django-windows-tools Documentation

Release 0.1.0

Antoine Martin

April 14, 2015

Contents

django-windows-tools is a small Django application providing management commands to help hosting Django projects in a Windows environment.

This project started when a Django project that started as a temporary proof of concept running on a Linux box became something that needed to go to production in a IIS/SQL Server environment.

We faced three concerns:

- Database access.
- Running a Django application behind IIS.
- Running Django background processes (Celery, Celery Beat)

The database is a no brainer with the help of django-mssql and pywin32 as it allowed an allmost seamless switch between MySQL and SQL Server.

For Hosting the Django project behind IIS, things became harder. There are several solutions around (such as the one from HeliconTech), but they are either unmaintained, convoluted or Closed Source. We came out with a solution that needs only Open Source software and that can easily be automated.

Last, for background and scheduled task, one wants to use celery and its beat scheduler. Again, we came out with a solution allowing to run the Django Background processes in a Windows Service.

django-windows-tools packages the solutions we found and provides Django management commands that ease the deployment and configuration of a Django project on Windows.

CHAPTER 1

Documentation

1.1 Quick Start

1.1.1 Installation and Configuration

You install the application with the command:

```
pip install django-windows-tools
```

Or you may install the latest development version for more fixed bugs:

```
pip install git+https://github.com/antoinemartin/django-windows-tools@master
```

Enable the `django_windows_tools` application to be able to use the management commands. Add the app to the project's list in `settings.py`:

```
INSTALLED_APPS += (
    'django_windows_tools',
)
```

1.1.2 FastCGI Configuration

Pre-requisites

On the host machine, you need to have :

- IIS 7 or better installed and running.
- The `CGI` module installed.

To host your Django project under `IIS` with the binding `www.mydjangoapp.com`, you need first to collect your static files with the command:

```
D:\sites\mydjangoapp> python manage.py collectstatic
```

And then run the following command with Administrator privileges :

```
D:\sites\mydjangoapp> python manage.py winfcgi_install --binding=http://www.mydjangoapp.com:80
```

The command will do the following:

- Create the FastCGI application to serve your Django application dynamic content.

- Create a site name `mydjangoapp` with the `www.mydjangoapp.com` binding pointing to the root of your project.
- Install a `web.config` file in the root of the project that handles the redirection of requests to the Django application.
- Create if needed a virtual directory to handle the serving of your static files through `IIS`.

To remove the site created with the preceding command, type:

```
D:\sites\mydjangoapp> python manage.py winfcgi_install --delete
```

the `winfcgi_install` command provides numerous options. To list them, type:

```
D:\sites\mydjangoapp> python help winfcgi_install
```

More information on how the configuration is done is provided in this Blog post.

1.1.3 Running Celery or other Background commands as a Windows Service

With the application installed, on the root of your project, type the following command:

```
D:\sites\mydjangoapp> python winservice_install
```

It will create two files, `service.py` and `service.ini` in the root directory of your project. The first one will help you install, run and remove the Windows Service. Ther later one contain the list of the management commands that will be run by the Windows Service.

Configuration

The `service.ini` is a configuration file that looks like the following:

```
[services]
# Services to be run on all machines
run=celeryd
clean=d:\logs\celery.log

[BEATSERVER]
# There should be only one machine with the celerybeat service
run=celeryd celerybeat
clean=d:\logs\celerybeat.pid;d:\logs\beat.log;d:\logs\celery.log

[celeryd]
command=celeryd
parameters=-f d:\logs\celery.log -l info

[celerybeat]
command=celerybeat
parameters=-f d:\logs\beat.log -l info --pidfile=d:\logs\celerybeat.pid

[runserver]
# Runs the debug server and listen on port 8000
# This one is just an example to show that any manage command can be used
command=runserver
parameters=--noreload --insecure 0.0.0.0:8000

[log]
```

```
filename=d:\logs\service.log
level=INFO
```

The `services` section contains :

- The list of background commands to run in the `run` directive.
- The list of files to delete when refreshed or stopped in the `clean` directive.

You can have several `services` sections in the same configuration file for different host servers. The Windows Service will try to find the section which name matches the name of the current server and will fallback to the `services` section if it does not find it. This allows you to deploy the same configuration file on serveral machines but only have one machine run the celery beat background process. In the preceding configuration, only the server named `BEATSERVER` will run the `celerybeat` command. The other ones will only run the `celeryd` command.

For each command name specified in the `run` directive, there must be a matching configuration section. The section contains two directives:

- `command` specifies the `manage.py` command to run.
- `parameters` specifies the parameters to the command.

In the previous configuration file, the `celeryd` configuration will spawn a process that will run the same command as :

```
D:\sites\mydjangoapp> python manage.py celeryd -f d:\logs\celery.log -l info
```

Lastly, the `log` section defines the log level and the the log destination file for the Windows Service.

Installation and start

The windows service is installed with the following command (run with Administrator privileges) :

```
D:\sites\mydjangoapp> python service.py --startup=auto install
```

It is started and stopped with the commands:

```
D:\sites\mydjangoapp> python service.py start
D:\sites\mydjangoapp> python service.py stop
```

It can be removed with the following commands:

```
D:\sites\mydjangoapp> python service.py remove
```

The Windows Service monitor changes to the `service.ini` configuration file. In case it is modified, the service does the following:

- Stop the background processes.
- Reread the configuration file.
- Start the background processes.

Customization

The `winservice_install` management command provides several options allowing to customize the name of the web service or of the script name. To obtain information about them, type:

```
D:\sites\mydjangoapp> python help winservice_install
```

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*