mod_wsgi (Apache)

If you are using the Apache webserver, consider using mod_wsgi.

Watch Out:

Please make sure in advance that any `app.run()` calls you might have in your application file are inside an `if __name__ == '__main__':` block or moved to a separate file. Just make sure it's not called because this will always start a local WSGI server which we do not want if we deploy that application to mod_wsgi.

Installing *mod_wsgi*

If you don't have *mod_wsgi* installed yet you have to either install it using a package manager or compile it yourself. The mod_wsgi installation instructions cover source installations on UNIX systems.

If you are using Ubuntu/Debian you can apt-get it and activate it as follows:

```
$ apt-get install libapache2-mod-wsgi
```

If you are using a yum based distribution (Fedora, OpenSUSE, etc..) you can install it as follows:

```
$ yum install mod_wsgi
```

On FreeBSD install *mod_wsgi* by compiling the *www/mod_wsgi* port or by using pkg_add:

```
$ pkg install ap22-mod_wsgi2
```

If you are using pkgsrc you can install *mod_wsgi* by compiling the *www/ap2-wsgi* package.

If you encounter segfaulting child processes after the first apache reload you can safely ignore them. Just restart the server.

v: 1.1.x

Creating a *.wsgi* file

To run your application you need a `yourapplication.wsgi` file. This file contains the code *mod_wsgi* is executing on startup to get the application object. The object called *application* in that file is then used as application.

For most applications the following file should be sufficient:

```
from yourapplication import app as application
```

If a factory function is used in a `__init__.py` file, then the function should be imported:

```
from yourapplication import create_app
application = create_app()
```

If you don't have a factory function for application creation but a singleton instance you can directly import that one as *application*.

Store that file somewhere that you will find it again (e.g.: `/var/www/yourapplication`) and make sure that *yourapplication* and all the libraries that are in use are on the python load path. If you don't want to install it system wide consider using a virtual python instance. Keep in mind that you will have to actually install your application into the virtualenv as well. Alternatively there is the option to just patch the path in the `.wsgi` file before the import:

```
import sys
sys.path.insert(0, '/path/to/the/application')
```

Configuring Apache

The last thing you have to do is to create an Apache configuration file for your application. In this example we are telling *mod_wsgi* to execute the application under a different user for security reasons:

```
<VirtualHost *>
    ServerName example.com

    WSGIDaemonProcess yourapplication user=user1 group=group1 threads=5
    WSGIScriptAlias / /var/www/yourapplication/yourapplication.wsgi

    <Directory /var/www/yourapplication>
        WSGIProcessGroup yourapplication
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

Note: WSGIDaemonProcess isn’t implemented in Windows and Apache will refuse to run with the above configuration. On a Windows system, eliminate those lines:

```
<VirtualHost *>
    ServerName example.com
    WSGIScriptAlias / C:\yourdir\yourapp.wsgi
    <Directory C:\yourdir>
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

Note: There have been some changes in access control configuration for Apache 2.4.

Most notably, the syntax for directory permissions has changed from httpd 2.2

```
Order allow,deny
Allow from all
```

to httpd 2.4 syntax

```
Require all granted
```

For more information consult the mod_wsgi documentation.

Troubleshooting

If your application does not run, follow this guide to troubleshoot:

Problem: application does not run, errorlog shows SystemExit ignored

> You have an `app.run()` call in your application file that is not guarded by an `if __name__ == '__main__':` condition. Either remove that `run()` call from the file and move it into a separate `run.py` file or put it into such an if block.

Problem: application gives permission errors

> Probably caused by your application running as the wrong user. Make sure the folders the application needs access to have the proper privileges set and the applic
> as the correct user (`user` and `group` parameter to the *WSGIDaemonProcess* directive)

Problem: application dies with an error on print

Keep in mind that mod_wsgi disallows doing anything with **`sys.stdout`** and **`sys.stderr`**. You can disable this protection from the config by setting the *WSGIRestrictStdout* to `off`:

```
WSGIRestrictStdout Off
```

Alternatively you can also replace the standard out in the .wsgi file with a different stream:

```
import sys
sys.stdout = sys.stderr
```

Problem: accessing resources gives IO errors

Your application probably is a single .py file you symlinked into the site-packages folder. Please be aware that this does not work, instead you either have to put the folder into the pythonpath the file is stored in, or convert your application into a package.

The reason for this is that for non-installed packages, the module filename is used to locate the resources and for symlinks the wrong filename is picked up.

Support for Automatic Reloading

To help deployment tools you can activate support for automatic reloading. Whenever something changes the `.wsgi` file, *mod_wsgi* will reload all the daemon processes for us.

For that, just add the following directive to your *Directory* section:

```
WSGIScriptReloading On
```

Working with Virtual Environments

Virtual environments have the advantage that they never install the required dependencies system wide so you have a better control over what is used where. If you want to use a virtual environment with mod_wsgi you have to modify your `.wsgi` file slightly.

Add the following lines to the top of your `.wsgi` file:

```
activate_this = '/path/to/env/bin/activate_this.py'
execfile(activate_this, dict(__file__=activate_this))
```

v: 1.1.x

For Python 3 add the following lines to the top of your `.wsgi` file:

```
activate_this = '/path/to/env/bin/activate_this.py'
with open(activate_this) as file_:
    exec(file_.read(), dict(__file__=activate_this))
```

This sets up the load paths according to the settings of the virtual environment. Keep in mind that the path has to be absolute.