello from
mod_python!</h1>"
     message += "<p>The time on
this server is %s</p>" %
(time_str)
     return message
```