Wikiprint Book

Title: Trac and mod_wsgi

Subject: Ecopath Developer Site - TracModWSGI

Version: 2

Date: 2024-04-24 00:48:47

Table of Contents

Trac and mod_wsgi

Important note: *Please use either version 1.3 or 2.3 or later of `mod_wsgi`. Version 2.0 has problems with downloading attachments (see ?#7205).*

?mod_wsgi is an Apache module for running WSGI-compatible Python applications directly on top of Apache:

> The mod_wsgi adapter is an Apache module that provides a WSGI compliant interface for hosting Python based web applications within Apache. The adapter is written completely in C code against the Apache C runtime and for hosting WSGI applications within Apache provides significantly better performance than using existing WSGI adapters for mod_python or CGI.

It is already possible to run Trac on top of mod_wsgi. This can be done by writing the following application script, which is just a Python file, though usually saved with a .wsgi extension).

```
import os

os.environ['TRAC_ENV'] = '/usr/local/trac/mysite'
os.environ['PYTHON_EGG_CACHE'] = '/usr/local/trac/mysite/eggs'

import trac.web.main
application = trac.web.main.dispatch_request
```

The `TRAC_ENV` variable should naturally be the directory for your Trac environment (if you have several Trac environments in a directory, you can also use `TRAC_ENV_PARENT_DIR` instead), while the `PYTHON_EGG_CACHE` should be a directory where Python can temporarily extract Python eggs. For clarity, you should give this file a `.wsgi` extension. You should probably put the file in it's own directory, since you will open up its directory to Apache. You can create a .wsgi files which handles all this for you by running the TracAdmin command `deploy`.

If you have installed trac and eggs in a path different from the standard one you should add that path by adding the following code on top of the wsgi script:

```
import site
site.addsitedir('/usr/local/trac/lib/python2.4/site-packages')
```

Change it according to the path you installed the trac libs at.

After you've done preparing your wsgi-script, add the following to your httpd.conf.

```
WSGIScriptAlias /trac /usr/local/trac/mysite/apache/mysite.wsgi

<Directory /usr/local/trac/mysite/apache>
   WSGIApplicationGroup %{GLOBAL}
   Order deny,allow
   Allow from all
</Directory>
```

Here, the script is in a subdirectory of the Trac environment. In order to let Apache run the script, access to the directory in which the script resides is opened up to all of Apache. Additionally, the `WSGIApplicationGroup` directive ensures that Trac is always run in the first Python interpreter created by mod_wsgi; this is necessary because the Subversion Python bindings, which are used by Trac, don't always work in other subinterpreters and may cause requests to hang or cause Apache to crash as a result. After adding this configuration, restart Apache, and then it should work.

To test the setup of Apache, mod_wsgi and Python itself (ie. without involving Trac and dependencies), this simple wsgi application can be used to make sure that requests gets served (use as only content in your .wsgi script):

```
def application(environ, start_response):
        start_response('200 OK',[('Content-type','text/html')])
        return ['<html><body>Hello World!</body></html>']
```

See also the mod_wsgi ?installation instructions for Trac.

For troubleshooting tips, see the mod_python troubleshooting section, as most Apache-related issues are quite similar, plus discussion of potential ?application issues when using mod_wsgi.

Trac with PostgreSQL

When using the mod_wsgi adapter with multiple Trac instances and PostgreSQL (or MySQL?) as a database back-end the server can get a lot of open database connections. (and thus PostgreSQL processes)

A workable solution is to disabled connection pooling in Trac. This is done by setting poolable = False in trac.db.postgres_backend on the PostgreSQLConnection class.

But it's not necessary to edit the source of trac, the following lines in trac.wsgi will also work:

```
import trac.db.postgres_backend
trac.db.postgres_backend.PostgreSQLConnection.poolable = False
```

Now Trac drops the connection after serving a page and the connection count on the database will be kept minimal.

Getting Trac to work nicely with SSPI and 'Require Group'

If like me you've set Trac up on Apache, Win32 and configured SSPI, but added a 'Require group' option to your apache configuration, then the SSPIOmitDomain option is probably not working. If its not working your usernames in trac are probably looking like 'DOMAIN\user' rather than 'user'.

This WSGI script 'fixes' things, hope it helps:

```
import os
import trac.web.main

os.environ['TRAC_ENV'] = '/usr/local/trac/mysite'
os.environ['PYTHON_EGG_CACHE'] = '/usr/local/trac/mysite/eggs'

def application(environ, start_response):
   if "\\" in environ['REMOTE_USER']:
       environ['REMOTE_USER'] = environ['REMOTE_USER'].split("\\", 1)[1]
   return trac.web.main.dispatch_request(environ, start_response)
```

See also: TracGuide, TracInstall, FastCGI, ModPython, ?TracNginxRecipe