

GETTING STARTED WITH MOD_WSGI

Graham Dumpleton

PyCon Australia
26th June 2010



WHAT IS WSGI?

- Specification for an interface between web servers and Python web applications or frameworks.
- Intended to promote web application portability across a variety of web servers.
- PEP 333 - <http://www.python.org/dev/peps/pep-0333>
- More Information - <http://www.wsgi.org>

WHAT IS MOD_WSGI?

- An Apache module to support hosting WSGI applications in conjunction with the Apache web server.
- Intercepts requests to designated URLs and passes requests off to the WSGI application specified in the target WSGI script file that the URL maps to.

HELLO WORLD

```
def application(environ, start_response):  
    status = '200 OK'  
    output = 'Hello World!'  
  
    response_headers = [('Content-type', 'text/plain'),  
                        ('Content-Length', str(len(output)))]  
    start_response(status, response_headers)  
  
    return [output]
```


MAPPING A URL

- Mapping WSGI application at root of web server.

```
WSGIScriptAlias / /home/grumpy/example-1/hello.wsgi
```

- Mapping WSGI application at sub URL of web server.

```
WSGIScriptAlias /suburl /home/grumpy/example-1/hello.wsgi
```

FORBIDDEN



client denied by server configuration: /home/grumpy/example-1/hello.wsgi

DENY FROM ALL

- By default Apache denies access to everything.

```
<Directory />  
Order deny,allow  
Deny from all  
</Directory>
```

- Thus we need to open up access to our script.

```
<Directory /home/grumpy/example-1>  
Order deny,allow  
Allow from all  
</Directory>
```


FORBIDDEN



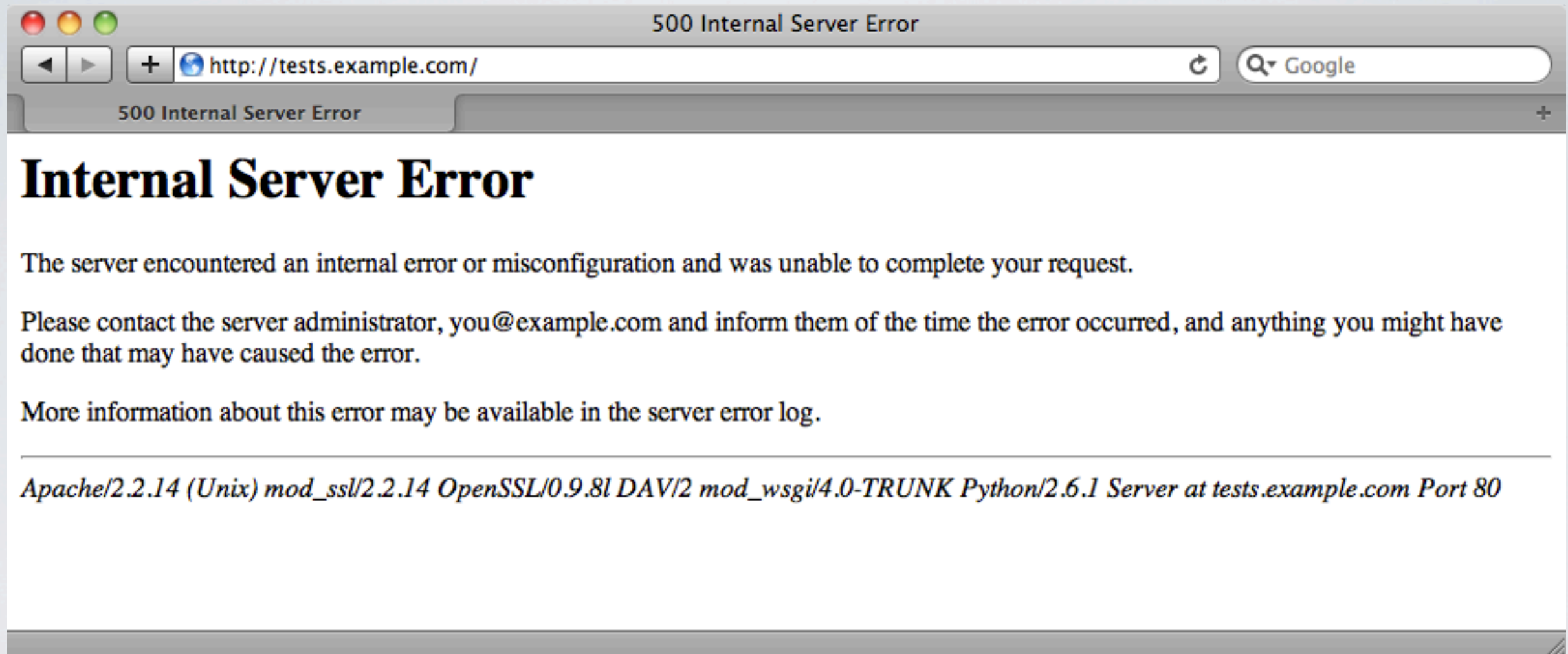
(13)Permission denied: access to / denied

DRWXR-X---

- Apache server child processes runs as a special user.
- The Apache user must be able to search directories down to where the WSGI script file is located.
- User account home directories often have permissions of drwxr-x--- and access is thereby restricted.
- Make directories accessible to others.

`chmod o+rx /home/grumpy`
- Better still, don't put WSGI script files under home directories.

INTERNAL SERVER ERROR



```
(13)Permission denied: mod_wsgi (pid=666, process='',  
application='tests.example.com|'): Call to fopen() failed for '/home/  
grumpy/example-1/echo.wsgi'.
```


-RW-R-----

- Apache server child processes runs as a special user.
- The Apache user must be able to read the WSGI script file.
- A user account umask of 0007 will result in files being created with permissions of -rw-r----- and access is thereby restricted.
- Make files readable to others.

```
chmod o+r /home/grumpy/example-1/hello.wsgi
```

SUCCESS



DJANGO APPLICATION

- Install Django.
- Create an empty project.

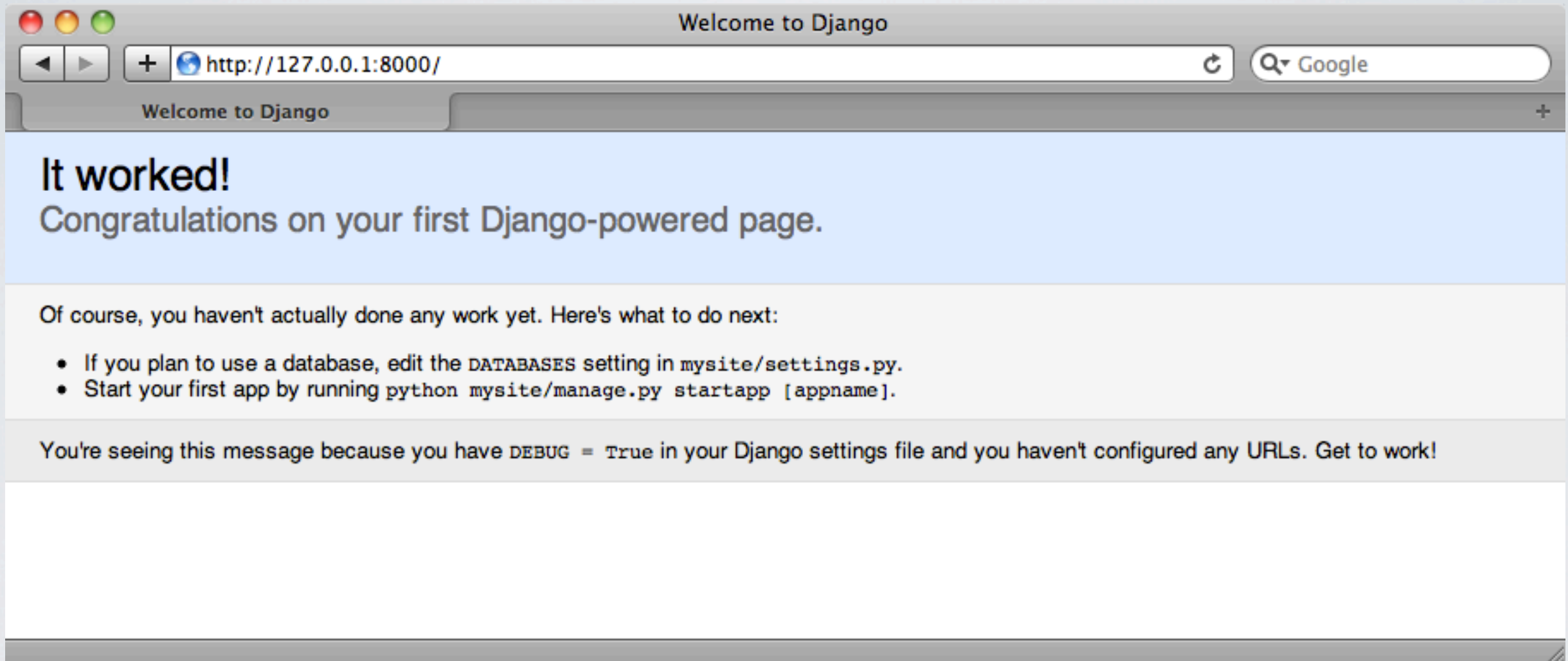
```
mkdir /home/grumpy/example-2  
cd /home/grumpy/example-2
```

```
django-admin.py startproject mysite
```

- Run the Django development server.

```
python mysite/manage.py runserver
```

SUCCESS



DJANGO WSGI SCRIPT

```
import os
import sys

os.environ[ 'DJANGO_SETTINGS_MODULE' ] = 'mysite.settings'

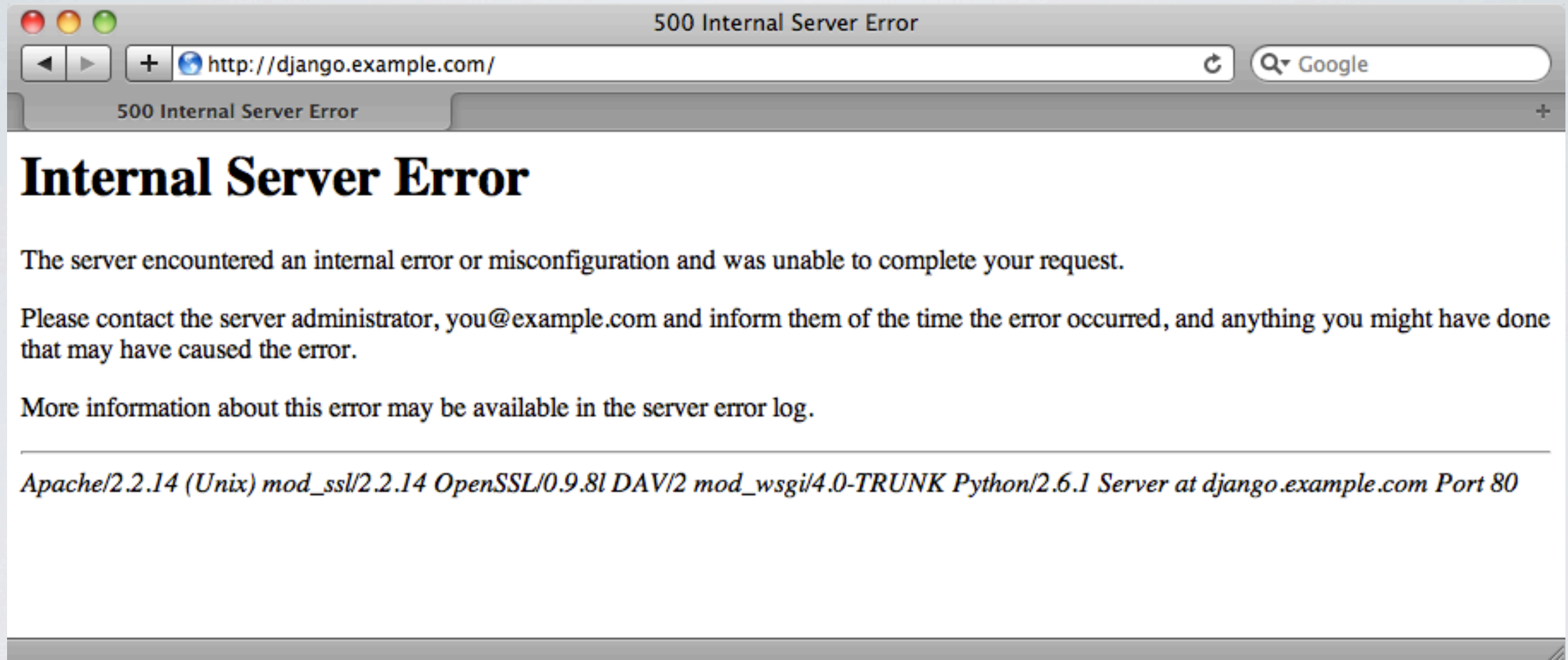
import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```

DJANGO APACHE CONFIG

```
WSGIScriptAlias / /home/grumpy/example-2/apache/mysite.wsgi
```

```
<Directory /home/grumpy/example-2/apache>  
Order deny,allow  
Allow from all  
</Directory>
```


INTERNAL SERVER ERROR



```
raise ImportError("Could not import settings '%s' (Is it on sys.path?  
Does it have syntax errors?): %s" % (self.SETTINGS_MODULE, e))  
ImportError: Could not import settings 'mysite.settings' (Is it on  
sys.path? Does it have syntax errors?): No module named mysite.settings
```

SYS.PATH

- Python modules/packages are not imported relative to current working directory.
- Python modules/packages are not imported relative to the directory containing the WSGI script file.
- The PYTHONPATH of the user who owns the WSGI script file is not consulted.
- Therefore must explicitly designate directories to search for Python modules/packages.

SYS.PATH.INSERT()

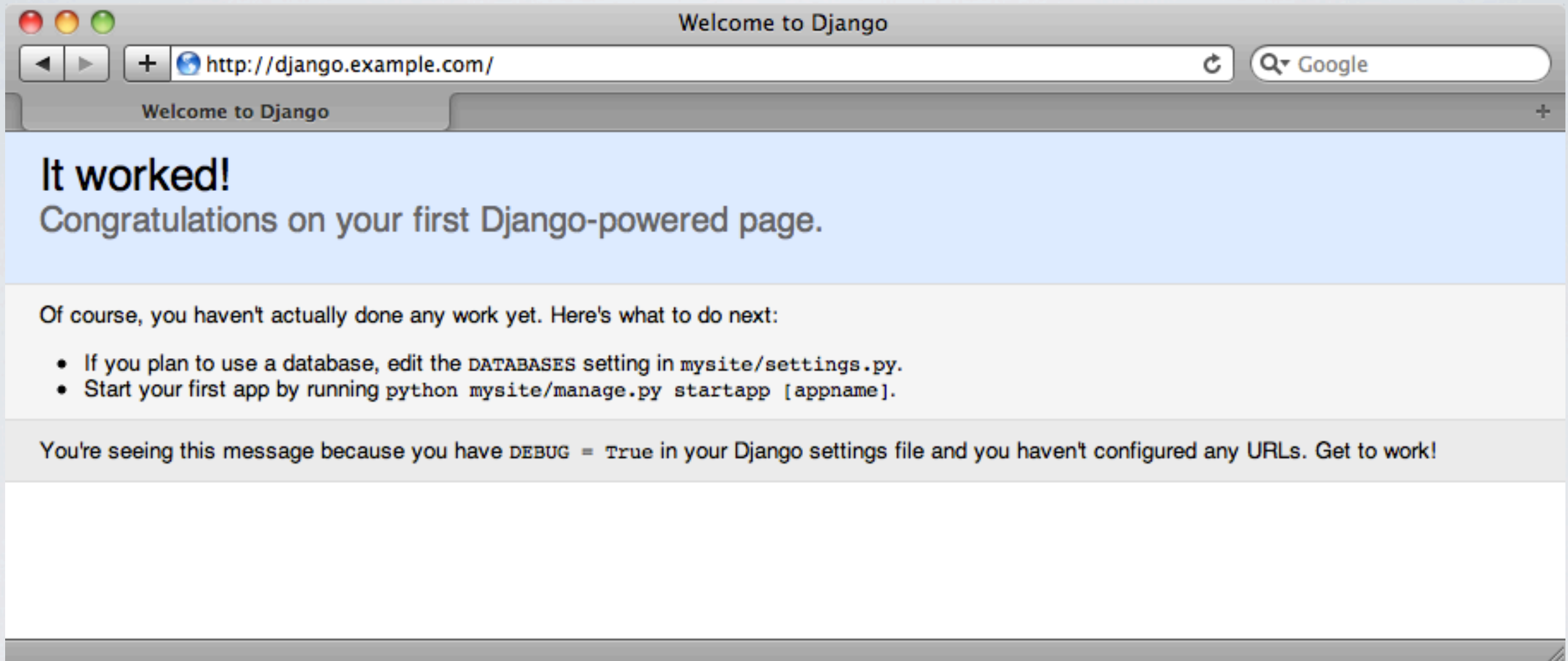
```
import os
import sys

root = os.path.join(os.path.dirname(__file__), '..')
sys.path.insert(0, root)

os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'

import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```

SUCCESS



VIRTUAL ENVIRONMENT

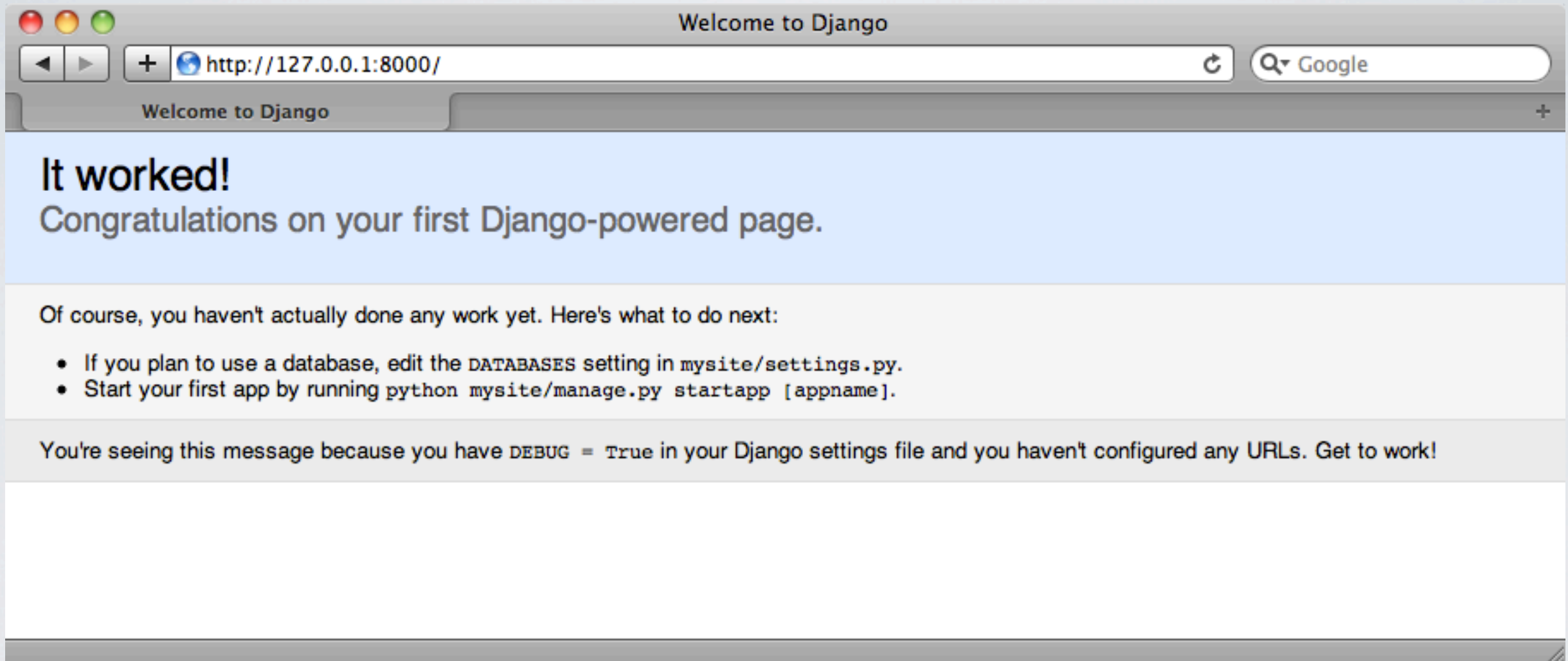
- Create and activate virtual environment

```
cd /home/grumpy/example-2  
virtualenv --no-site-packages environ  
source environ/bin/activate
```

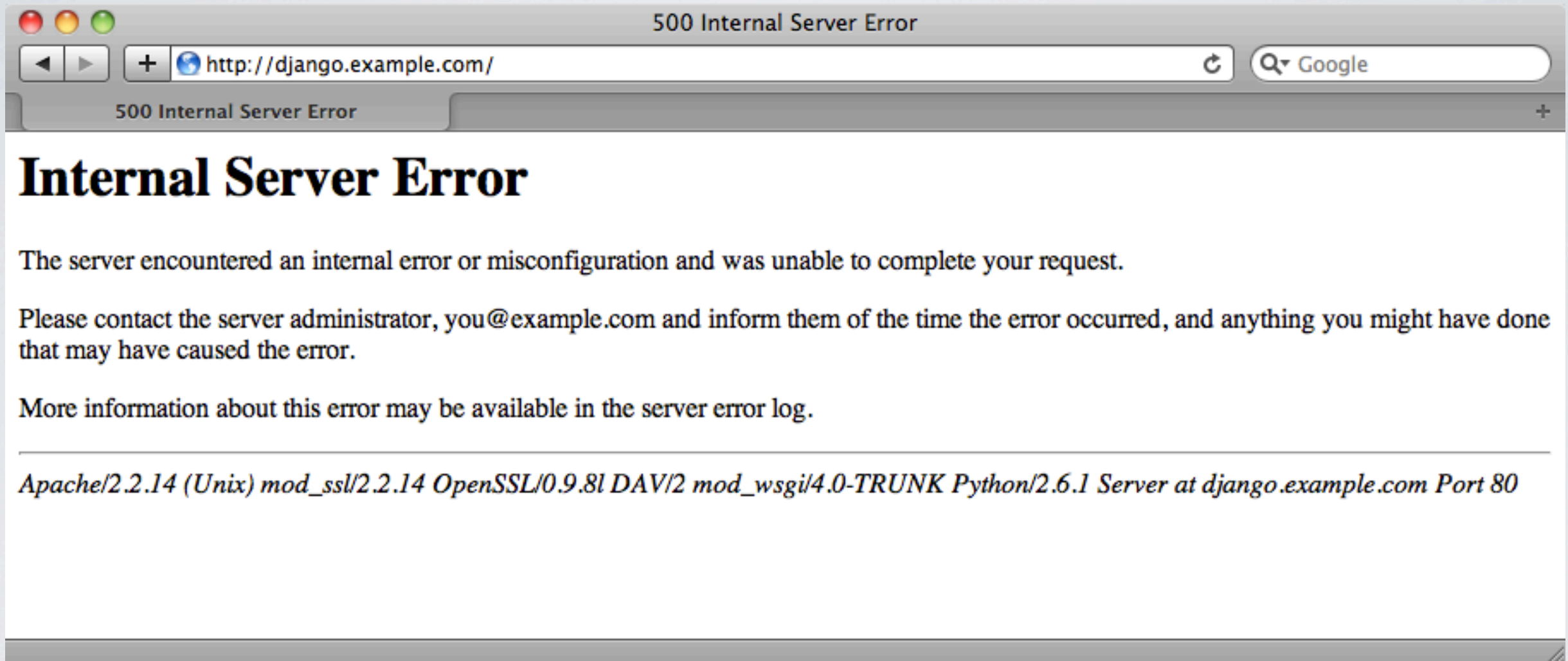
- Install Django and other required packages.

```
easy_install Django
```

SUCCESS



INTERNAL SERVER ERROR



```
ImportError: No module named django.core.handlers.wsgi
```

SYS.PATH.INSERT()

```
import os
import sys

root = os.path.join(os.path.dirname(__file__), '..')
sys.path.insert(0, root)

packages = os.path.join(root,
    'environ/lib/python2.6/site-packages'
sys.path.insert(0, packages)

os.environ[ 'DJANGO_SETTINGS_MODULE' ] = 'mysite.settings'

import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```


SITE.ADDSITEDIR()

```
import os
import sys
import site

root = os.path.join(os.path.dirname(__file__), '..')
sys.path.insert(0, root)

packages = os.path.join(root,
    'environ/lib/python2.6/site-packages'
site.addsitedir(packages)

os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'

import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```

ACTIVATE THIS

```
import os
import sys

root = os.path.join(os.path.dirname(__file__), '..')
sys.path.insert(0, root)

activate_this = os.path.join(root,
    'environ/bin/activate_this.py')
execfile(activate_this, dict(__file__=activate_this))

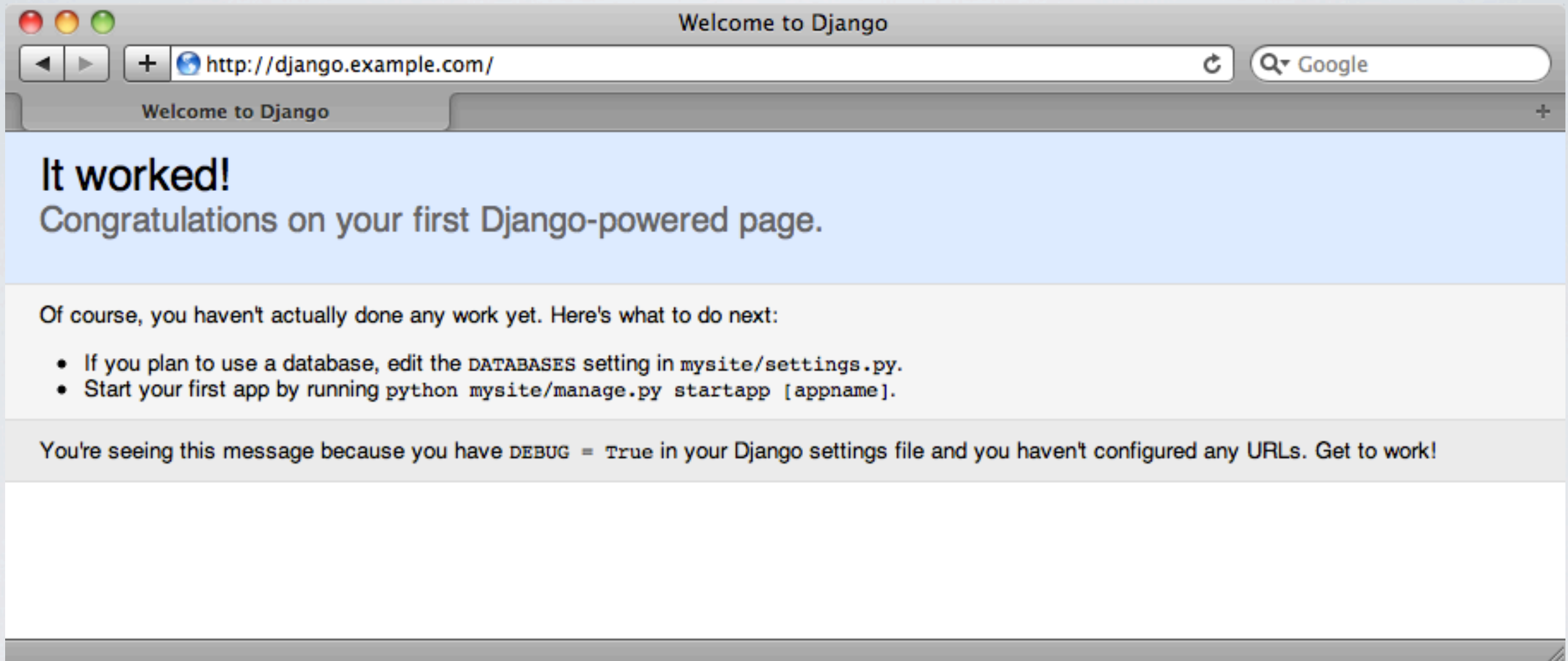
os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'

import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```


SITE-PACKAGES

- Using `sys.path.insert()` allows directory to be placed first, but `.pth` files do not get processed.
- Using `site.addsitedir()` results in `.pth` files being processed, but new directories get placed last.
- Using `activate_this.py` results in `.pth` files being processed and `sys.path` reordered such that new directories get placed first.
- Value of `sys.prefix` is however altered by `activate_this.py` and no certainty over whether this may cause later issues.

SUCCESS

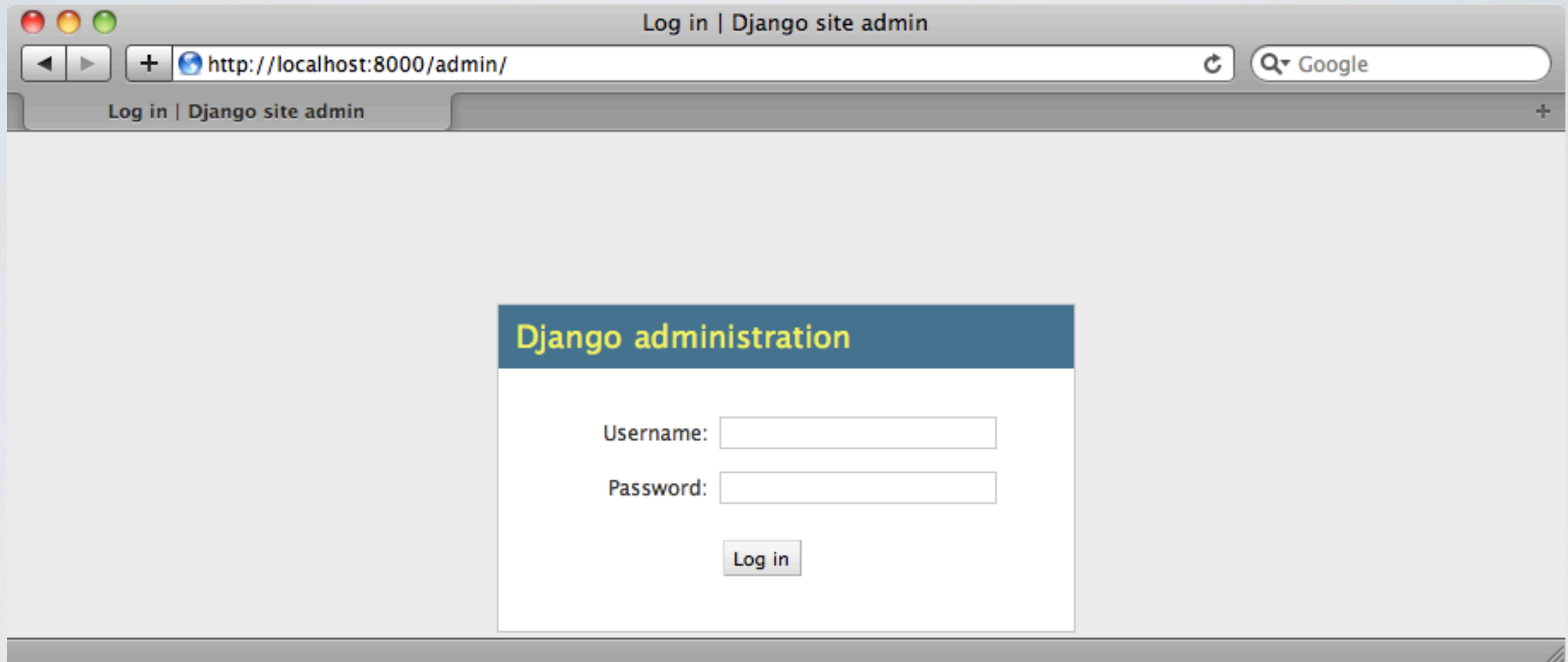


DJANGO ADMIN PAGES

- Add 'django.contrib.admin' to INSTALLED_APPS in the file 'mysite/settings.py'.
- Add (r'^admin/', include(admin.site.urls)) to 'urlpatterns' in the file 'mysite/urls.py'.
- Configure 'DATABASES' in the file 'mysite/settings.py'.
- Synchronise database model.

```
python mysite/manage.py syncdb
```

SUCCESS



The image shows a web browser window with the title "Log in | Django site admin". The address bar contains the URL "http://localhost:8000/admin/". Below the address bar, there is a search bar with the text "Google". The main content area of the browser displays the Django administration login page. The page has a dark blue header with the text "Django administration" in yellow. Below the header, there is a white box containing the login form. The form has two input fields: "Username:" and "Password:". Below these fields is a "Log in" button.

Log in | Django site admin

http://localhost:8000/admin/

Google

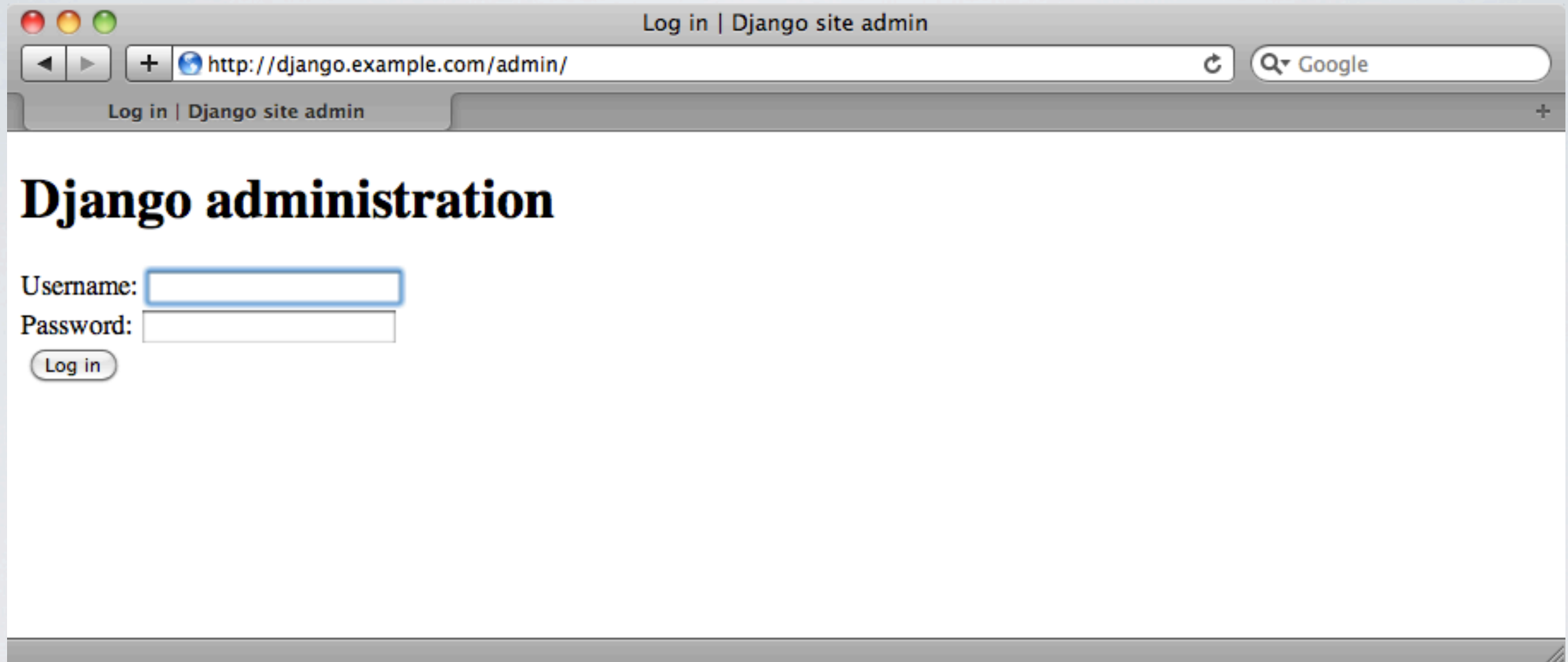
Django administration

Username:

Password:

Log in

NOT FOUND



Log in | Django site admin

Log in | Django site admin

Django administration

Username:

Password:

STATIC MEDIA FILES

- Django's static media files are only served automatically by the Django development server and not when using Django WSGIHandler object.
- Must manually map any static media files to appropriate sub URL in Apache.

ALLOW FROM ALL

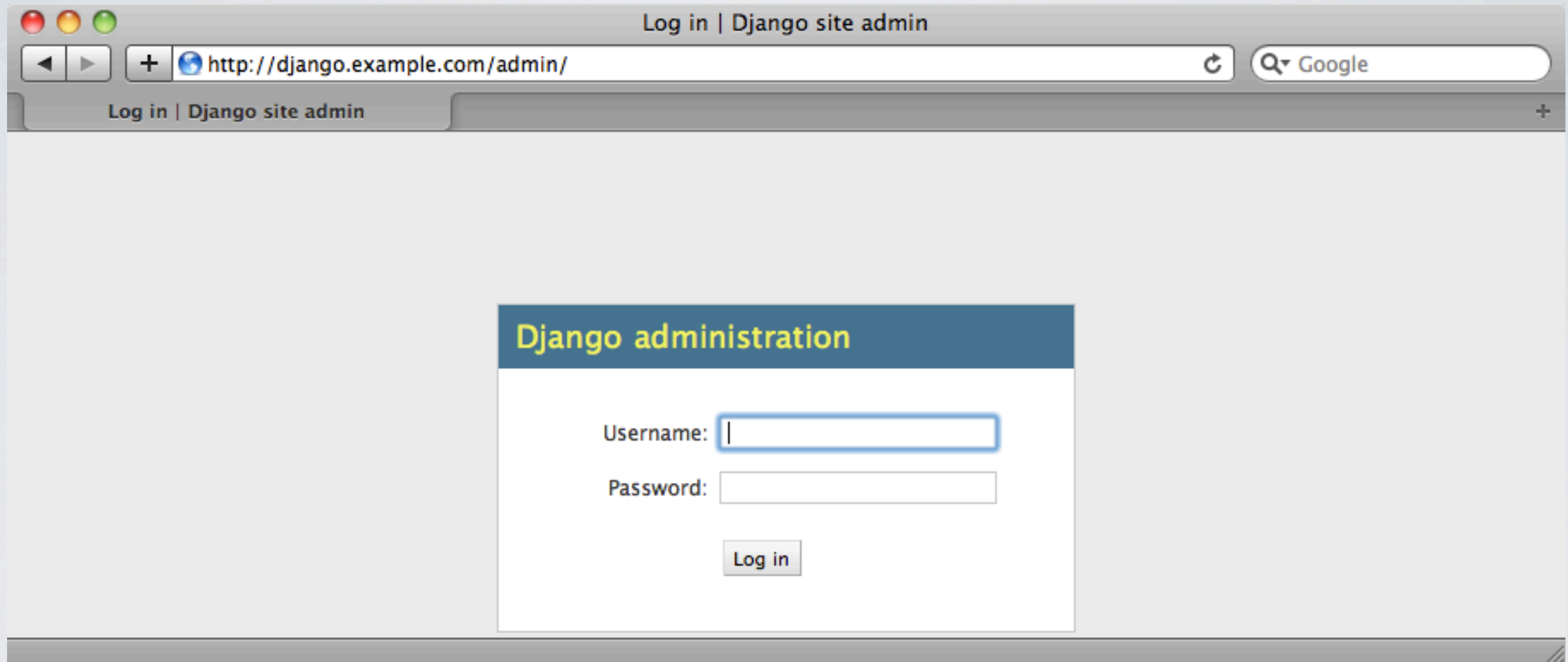
```
WSGIScriptAlias / /home/grumpy/example-2/apache/mysite.wsgi
```

```
<Directory /home/grumpy/example-2/apache>  
Order deny,allow  
Allow from all  
</Directory>
```

```
Alias /media/ /home/grumpy/example-2/media/
```

```
<Directory /home/grumpy/example-2/media/>  
Order deny,allow  
Allow from all  
</Directory>
```

SUCCESS



The screenshot shows a web browser window with the title "Log in | Django site admin". The address bar contains the URL "http://django.example.com/admin/". A search bar with the Google logo is visible on the right. The main content area displays the "Django administration" login form, which includes fields for "Username:" and "Password:", and a "Log in" button.

Log in | Django site admin

http://django.example.com/admin/

Google

Log in | Django site admin

Django administration

Username:

Password:

Log in

OTHER ISSUES

- Have to restart Apache every time a change is made to the application code.
- Data directories used by the application have to be writable by the Apache user.
- Occasionally notice that the web server is slow to respond to a request.

DAEMON MODE

```
WSGIDaemonProcess example-2 processes=2 threads=5 \  
    display-name=example-2
```

```
WSGIScriptAlias / /home/grumpy/example-2/apache/mysite.wsgi
```

```
<Directory /home/grumpy/example-2/apache>
```

```
WSGIProcessGroup example-2
```

```
Order deny,allow
```

```
Allow from all
```

```
</Directory>
```


DAEMON MODE

- Application runs in separate processes to Apache server child processes.
- Can touch the WSGI script file to have application be reloaded on next request.
- Send processes in daemon process group a TERM signal to have them reload immediately.
- Processes can be identified in 'ps' output if 'display-name' used and can use 'killall' to send them all signals to reload.
- Daemon mode not available on Windows or Apache 1.3.

ACCESS PRIVILEGES

```
WSGIDaemonProcess django processes=2 threads=5 \  
    display-name=example-2 user=grumpy group=grumpy
```

```
WSGIScriptAlias / /home/grumpy/example-2/apache/mysite.wsgi
```

```
<Directory /home/grumpy/example-2/apache>
```

```
WSGIProcessGroup example-2
```

```
Order deny,allow
```

```
Allow from all
```

```
</Directory>
```


ACCESS PRIVILEGES

- Application runs as specified user within specified group rather than Apache user and group.
- WSGI script file/directory must still be accessible by the Apache user.
- Application code files need only be readable to user the daemon process runs as.
- Data directories used by the application need only be writable to the user the daemon process runs as.

SCRIPT PRELOADING

```
WSGIDaemonProcess example-2 processes=2 threads=5 \  
    display-name=%{GROUP} user=grumpy group=grumpy
```

```
WSGIScriptAlias / /home/grumpy/example-2/apache/mysite.wsgi \  
    process-group=example-2 application-group=%{GLOBAL}
```

```
<Directory /home/grumpy/example-2/apache>  
Order deny,allow  
Allow from all  
</Directory>
```


SCRIPT PRELOADING

- Application bound to a specific process group and application group (interpreter).
- Because both process group and application group are fixed, the WSGI script file can be preloaded when the process is first started.
- Provided that the WSGI script preloads all required code, then no detectable delay should occur upon first request received by the application.
- The Django WSGIHandler alone doesn't however preload all required code. :-)

SCRIPT PRELOADING

```
import sys
site = os.path.join(os.path.dirname(__file__), '..', 'mysite')
sys.path.insert(0, site)

import settings

import django.core.management
django.core.management.setup_environ(settings)
utility = django.core.management.ManagementUtility()
command = utility.fetch_command('runserver')
command.validate()

import django.conf, django.utils
django.utils.translation.activate(
    django.conf.settings.LANGUAGE_CODE)

import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```

SUMMARY OF MAIN POINTS

- Need to tell Apache what resources it should allow access to.
- Apache user needs access to WSGI script file/directory.
- Application user needs read access to source code.
- Application user needs to be able to write to data directories.
- Python needs to know where to search for modules/packages.
- Daemon mode is the preferred way of deploying application.

ONLINE RESOURCES

- The mod_wsgi documentation.

<http://www.modwsgi.org>

<http://code.google.com/p/modwsgi/wiki/WhereToGetHelp>

<http://code.google.com/p/modwsgi/wiki/VirtualEnvironments>

<http://code.google.com/p/modwsgi/wiki/ReloadingSourceCode>

- Personal blog posts.

<http://blog.dscpl.com.au>

<http://blog.dscpl.com.au/2010/03/improved-wsgi-script-for-use-with.html>