Install and run Python Django using cPanel

Django is a Python-based framework that allows you to create websites quickly and easily. Use this description to set up the Django framework. After setup, you will get a working Django page that allows you to:

- Load static pages with a specific domain name
- Loading the Django administration interface
- SQLite database usage

CREATE A PYTHON APPLICATION IN THE CPANEL INTERFACE

1. Log in to your cPanel account
2. Find the **Python application setup** menu item and click on it
3. Setting up a Python application
    1. On the page that appears, click **Create Application**
        1. For **Python version**, select version **3.8.1**.
        2. Enter the application directory **(Application root /home/cpanel_username/**), for example: application
        3. In the **Application URL** section, select the domain name and leave the following field blank.
        4. Also leave the **Application launch file** and **Application entry point** fields blank.

> In this case, cPanel will automatically create the *passenger_wsgi.py* file and the default interpreted boot environment.

1. 1. 5. In the field after **Passenger log file**, you can enter the name of the log file that can help you troubleshoot.
        6. Then click the **Create** button in the upper right corner. cPanel then creates the Python application and sets the environment variables.

7. Copy the command after **Enter to the virtual environment. To enter to virtual environment, run the command:** at the top of the page, as you will need this command to configure the application.

SETTING UP DJANGO

After creating a Python application, the following tasks must be performed from the command line (You can also use the **Terminal** menu item in the cPanel interface for this purpose, but you can also connect to the repository using an SSH client.):

- Installing Django
- Create and configure the Django project
- Configure the Passenger for use with Django

Procedure for the above steps:

1. Log in to your webspacee via SSH or launch the **Terminal** menu item available from the cPanel interface.
2. Activate the virtual environment by pasting and running the previously copied command, for example:

```
source /home/cpanel_username/virtualenv/application/3.8/bin/activate && cd /home/cpanel_username/application
```

> The command line now begins with (**application: 3.8**), indicating that the application is working in a virtual environment with Python 3.8. All of the following commands in this article assume that you are working in a Python virtual environment. If you log out of the SSH session (or deactivate the virtual environment with the **deactivate** command), make sure that you have reactivated the virtual environment before performing any of the following steps.

3. To install Django, issue the following commands:

```
cd ~
pip install django==2.1.8
```

> To verify the Django installation, issue the following command:

```
django-admin --version
```

4. Create the Django project by issuing the following command:

```
django-admin startproject application ~/application
```

5. Create static project directories by issuing the following commands:

```
mkdir -p ~/application/templates/static_pages
mkdir ~/application/static_files
mkdir ~/application/static_media
```

6. Using a text editor, open the ***~/application/application/settings.py*** file and make the following changes to the file:
   - Find the **ALLOWED_HOSTS** line item and make changes as follows. Replace *example.com* with your own domain name:

     ```
     ALLOWED_HOSTS = ['example.com']
     ```

   - Locate the **TEMPLATES** block and modify it as follows:

     ```
     TEMPLATES = [
        {
           'BACKEND': 'django.template.backends.django.DjangoTemplates',
           'DIRS': [os.path.join(BASE_DIR,'templates')],
           'APP_DIRS': True,
           'OPTIONS': {
              'context_processors': [
                 'django.template.context_processors.debug',
                 'django.template.context_processors.request',
                 'django.contrib.auth.context_processors.auth',
                 'django.contrib.messages.context_processors.messages',
              ],
           },
        },
     ]
     ```

   - Find the line **STATIC_URL** and add the following lines:

     ```
     STATIC_URL = '/static/'
     STATIC_ROOT = os.path.join(BASE_DIR, 'static_files')

     MEDIA_URL = '/media/'
     MEDIA_ROOT = os.path.join(BASE_DIR, "static_media")
     ```

7. Using a text editor, open the ***~/application/application/urls.py*** file, then delete the entire contents and paste the following contents:

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from django.conf.urls import url
from django.views.generic.base import TemplateView

urlpatterns = [
```

```
    path('admin/', admin.site.urls),
    url(r'^$', TemplateView.as_view(template_name='static_pages/index.html'), name='home'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

8. Using a text editor, open the file ***~/application/passenger_wsgi.py*** then delete the entire contents and paste the following contents:

```
import os
import sys

import django.core.handlers.wsgi
from django.core.wsgi import get_wsgi_application

# Set up paths and environment variables
sys.path.append(os.getcwd())
os.environ['DJANGO_SETTINGS_MODULE'] = 'alkalmazas.settings'

# Set script name for the PATH_INFO fix below
SCRIPT_NAME = os.getcwd()

class PassengerPathInfoFix(object):
    """
        Sets PATH_INFO from REQUEST_URI because Passenger doesn't provide it.
    """
    def __init__(self, app):
        self.app = app

    def __call__(self, environ, start_response):
        from urllib.parse import unquote
        environ['SCRIPT_NAME'] = SCRIPT_NAME
        request_uri = unquote(environ['REQUEST_URI'])
        script_name = unquote(environ.get('SCRIPT_NAME', ''))
        offset = request_uri.startswith(script_name) and len(environ['SCRIPT_NAME']) or 0
        environ['PATH_INFO'] = request_uri[offset:].split('?', 1)[0]
        return self.app(environ, start_response)

# Set the application
application = get_wsgi_application()
application = PassengerPathInfoFix(application)
```

9. Then create the default index.html file within the ***~/application/templates/static_pages*** directory. this file is a default static file that displays only the text **Hello World**.
10. Issue the following command:

```
python ~/application/manage.py migrate
```

11. Create the administrator user using the following command: To do this, first issue the following command:

```
python ~/application/manage.py createsuperuser
```

When prompted for the **Username**, type the new user name and press enter.
When prompted for an **email address**, type the email address and press enter.
When prompted for the administrator password, type the new **password**, and then press enter:

12. You can compile static content by issuing the following command:

```
python ~/application/manage.py collectstatic
```

> When prompted to overwrite existing files, type **yes** and then press enter.

13. Restart Python in the cPanel interface.
    - To do this, find and click **Python Setup** in the cPanel
    - On the **Web Applications** tab, locate the application, and then click **Restart Application** in the **Actions** column.
14. The last step is to test the application via the set URL:
    - In a browser, open the website (http://www.example.com/) where the index.html file should appear. It is important to replace the domain name example.com with the domain name you are using.
    - The next step is to open the address http://www.example.com/admin/ where the administration interface of the Django application should appear. It is important to replace the domain name example.com with the domain name you are using.

> In case the desired content does not appear in the browser, try running the application from the command line using the following command:

```
python ~/application/passenger_wsgi.py
```

> In this case, the console will display the cause of the error. If you specified the logging path when setting up Python, errors are also recorded in the log file.

MORE INFORMATION

Since the current application is already running, you can actually start developing your own application, for which you can find further help at the following links:

- The official documentation for Django is available at: http://docs.djangoproject.com.
- For Django extensions, visit https://github.com/django-extensions/django-extensions.
- The *south* library is a popular add-on to database migrations that you can learn more about at https://pypi.python.org/pypi/South.
- The *fabric* library helps simplify application development, more detailed information can be found here http://docs.fabfile.org.

Revision #1
Created 25 October 2023 12:18:25 by Judit Pásztor
Updated 25 October 2023 12:26:01 by Judit Pásztor