Trac and mod_python

Trac supports ?mod_python, which speeds up Trac's response times considerably, especially compared to CGI, and permits use of many Apache features not possible with tracd/mod_proxy.

These instructions are for Apache 2; if you are still using Apache 1.3, you may have some luck with ?TracModPython2.7.

Simple configuration

If you just installed mod_python, you may have to add a line to load the module in the Apache configuration:

```
LoadModule python_module modules/mod_python.so
```

Note: The exact path to the module depends on how the HTTPD installation is laid out.

On Debian using apt-get

```
apt-get install libapache2-mod-python libapache2-mod-python-doc
```

(Still on Debian) after you have installed mod_python, you must enable the modules in apache2 (equivalent of the above Load Module directive):

```
a2enmod mod_python
```

On Fedora use, using yum:

```
yum install mod_python
```

You can test your mod_python installation by adding the following to your httpd.conf. You should remove this when you are done testing for security reasons. Note: mod_python.testhandler is only available in mod_python 3.2+.

```
<Location /mpinfo>
  SetHandler mod_python
  PythonInterpreter main_interpreter
  PythonHandler mod_python.testhandler
</Location>
```

A simple setup of Trac on mod_python looks like this:

```
<Location /projects/myproject>
  SetHandler mod_python
  PythonInterpreter main_interpreter
  PythonHandler trac.web.modpython_frontend
  PythonOption TracEnv /var/trac/myproject
  PythonOption TracUriRoot /projects/myproject
</Location>
```

The option **`TracUriRoot`** may or may not be necessary in your setup. Try your configuration without it; if the URLs produced by Trac look wrong, if Trac does not seem to recognize URLs correctly, or you get an odd "No handler matched request to..." error, add the **`TracUriRoot`** option. You will notice that the `Location` and **`TracUriRoot`** have the same path.

The options available are

```
    # For a single project
   PythonOption TracEnv /var/trac/myproject
   # For multiple projects
   PythonOption TracEnvParentDir /var/trac/myprojects
   # For the index of multiple projects
   PythonOption TracEnvIndexTemplate /srv/www/htdocs/trac/project_list_template.html
   # A space delimitted list, with a "," between key and value pairs.
   PythonOption TracTemplateVars key1,val1 key2,val2
```

```
    # Useful to get the date in the wanted order
    PythonOption TracLocale en_GB.UTF8
    # See description above
    PythonOption TracUriRoot /projects/myproject
```

Python Egg Cache

Compressed python eggs like Genshi are normally extracted into a directory named `.python-eggs` in the users home directory. Since apache's home usually is not writable an alternate egg cache directory can be specified like this:

```
PythonOption PYTHON_EGG_CACHE /var/trac/myprojects/egg-cache
```

or you can uncompress the Genshi egg to resolve problems extracting from it.

Configuring Authentication

Creating password files and configuring authentication works similar to the process for CGI:

```
<Location /projects/myproject/login>
 AuthType Basic
 AuthName "myproject"
 AuthUserFile /var/trac/myproject/.htpasswd
 Require valid-user
</Location>
```

Configuration for mod_ldap authentication in Apache is a bit tricky (httpd 2.2.x and OpenLDAP: slapd 2.3.19)

i. You need to load the following modules in Apache httpd.conf

```
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
```

i. Your httpd.conf also needs to look something like:

```
<Location /trac/>
 SetHandler mod_python
 PythonInterpreter main_interpreter
 PythonHandler trac.web.modpython_frontend
 PythonOption TracEnv /home/trac/
 PythonOption TracUriRoot /trac/
 Order deny,allow
 Deny from all
 Allow from 192.168.11.0/24
 AuthType Basic
 AuthName "Trac"
 AuthBasicProvider "ldap"
 AuthLDAPURL "ldap://127.0.0.1/dc=example,dc=co,dc=ke?uid?sub?(objectClass=inetOrgPerson)"
 authzldapauthoritative Off
 require valid-user
</Location>
```

Or the LDAP interface to a Microsoft Active Directory:

```
<Location /trac/>
 SetHandler mod_python
 PythonInterpreter main_interpreter
 PythonHandler trac.web.modpython_frontend
 PythonOption TracEnv /home/trac/
 PythonOption TracUriRoot /trac/
 Order deny,allow
 Deny from all
```

```
 Allow from 192.168.11.0/24
 AuthType Basic
 AuthName "Trac"
 AuthBasicProvider "ldap"
 AuthLDAPURL "ldap://adserver.company.com:3268/DC=company,DC=com?sAMAccountName?sub?(objectClass=user)"
 AuthLDAPBindDN       ldap-auth-user@company.com
 AuthLDAPBindPassword "the_password"
 authzldapauthoritative Off
 # require valid-user
 require ldap-group CN=Trac Users,CN=Users,DC=company,DC=com
</Location>
```

Note 1: This is the case where the LDAP search will get around the multiple OUs, conecting to Global Catalog Server portion of AD (Notice the port is 3268, not the normal LDAP 389). The GCS is basically a "flattened" tree which allows searching for a user without knowing to which OU they belong.

Note 2: Active Directory requires an authenticating user/password to access records (AuthLDAPBindDN and AuthLDAPBindPassword).

Note 3: The directive "require ldap-group ..." specifies an AD group whose members are allowed access.

Setting the Python Egg Cache

If the Egg Cache isn't writeable by your Web server, you'll either have to change the permissions, or point Python to a location where Apache can write. This can manifest itself as a *500 internal server error* and/or a complaint in the syslog.

```
<Location /projects/myproject>
 ...
 PythonOption PYTHON_EGG_CACHE /tmp
 ...
</Location>
```

Setting the PythonPath

If the Trac installation isn't installed in your Python path, you'll have to tell Apache where to find the Trac mod_python handler using the `PythonPath` directive:

```
<Location /projects/myproject>
 ...
 PythonPath "sys.path + ['/path/to/trac']"
 ...
</Location>
```

Be careful about using the PythonPath directive, and *not* `SetEnv PYTHONPATH`, as the latter won't work.

Setting up multiple projects

The Trac mod_python handler supports a configuration option similar to Subversion's `SvnParentPath`, called `TracEnvParentDir`:

```
<Location /projects>
 SetHandler mod_python
 PythonInterpreter main_interpreter
 PythonHandler trac.web.modpython_frontend
 PythonOption TracEnvParentDir /var/trac
 PythonOption TracUriRoot /projects
</Location>
```

When you request the `/projects` URL, you will get a listing of all subdirectories of the directory you set as `TracEnvParentDir` that look like Trac environment directories. Selecting any project in the list will bring you to the corresponding Trac environment.

If you don't want to have the subdirectory listing as your projects home page you can use a

```
<LocationMatch "/.+/">
```

This will instruct Apache to use mod_python for all locations different from root while having the possibility of placing a custom home page for root in your DocumentRoot folder.

You can also use the same authentication realm for all of the projects using a `<LocationMatch>` directive:

```
<LocationMatch "/projects/[^/]+/login">
 AuthType Basic
 AuthName "Trac"
 AuthUserFile /var/trac/.htpasswd
 Require valid-user
</LocationMatch>
```

Virtual Host Configuration

Below is the sample configuration required to set up your trac as a virtual server (i.e. when you access it at the URLs like http://trac.mycompany.com):

```
<VirtualHost * >
   DocumentRoot /var/www/myproject
   ServerName trac.mycompany.com
   <Location />
       SetHandler mod_python
       PythonInterpreter main_interpreter
       PythonHandler trac.web.modpython_frontend
       PythonOption TracEnv /var/trac/myproject
       PythonOption TracUriRoot /
   </Location>
   <Location /login>
       AuthType Basic
       AuthName "MyCompany Trac Server"
       AuthUserFile /var/trac/myproject/.htpasswd
       Require valid-user
   </Location>
</VirtualHost>
```

This does not seem to work in all cases. What you can do if it does not:

- Try using `<LocationMatch>` instead of `<Location>`
- <Location /> may, in your server setup, refer to the complete host instead of simple the root of the server. This means that everything (including the login directory referenced below) will be sent to python and authentication does not work (i.e. you get the infamous Authentication information missing error). If this applies to you, try using a sub-directory for trac instead of the root (i.e. /web/ and /web/login instead of / and /login).
- Depending on apache's `NameVirtualHost` configuration, you may need to use `<VirtualHost *:80>` instead of `<VirtualHost *>`.

For a virtual host that supports multiple projects replace "`TracEnv`" /var/trac/myproject with "`TracEnvParentDir`" /var/trac/

Note: DocumentRoot should not point to your Trac project env. As Asmodai wrote on #trac: "suppose there's a webserver bug that allows disclosure of DocumentRoot they could then leech the entire Trac environment".

Troubleshooting

In general, if you get server error pages, you can either check the Apache error log, or enable the `PythonDebug` option:

```
<Location /projects/myproject>
 ...
 PythonDebug on
</Location>
```

For multiple projects, try restarting the server as well.

Expat-related segmentation faults

This problem will most certainly hit you on Unix when using Python 2.4. In Python 2.4, some version of Expat (an XML parser library written in C) is used, and if Apache is using another version, this results in segmentation faults. As Trac 0.11 is using Genshi, which will indirectly use Expat, that problem can

now hit you even if everything was working fine before with Trac 0.10.

See Graham Dumpleton's detailed ?explanation and workarounds for the issue.

Form submission problems

If you're experiencing problems submitting some of the forms in Trac (a common problem is that you get redirected to the start page after submission), check whether your `DocumentRoot` contains a folder or file with the same path that you mapped the mod_python handler to. For some reason, mod_python gets confused when it is mapped to a location that also matches a static resource.

Problem with virtual host configuration

If the <Location /> directive is used, setting the `DocumentRoot` may result in a *403 (Forbidden)* error. Either remove the `DocumentRoot` directive, or make sure that accessing the directory it points is allowed (in a corresponding `<Directory>` block).

Using <Location /> together with `SetHandler` resulted in having everything handled by mod_python, which leads to not being able download any CSS or images/icons. I used <Location /trac> `SetHandler None` </Location> to circumvent the problem, though I do not know if this is the most elegant solution.

Using .htaccess

Although it may seem trivial to rewrite the above configuration as a directory in your document root with a `.htaccess` file, this does not work. Apache will append a "/" to any Trac URLs, which interferes with its correct operation.

It may be possible to work around this with mod_rewrite, but I failed to get this working. In all, it is more hassle than it is worth. Stick to the provided instructions. 🙂

A success story: For me it worked out-of-box, with following trivial config:

```
SetHandler mod_python
PythonInterpreter main_interpreter
PythonHandler trac.web.modpython_frontend
PythonOption TracEnv /system/path/to/this/directory
PythonOption TracUriRoot /path/on/apache

AuthType Basic
AuthName "ProjectName"
AuthUserFile /path/to/.htpasswd
Require valid-user
```

The `TracUriRoot` is obviously the path you need to enter to the browser to get to the trac (e.g. domain.tld/projects/trac)

Additional .htaccess help

If you are using the .htaccess method you may have additional problems if your trac directory is inheriting .htaccess directives from another. This may also help to add to your .htaccess file:

```
<IfModule mod_rewrite.c>
 RewriteEngine Off
</IfModule>
```

Win32 Issues

If you run trac with mod_python < 3.2 on Windows, uploading attachments will **not** work. This problem is resolved in mod_python 3.1.4 or later, so please upgrade mod_python to fix this.

OS X issues

When using mod_python on OS X you will not be able to restart Apache using `apachectl restart`. This is apparently fixed in mod_python 3.2, but there's also a patch available for earlier versions ?here.

SELinux issues

If Trac reports something like: *Cannot get shared lock on db.lock* The security context on the repository may need to be set:

```
chcon -R -h -t httpd_sys_content_t PATH_TO_REPOSITORY
```

See also ?http://subversion.tigris.org/faq.html#reposperms

FreeBSD issues

Pay attention to the version of the installed mod_python and sqlite packages. Ports have both the new and old ones, but earlier versions of pysqlite and mod_python won't integrate as the former requires threaded support in python, and the latter requires a threadless install.

If you compiled and installed apache2, apache wouldn?t support threads (cause it doesn?t work very well on FreeBSD). You could force thread support when running ./configure for apache, using --enable-threads, but this isn?t recommendable. The best option ?seems to be adding to /usr/local/apache2/bin/ennvars the line

```
export LD_PRELOAD=/usr/lib/libc_r.so
```

Subversion issues

If you get the following Trac Error `Unsupported version control system "svn"` only under mod_python, though it works well on the command-line and even with TracStandalone, chances are that you forgot to add the path to the Python bindings with the PythonPath directive. (The better way is to add a link to the bindings in the Python `site-packages` directory, or create a `.pth` file in that directory.)

If this is not the case, it's possible that you're using Subversion libraries that are binary incompatible with the apache ones (an incompatibility of the `apr` libraries is usually the cause). In that case, you also won't be able to use the svn modules for Apache (`mod_dav_svn`).

You also need a recent version of `mod_python` in order to avoid a runtime error (`argument number 2: a 'apr_pool_t *' is expected`) due to the default usage of multiple sub-interpreters. 3.2.8 *should* work, though it's probably better to use the workaround described in ?#3371, in order to force the use of the main interpreter:

```
PythonInterpreter main_interpreter
```

This is anyway the recommended workaround for other well-known issues seen when using the Python bindings for Subversion within mod_python (?#2611, ?#3455). See in particular Graham Dumpleton's comment in ?#3455 explaining the issue.

Page layout issues

If the formatting of the Trac pages look weird chances are that the style sheets governing the page layout are not handled properly by the web server. Try adding the following lines to your apache configuration:

```
Alias /myproject/css "/usr/share/trac/htdocs/css"
<Location /myproject/css>
   SetHandler None
</Location>
```

Note: For the above configuration to have any effect it must be put after the configuration of your project root location, i.e. `<Location /myproject />`.

HTTPS issues

If you want to run Trac fully under https you might find that it tries to redirect to plain http. In this case just add the following line to your apache configuration:

```
<VirtualHost * >
   DocumentRoot /var/www/myproject
   ServerName trac.mycompany.com
   SetEnv HTTPS 1
   ....
</VirtualHost>
```

Fedora 7 Issues

Make sure you install the 'python-sqlite2' package as it seems to be required for TracModPython but not for tracd

Segmentation fault with php5-mhash or other php5 modules

You may encounter segfaults (reported on debian etch) if php5-mhash module is installed. Try to remove it to see if this solves the problem. See debian bug report ?http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=411487

Some people also have troubles when using php5 compiled with its own 3rd party libraries instead of system libraries. Check here ?http://www.djangoproject.com/documentation/modpython/#if-you-get-a-segmentation-fault

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