Introduktion till mod_python och HTMLgen

(Eller: Bah!)

Erik Forsberg

forsberg@lysator.liu.se

Innehåll

- Inledning: CGI i Python
- Det vanliga sättet, cgi-modulen och #!
- Det mindre vanliga sättet, fastcgi.
- Ett jätteintegrerat sätt, Zope
- Ett annat mindre vanligt sätt, mod_python
- (H)u(T)tu(M)ul(L)e-generering med python

CGI i Python

- CGI är inte Perl :-)
- Skriv ut Lite headers (Content-Type framförallt) och sen vad du vill skriva.
- Kör ditt skript som vanligt med #!/usr/bin/env python
  - Minimalt exempel:

```
#!/usr/bin/env/python
print "Content-Type: text/plain\n"
print "Hello, World!"
```

- På Lysator: Kalla saken för *.cgi och låt det vara exekverbart, samt ha rätt path till Python..

Det vanliga sättet, cgi-modulen och #!

- Exempel

```
#!/usr/bin/env python
import cgi
form = cgi.FieldStorage()
if not form.has_key("name"):
   print "<H1>Error</H1>"
   print "Please fill in the name"
   return
print "<p>name:", form["name"].value
...further form processing here...
```

- Ett gäng funktioner för att parsa indata på ett mer "kontrollerat" sätt finns.
- Se referensen.

Det mindre vanliga sättet, fastcgi.

- En process som anropas via en socket från oebbservern
- Du slipper spawna process för varje anrop.
- Det är betydligt enklare att hålla i ett state
- Öppen standard. Går att köra på flera oebbservrar.
- Exempel på applikation: WebKOM
- http://www.fastcgi.com

Ett jätteintegrerat sätt, Zope

- Ämne för en framtida UppLYSning (?)
- http://www.zope.org/

Ett annat mindre vanligt sätt, mod_python

- Modul för Apache
- Varje Apacheserver har en Pythoninterpretator i sig
- Process behöver inte spawnas för varje anrop.
- Databasanslutningar etc. och vissa data finns kvar mellan anrop
- http://www.modpython.org

Installation av mod_python

- Kompileras och installeras som alla andra Apache-moduler.
- Finns som Debian-paket.
- Fungerar med Python 2.1

Konfiguration av mod_python

- Enkelt exempel

```
# srm.conf
Alias /modpyex/ /home/erik/dev/python/modpyex/
<Directory "/home/erik/dev/python/modpyex/">
 Options -Indexes
 SetHandler python-program
        PythonHandler ex
</Directory>
```

Ger ungefär samma funktionalitet som ett CGI-script.

Exempel på script

```
from mod_python import apache
def handler(req):
    req.contenttype = "text/plain"
    req.send_http_header()
    req.write("Hello, world!")
    return apache.OK
```

Objekt

- Requestobjekt
  - Skickas med som argument till handlern vid varje request
  - Metoder och medlemmar som ger information om requesten.
    - add_handler() kan dynamiskt bestämma vad som ska hända efter den nuvarande handlern.
    - get_basic_auth_pw() hämtar info från ’Basic authentication’
    - get_remote_host() hämtar info om klienten (IP/hostnamn)
    - read() läser data från klienten (ex.vis POST)
    - method innehåller metoden som användes vid requesten.
    - headers_in innehåller alla headers som klienten sände
    - unparsed_uri innehåller oparsad URI
    - uri innehåller path-delen av URI.
    - connection är ett Connection-objekt
    - server är ett Server-objekt.

Requestobjekt, fortsättning

- Metoder och medlemmar som används för att skicka tillbaka info till klienten.
  - send_http_header() skickar headers. Måste köras före write()
  - write() skriver data till klienten.

  - headers_out innehåller alla headers som kommer att sändas när send_http_header() körs.
  - content_type sätter man till vad datat har för typ, innan send_http_header()

Objekt, fortsättning

- Connectionobjekt
  - Innehåller information om anslutningen som används för just den här requesten
    - local_addr, remote_addr, keepalives, etc..
    - Mer finns, se mod_pythons dokumentation.
- Serverobjekt
  - Refererar till servern som anropet kom till
    - register_cleanup() registrerar ett anrop som körs när just det här serverbarnet dör.
    - Info om serverns hostname, administratör, port, keep-alive-inställningar etc..
    - Se mod_pythons dokumentation.

Handlers

- Ger olika hakar i requestens gång
  - PythonPostReadRequestHandler
  - PythonTransHandler
  - PythonHeaderParserHandler
  - PythonAccessHandler
  - PythonAuthenHandler
  - PythonTypeHandler
  - PythonFixupHandler
  - PythonHandler
  - PythonInitHandler
  - PythonLogHandler
  - PythonCleanupHandler
  - Standardhandlers

Handlers, fortsättning

- PythonPostReadRequestHandler
  - Körs precis efter att requesten blivit läst
  - Används ex.vis för att avgöra vad som ska köras baserat på headers
  - Typisk sak att göra: Sätt en handler med add_handler()
  - Kan ej existera i <Directory> <Location> <File> eller .htaccess
- PythonTransHandler
  - Körs efter PythonPostReadRequstHandler
  - Används för att översätta URI till riktigt filnamn
  - Kan ej existera i <Directory> <Location> <File> eller .htaccess
- PythonHeaderParserHandler
  - Körs efter PythonTransHandler
  - Ungefär som PythonPostReadRequestHandler, men nu vet vi vart requesten ska.

Handlers, fortsättning

- PythonAccessHandler
  - Körs efter PythonHeaderParserHandler
  - Används ex.vis för att kolla om ett visst IP är tillåtet för just den här requesten.
  - Fånigt Exempel

```
 def accesshandler(req):
     apache.log_error("PythonAccessHandler now running, remote IP = %s" %
req.connection.remote_ip)
     if "207.46." == req.connection.remote_ip[0:7]: # Disallow some MS
employees..
         return apache.HTTP_FORBIDDEN
     else:
         return apache.OK
```

Handlers, fortsättning

- PythonAuthenHandler
  - Körs efter PythonAccessHandler
  - Avgör om användaren i andra änden är tillåten, med Basic Auth.
  - Kräver givetvis att Authentifiering är påslaget i Apache.
  - Exempel:

```
def authenhandler(req):
    apache.log_error("PythonAuthenHandler now running")
    pw = req.get_basic_auth_pw()
    user = req.connection.user
    if user == "RING" and pw == "IKEA":
        return apache.OK
    else:
        return apache.HTTP_UNAUTHORIZED
```

  - Smartare är givetvis att glo i nån slags databas efter rätt data.

Handlers, fortsättning

- PythonTypeHandler
  - "This routine is called to determine and/or set the various document type information bits, like Content-type (via r->content_type), language, et cetera"
  - Konstig!

Handlers, fortsättning

- PythonFixupHandler
  - Körs före content-handlern, men efter authentifiering.
  - Kan användas för att fixa headers eller motsvarande så att de följer nå gon slags standard.
  - Kunde inte komma på något bra exempel :-)

Handlers, fortsättning

- PythonHandler
  - Används för att ta göra nått vettigt :-)
  - GrundTODO:
    - Läs vad som ska göras i req
    - Skicka headers med send_http_header()
    - Skriv saker med req.write()
    - Returnera rätt returvärde
      - apache.OK
      - apache.HTTP_INTERNAL_SERVER_ERROR
      - apache.HTTP_NOT_FOUND
      - ...
    - Klart!

PythonHandler, fortsättning

- Ett fakultetsexempel:

```
def handler(req):
    form = util.FieldStorage(req)
    apache.log_error("PythonHandler now running")
    req.contenttype = "text/html"
    req.send_http_header()
    if form.has_key("fakof"):
        fakof = int(form["fakof"])
        req.res = 1
        while fakof:
            req.res*=fakof
            fakof-=1
        req.fakof = int(form["fakof"])
        req.write("fak(%d) = %d" % (req.fakof, req.res))
    else:
        req.write("Please give me a value as the parameter 'fakof'")
    return apache.OK
```

Handlers, fortsättning

- PythonLogHandler
  - Loggar saker efter svaret på requesten
  - Lagom fånigt exempel:

```
def loghandler(req):
    try:
        apache.log_error("Calculated fak(%d) which is %d" % (req.fakof,
req.res))
    except AttributeError:
        apache.log_error("No value was submitted for calculation")
    return apache.OK
```

    - Notera vidareskickning av data i req

```
[Sat Sep 22 16:29:09 2001] [error] Calculated fak(12) which is 479001600
```

Handlers, fortsättning

- PythonCleanupHandler
  - Anropas absolut sist, innan requestobjektet förstörs av apache
  - Returvärdet spelar ingen roll, till skillnad från alla andra Handlers.
  - Används kanske för att låsa upp lås eller motsv.

Standardhandlers

- CGI Handler - emulerar CGI-miljön för att köra gamla script under mod_python
- Zpublisher - knyckt från Zope.

Standardhandlers, fortsättning

- *Publisher handler - smidigt sätt att göra många funktioner tillgängliga*
  - Givet följande Apache-konfiguration:

```
Alias /modpypub/ /home/erik/dev/python/modpypub/
<Directory "/home/erik/dev/python/modpypub/">
 SetHandler python-program
 PythonHandler mod_python.publisher
</Directory>
```

  - ..och följande kod i form.py

```
def say(req, what="Nothing"):
    return "I am saying %s" % what
```

  - ..så kommer en request till /modpypub/form/say ge..

```
"I am saying Nothing"
```

Andra Apachekonfigurationsdirektiv

- PythonEnablePdb
  - debugga moduler i Pdb
- PythonDebug
  - Skicka ofångade Exceptions även till klienten
- PythonImport
  - importera moduler en gång för alla.
  - OBS: Körs innan all konfiguration är klar. Undvik.

Apachedirektiv, fortsättning..

- PythonInterpPerDirectory
  - En subinterpretator per directory
  - Default: En per virtuell server
- PythonInterpPerDirective
  - En subinterpretator per Directive
- PythonInterpreter
  - bestäm vilken subinterpretator som ska användas

Apachedirektiv, fortsättning..

- PythonHandlerModule
  - minimera antalet Handler-Directives
- PythonAutoReload - kolla om filen har ändrats sedan sista requesten
  - Typiskt bra vid utveckling. Ändra i filen, gör reload och se resultatet.
  - Funkar inte för moduler, men det går att gå runt. Såhär:

```
DEBUG = 1
if DEBUG:
    modpydb = reload(modpydb)
```

  - Nu laddas modpydb om varje gång en request utförs. Stängs lämpligen av i släppt kod.

Apachedirektiv, fortsättning.. (forts.)

- PythonOptimize - slå av/på optimering
- PythonOption - ge parametrar som kan hämtas med req.get_options()
  - Användbar ex.vis vid många virtuella servrar
- PythonPath - sätt Pythons sys.path
  - OBS: Byter ut den.

```
PythonPath "sys.path+['/mydir']"
```

mod_python.util

- FieldStorage

```
form = util.FieldStorage(req)
form.has_key("fakof"):
form["fakof"]
```

  - som cgi.FieldStorage, fast med dictionaryinterface
- Field
  - Används i princip bara vid filuppladdningar
- parse_qs
  - Tolka en application/x-www-form-urlencoded, returnera dictionary
- parse_qsl
  - Tolka en application/x-www-form-urlencoded, returnera lista name/value.

Ett exempel på lagom bra databasaccess

```
import MySQLdb
import string
from constants import DB, USER, PASS, UNIX_SOCKET

def _db_login(relogin = 0):
    """Login to the database
    """

    global DB_CONN

    if relogin:
        DB_CONN = MySQLdb.connect(db=DB, user=USER, passwd=PASS,
                                  unix_socket=UNIX_SOCKET)
        return DB_CONN
    else:
        try:
            d = DB_CONN
            return d

        except NameError:
            DB_CONN = MySQLdb.connect(db=DB, user=USER, passwd=PASS,
                                      unix_socket=UNIX_SOCKET)
            return DB_CONN
```

modpydb.py fortsättning..

```
def run_sql(sql, param=None, n=0, with_desc=0):
    """ Runs SQL on the server and returns result
    """

    db = _db_login()

    if param:
        param = tuple(param)

    try:
        cur = db.cursor()
        rc = cur.execute(sql, param)

    except:
        # it's possible that mysql connection
        # timed out. Try one more time.
        db = _db_login(relogin = 1)
        cur = db.cursor()
        rc = cur.execute(sql, param)
```

modpydb.py fortsättning..

```
        if string.upper(string.split(sql)[0]) in ("SELECT", "SHOW", "DESC",
"DESCRIBE"):
            if n:
                recset = cur.fetchmany(n)
            else:
                recset = cur.fetchall()

            if with_desc:
                return recset, cur.description
            else:
                return recset
        else:
            return rc


    def insert_id():
        """ Get the insert ID of the last insert
        """
        db = _db_login()
        return db._db.insert_id()
```

- En databasöppning per apacheprocess.
- Databaskopplet återanvänds

(POOM)

- En klass definieras för varje möjlig sida/kommando.
- Varje klass har en metod "response"

```
dispatchers = {"mainpage"    : MainPageHandler}

def handler(req):
    cmdpart = commandpart(req.uri)
    try:
        response_type = dispatchers[cmdpart]
        action = response_type(req)
        return action.response()
    except:
```

Exempel, fortsättning..

```
            import traceback
            timetext = time.strftime("%y%m%d-%H%M",
time.localtime(time.time()))
            f = open(join(constants.TRACEBACKDIR, timetext), 'w')
            f.write("Time of exception: %s\n\n" % timetext)
            traceback.print_exc(file=f)
            f.write("\n")
            f.write("Remote IP: %s\n" % req.connection.remote_ip)
            f.write("Request  : %s\n" % req.the_request)
            f.close()
            f = open(join(constants.TRACEBACKDIR, timetext), 'r')
            ret = Error(req).response(title="Exception",
                                      errortext = "<PRE>%s</PRE>" % f.read())
            f.close()
            return ret


    class MainPageHandler():
          def __init__(self, req):
        self.req = req
          def response(self):
       ...
```

HTMLgen

(En oerhört användarfientlig genomgång av en skojig modul)

HTML-generering med HTMLgen

- Generering av HTML
  - Från template..
  - Med färdiga dokumentklasser..
  - Eller bara de bitar du behöver.
- Dokumentation:

  http://starship.python.net/crew/friedrich/HTMLgen/html/main.h
- Download: http://starship.python.net/lib.html
- RPM: http://www.lysator.liu.se/lyskom/klienter/webkom/
- ’apt-get install htmlgen’

HTMLgen med templates - TemplateDocument

- Exempeltemplate

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
     "http://www.w3.org/TR/REC-html40/loose.dtd">
<HEAD>
<LINK rel="stylesheet" type="text/css" media="screen"
href="/fufreg/fufreg.css">
<TITLE>Test av template</TITLE>
</HEAD>
<BODY>
{foo}
</BODY>
</HTML>
```

- Exempelkod för generering

```
from HTMLgen import *
T = TemplateDocument("foo.tmpl.html")
T.substitutions["foo"] = "Ja, eller en kaktus"
T.write("foo.html")
```

..med templates, fortsättning.

- Resultat

```
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
         "http://www.w3.org/TR/REC-html40/loose.dtd">
    <HEAD>
    <LINK rel="stylesheet" type="text/css" media="screen"
href="/fufreg/fufreg.css">
    <TITLE>Test av template</TITLE>
    </HEAD>
    <BODY>
    Ja, eller en kaktus
    </BODY>
```

HTMLgen med templates - AutoTemplateDocument

- Samma fil kan användas som både template och utfil.
- Exempelkod, samma infil som tidigare:

```
from HTMLgen import *
AT = AutoTemplateDocument("foo.tmpl.html")
AT.substitutions["foo"] = "Våld och vaselin"
AT.write("foo.html")
```

- foo.html ser nu ut såhär:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
     "http://www.w3.org/TR/REC-html40/loose.dtd">
<HEAD>
<LINK rel="stylesheet" type="text/css" media="screen"
href="/fufreg/fufreg.css">
<TITLE>Test av template</TITLE>
</HEAD>
<BODY>
<!--{foo}Begin-->Våld och vaselin<!--{foo}End-->
</BODY>
</HTML>
```

AutoTemplateDocument, fortsättning

- Filen kan åter läsas in i ett AutoTemplateDocument och skrivas ut igen.
  - Behovet av ett extra template är alltså eliminerat.

HTMLgen med färdiga dokumentklasser

- BasicDocument
- FramesetDocument
- SimpleDocument
- SeriesDocument

HTMLgen med BasicDocument

- Enkel klass med bara de enklaste HTML-funktionerna definierade
- Exempelkod:

```
from HTMLgen import BasicDocument
import HTMLcolors
B = BasicDocument(title = "A test document",
    bgcolor = HTMLcolors.BLACK,
    textcolor = HTMLcolors.RED,
    linkcolor = HTMLcolors.OLIVE)
B.append("Foo!")
print B
```

..med BasicDocument, fortsättning

- Resultat

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>

<!-- This file generated using Python HTMLgen module. -->
<HEAD>
  <META NAME="GENERATOR" CONTENT="HTMLgen 2.2.2">
        <TITLE>A test document</TITLE> </HEAD>
<BODY BGCOLOR="#000000" TEXT="#EE0000" LINK="#999966">
Foo!

</BODY> </HTML>
```

HTMLgen med FramesetDocument

- För dokument med Frames.
- Addera instanser av Frameset
- Addera instanser av Frame till Framset

HTMLgen med SimpleDocument

- Använder en resursfil för att sätta diverse dokumentattribut

```
# Resource file used by the Document class of the HTML module
from HTMLcolors import *
stylesheet = 'HTMLgen.css'
author = '2000 Erik Forsberg'
email = 'forsberg@lysator.liu.se'
bgcolor = WHITE
textcolor = BLACK
linkcolor = RED
vlinkcolor = RED6
banner = None
logo = ('Buzz.gif', 56, 51)
blank = (blank.gif', 66, 22)
prev = ('back.gif', 66, 22)
next = ('next.gif', 66, 22)
top = ('top.gif', 66, 22)
home = ('home.gif', 66, 22)
gohome = None
background = '../image/bg-dots.gif'
```

Exempel med ovanst. resursfil:

```
from HTMLgen import *

sd = SimpleDocument(resource="foo.RC")
k = Container()
k.append(Heading(1, "Header"))
k.append("Lite text..")
k.append(HR())
sd.append(k)
print sd
sd.write("/tmp/foo.SimpleDocument.html")
```

..ger resultatet..

```
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
    <HTML>

    <!-- This file generated using Python HTMLgen module. -->
    <HEAD>
      <META NAME="GENERATOR" CONTENT="HTMLgen 2.2.2">
            <TITLE></TITLE>

     <LINK rel=stylesheet href="HTMLgen.css" type=text/css
title="HTMLgen.css">
    </HEAD>
    <BODY BGCOLOR="#FFFFFF" BACKGROUND="../image/bg-dots.gif" TEXT="#000000"
LINK="#EE0000" VLINK="#990000">
    <H1>Header</H1>

    Lite text..
    <HR>



    </BODY> </HTML>
```

HTMLgen med SeriesDocument

- Påbyggd SimpleDocument med header och footer mm.

Övriga HTMLgen

- Meta
- Href
- Name
- MailTo
- List, 4 typer
- Form, med alla inputsorter
- Table, två sorter
- Alla övriga HTML-taggar har varsin klass.

Tabellgenerering

- Två sätt
  - Table
  - TableLite

Tabellgenerering - Table

- Generera tabeller från Python-listor
- Exempel:

```
from HTMLgen import *

tb = [["13-12-1", 14000], ["13-12-2", 15600]]
t = Table("TableCaption", heading=["Kontonummer", "Belopp"], body=tb)
print t
```

..ger resultatet..

```
    <A NAME="TableCaption"></A>
    <P><TABLE border=2 cellpadding=4 cellspacing=1 width="100%">
    <CAPTION align=top><STRONG>TableCaption</STRONG></CAPTION>
    <TR Align=center> <TH ColSpan=1>Kontonummer</TH><TH
ColSpan=1>Belopp</TH></TR>
    <TR><TD Align=left >13-12-1</TD>
    <TD Align=left >14000</TD>
    </TR>
    <TR><TD Align=left >13-12-2</TD>
    <TD Align=left >15600</TD>
    </TR>
    </TABLE><P>
```

Tabellgenerering - TableLite

```
from HTMLgen import *

tl = TableLite()
tl.append(TH(TD("Bankkonto"), TD("Belopp")))
tl.append(TR(TD("13-21-1"), TD(133434)))
print tl
```

..ger output..

```
<TABLE><TH><TD>Bankkonto</TD><TD>Belopp</TD></TH><TR><TD>13-21-1</TD><TD>133434</
    </TABLE>
```

Container

- Är en klass vari man kan samla andra klasser.

Exemplet..

```
from HTMLgen import *
import HTMLcolors


k = Container()
k.append(Heading(1, "Header"))
k.append("Lite text..")
k.append(HR())


print k
```

..ger output..

```
<H1>Header</H1>


Lite text..
<HR>
```

FIN

http://www.lysator.liu.se/~forsberg/

forsberg@lysator.liu.se