django-configurations Documentation

Release dev

Jannis Leidel

September 03, 2013

CONTENTS

django-configurations eases Django project configuration by relying on the composability of Python classes. It extends the notion of Django's module based settings loading with well established object oriented programming patterns.

CHAPTER

ONE

QUICKSTART

Install django-configurations:

```
pip install django-configurations
```

Then subclass the included `configurations.Settings` class in your project's `settings.py` or any other module you're using to store the settings constants, e.g.:

```
from configurations import Settings

class MySiteSettings(Settings):
    DEBUG = True
```

Set the `DJANGO_CONFIGURATION` environment variable to the name of the class you just created, e.g. in bash:

```
export DJANGO_CONFIGURATION=MySettings
```

and the `DJANGO_SETTINGS_MODULE` environment variable to the module import path as usual, e.g. in bash:

```
export DJANGO_SETTINGS_MODULE=mysite.settings
```

To enable Django to use your configuration you now have to modify your `manage.py` or `wsgi.py` script to use django-configurations's versions of the appropriate starter functions, e.g. a typical `manage.py` using django-configurations would look like this:

```
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mysite.settings')
    os.environ.setdefault('DJANGO_', 'MySettings')

    from configurations.management import execute_from_command_line

    execute_from_command_line(sys.argv)
```

Notice in line 9 we don't use the common tool `django.core.management.execute_from_command_line` but instead `configurations.management.execute_from_command_line`.

The same applies to your `wsgi.py` file, e.g.:

```
import os

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mysite.settings')
os.environ.setdefault('DJANGO_CONFIGURATION', 'MySettings')
```

```
from configurations.wsgi import get_wsgi_application

application = get_wsgi_application()
```

Here we don't use the default `django.core.wsgi.get_wsgi_application` function but instead `configurations.wsgi.get_wsgi_application`.

That's it! You can now use your project with `manage.py` and your favorite WSGI enabled server.

CHAPTER

TWO

WAIT, WHAT?

django-configurations helps you organize the configuration of your Django project by providing the glue code to bridge between Django's module based settings system and programming patterns like mixins, facades, factories and adapters that are useful for non-trivial configuration scenarios.

It allows you to use the native abilities of Python inheritance without the side effects of module level namespaces that often lead to the unfortunate use of the `import *` anti-pattern.

CHAPTER

THREE

OKAY, HOW DOES IT WORK?

Any subclass of the `configurations.Settings` class will automatically use the values of its class and instance attributes (including properties and methods) to set module level variables of the same module – that's how Django will interface to the django-configurations based settings during startup and also the reason why it requires you to use its own startup functions.

That means when Django starts up django-configurations will have a look at the `DJANGO_CONFIGURATION` environment variable to figure out which class in the settings module (as defined by the `DJANGO_SETTINGS_MODULE` environment variable) should be used for the process. It then instantiates the class defined with `DJANGO_CONFIGURATION` and copies the uppercase attributes to the module level variables.

CHAPTER

FOUR

BUT ISN'T THAT MAGIC?

Yes, it looks like magic, but it's also maintainable and non-intrusive. No monkey patching is needed to teach Django how to load settings via django-configurations because it uses Python import hooks (PEP 302) behind the scenes.

CHAPTER

FIVE

USAGE PATTERNS

There are various configuration patterns that can be implemented with django-configurations. The most common pattern is to have a base class and various subclasses based on the enviroment they are supposed to be used in, e.g. in production, staging and development.

5.1 Server specific settings

For example, imagine you have a base setting class in your `settings.py` file:

```
from configurations import Settings

class Base(Settings):
    TIME_ZONE = 'Europe/Berlin'

class Dev(Base):
    DEBUG = True
    TEMPLATE_DEBUG = DEBUG

class Prod(Base):
    TIME_ZONE = 'America/New_York'
```

You can now set the `DJANGO_CONFIGURATION` environment variable to one of the class names you've defined, e.g. on your production server it should be `Prod`. In bash that would be:

```
export DJANGO_SETTINGS_MODULE=mysite.settings
export DJANGO_CONFIGURATION=Prod
```

5.2 Global settings defaults

Every `configurations.Settings` subclass will automatically contain Django's global settings as class attributes, so you can refer to them when setting other values, e.g.:

```
from configurations import Settings

class Base(Settings):
    TEMPLATE_CONTEXT_PROCESSORS = \
        Settings.TEMPLATE_CONTEXT_PROCESSORS + (
            'django.core.context_processors.request',
        )
```

```
    @property
    def LANGUAGES(self):
        return Settings.LANGUAGES + (('tlh', 'Klingon'),)
```

5.3 Mixins

You might want to apply some configuration values for each and every project you're working on without having to repeat yourself. Just define a few mixin you re-use multiple times:

```
class FullPageCaching(object):
    USE_ETAGS = True
```

Then import that mixin class in your site settings module and use it with a Settings class:

```
from configurations import Settings

class AcmeProd(Settings, FullPageCaching):
    DEBUG = False
    # ...
```

CHAPTER

SIX

THANKS

- The Pinax project for spearheading the efforts to extend the Django project metaphor with reusable project templates and a flexible configuration environment.
- django-classbasedsettings by Matthew Tretter for being the immediate inspiration for django-configurations.

CHAPTER

SEVEN

BUGS AND FEATURE REQUESTS

As always you mileage may vary, so please don't hesitate to send in feature requests and bug reports at the usual place:

https://github.com/jezdez/django-configurations/issues

Thanks!