Django's System Check Framework

Adam Johnson - me@adamj.eu

13 July 2015

What is it?

- It checks your code!

What is it?

- Added to Django in version 1.7
- Checks things after 'ready' time, such as models, deprecated features, and contrib apps
- Can be run alone as manage.py command
- Looks like this:

```
$ ./manage.py check
System check identified some issues:

WARNINGS:

?: (1_6.W001) Some project unittests may not execute as
  expected.
     HINT: Django 1.6 introduced a new default test
     runner. It looks like this project was generated
     using Django 1.5 or earlier. You should ensure your
     tests are all running & behaving as expected. See https:/
     for more information.
```

Why is it good?

- Code quality!

Why is it good?

- “Beyond linting” - code quality checks that can only be done at runtime, not by flake8
- Extensible - third party apps can provide checks too
- Runs on nearly every manage command - you can’t forget to run it, unlike tests!

Using it

- How do you use it?

Using it

- Normally it does its work without you realizing, until you see an error and correct it
- However it's easy to write your own checks - the framework is very simple
- At YPlan, we already had some sanity checks in place as unit tests - converting them to checks has improved the situation
- I'll show you how now

An ex-unit test

- An example old sanity check:

```
class SupportedFeaturesTest(SimpleTestCase):
    """Check features that might not be supported in
       some OS versions"""
    def test_strftime_dash_strips_leading_zeroes(self):
        # Not every unix strftime has %P
        dt = datetime(2000, 10, 11, 2, 12)
        self.assertEqual(
            dt.strftime('%-I:%M %P'),
            '2:12 am'
        )
```

- Bad! The test might not be run; or if it is and fails, its output is buried amongst all the other tests dependent on this failing

An ex-unit test

- Let's make it better!

Writing a check

- A basic check:

```
from django.core.checks import Error, Tags, register

@register(Tags.compatibility)
def check_system(app_configs, **kwargs):
    # app_configs is a list of AppConfig objects
    # kwargs is for future extension
    errors = []
    if bad_thing():
        errors.append(
            Error(
                "My error message",
                id='mycode.001'
            )
        )
    return errors
```

- Put in `myapp/apps.py` next to your `MyAppConfig` class

Converting our unit test

- Magicked into a mini check function:

```
def check_system(app_configs, **kwargs):
    errors = []
    errors.extend(_check_strftime())
    return errors

def _check_strftime():
    errors = []
    dt = datetime(2000, 10, 11, 2, 12)
    if dt.strftime('%-I:%M %P') != '2:12 am':
        errors.append(Error(
            "strftime does not appear to support using "
            "%- to strip leading zeroes",
            id='yplan.E007'
        ))
    return errors
```

Thank you

- me@adamj.eu ; blog at adamj.eu/tech/