Testing in Django

Django User Group Cologne, 19.9.2012

Stefan T. Oertel | @schneck

Wozu testen?

Aus Entwicklersicht:

- Sicherheit, dass alle Komponenten so funktionieren wie erwartet
- Sicherheit, dass ich/der Kollege das System nicht zerschossen hat
- Sicherheit beim Einführen neuer und Upgraden bestehender Komponenten
- Sicherheit, dass sich keine Typos oder Syntaxfehler eingeschlichen haben
- Sicherheit, dass das Deployment reibungslos verläuft

Sicherheit & entspanntes Arbeiten

Wozu testen?

Aus Kundensicht:

- Sicherheit, dass das System unterbrechungsfrei läuft
- Schmerzfreie und schnelle Deployments
- Einfachere Einführung neuer Features
- Mittelfristige Kostenersparnis

Stabilität & Kostenersparnis

Typische Vorbehalte

Beim Kunden

- Zu teuer
- Kein unmittelbarer Nutzen erkennbar

Typische Vorbehalte

Beim Kunden

- Zu teuer
- Kein unmittelbarer Nutzen erkennbar

Beim Entwickler

- Keine Notwendigkeit
- Keine Zeit
- Keine Lust

Welches Testing braucht man?

- Unittests
- Client-Tests ("In-Browser")
- Integrationstests/Systemtests

Django-Tests anlegen und ausführen

- nach "django-admin.py startapp foo": Test-Stub wird erzeugt
- django-admin.py test foo

Unittests

- Komponententests
- Pro Funktion ein Test
- Eigenständig, ohne Kontext

Unittests in Django

- Modelle
- Formulare
- Library-Module

Beispiel: Unittest für ein Modell

```python
from django.db import models

class MyModel(models.Model):
    name = models.CharField(max_length=255)

    def clean(self):
        from django.core.exceptions import ValidationError
        if not self.name.startswith('foo'):
            raise ValidationError('Name must start with foo')
```

```
>>> from myapp.models import *
>>> m = MyModel()
>>> m.name = 'bar'
>>> m.full_clean()
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "/Users/schneck/.virtualenvs/testing_dugc/lib/python2.6/site-packages/django/db/models/base.
py", line 824, in full_clean
    raise ValidationError(errors)
ValidationError: {'__all__': [u'Name must start with foo']}
```

Beispiel: Unittest für ein Modell

```python
from django.test import TestCase
from django.core.exceptions import ValidationError

from myapp.models import MyModel

class MyModelTest(TestCase):

    def test_name(self):
        m = MyModel(name='bar')
        self.assertRaises(ValidationError, m.full_clean)

        m = MyModel(name='foobar')
        m.full_clean()
```

```
$ ./manage.py test myapp
[...]

Creating test database for alias 'default'...
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
Destroying test database for alias 'default'...
```

Client-Tests

- Browser-Simulation
- Frontend-Tests
- Simulation von User-Interaktionen

Client-Tests in Django

- Allgemeine Tests (Views, Forms): Django TestClient
- Spezielle Tests (Javascript, Verhalten): Selenium
- django-selenium

Django-TestClient

```
<html>
<head>
    <title>My Testpage</title>
</head>
<body>
    <span id="result"></span>
</body>
</html>
```

Django-TestClient

```html
<html>
<head>
    <title>My Testpage</title>
</head>
<body>
    <span id="result"></span>
</body>
</html>
```

```python
from django.core.urlresolvers import reverse
from django.test import TestCase

class TestHome(TestCase):
    def test_home_index(self):
        response = self.client.get(reverse('home'))
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, '<span id="result"></span>')
```

Django-TestClient

```
<html>
<head>
    <title>My Testpage</title>
</head>
<body>
    <span id="result"></span>
</body>
</html>
```

```
from django.core.urlresolvers import reverse
from django.test import TestCase

class TestHome(TestCase):
    def test_home_index(self):
        response = self.client.get(reverse('home'))
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, '<span id="result"></span>')
```

```
$ ./manage.py test home
Creating test database for alias 'default'...
.
----------------------------------------------------------------------
Ran 1 test in 0.011s

OK
```

Selenium-Test 1: DOM

```
<html>
<head>
    <title>My Testpage</title>
</head>
<body>
    <span id="myspan">A Span</span>
</body>
</html>
```

Selenium-Test 1: DOM

```
<html>
<head>
    <title>My Testpage</title>
</head>
<body>
    <span id="myspan">A Span</span>
</body>
</html>
```

```
from django.core.urlresolvers import reverse
from django_selenium.testcases import SeleniumTestCase

class MyTestCase(SeleniumTestCase):

    def test_home(self):
        self.open_url(reverse('home'))
        self.failUnless(self.is_text_present('A Span'))
        self.assertEquals(self.get_title(), 'My Testpage')
```

Selenium-Test 1: DOM

```
<html>
<head>
    <title>My Testpage</title>
</head>
<body>
    <span id="myspan">A Span</span>
</body>
</html>
```

```
from django.core.urlresolvers import reverse
from django_selenium.testcases import SeleniumTestCase

class MyTestCase(SeleniumTestCase):

    def test_home(self):
        self.open_url(reverse('home'))
        self.failUnless(self.is_text_present('A Span'))
        self.assertEquals(self.get_title(), 'My Testpage')
```

```
$ ./manage.py test home --selenium
Creating test database for alias 'default'...
[13/Sep/2012 14:49:53] "GET / HTTP/1.1" 200 111
.
----------------------------------------------------------------------
Ran 1 test in 3.618s

OK
Destroying test database for alias 'default'...
```

Selenium-Test 2: JS

```
<html>
<body>
    <span id="result"></span>
    <button id="abutton" onclick="document.getElementById('result').innerHTML='myresult';">click me</button>
</body>
</html>
```

Selenium-Test 2: JS

```
<html>
<body>
    <span id="result"></span>
    <button id="abutton" onclick="document.getElementById('result').innerHTML='myresult';">click me</button>
</body>
</html>
```

```
from django.core.urlresolvers import reverse
from django_selenium.testcases import SeleniumTestCase

class MyTestCase(SeleniumTestCase):

    def test_home(self):
        self.open_url(reverse('home'))
        self.assertEqual(self.driver.get_text('#result'), '')
        self.driver.click('#abutton')
        self.assertEqual(self.driver.get_text('#result'), 'myresult')
```

Selenium-Test 2: JS

```
<html>
<body>
    <span id="result"></span>
    <button id="abutton" onclick="document.getElementById('result').innerHTML='myresult';">click me</button>
</body>
</html>
```

```
from django.core.urlresolvers import reverse
from django_selenium.testcases import SeleniumTestCase

class MyTestCase(SeleniumTestCase):

    def test_home(self):
        self.open_url(reverse('home'))
        self.assertEqual(self.driver.get_text('#result'), '')
        self.driver.click('#abutton')
        self.assertEqual(self.driver.get_text('#result'), 'myresult')
```

```
$ ./manage.py test home --selenium
[..]
----------------------------------------------------------------------
Ran 1 test in 4.778s

OK
Destroying test database for alias 'default'...
```

Integrationstests

- z.B. Kombination von Unittests und Client-Tests
- sehr nützlich: Jenkins

Integrationstests

- Beispiel: HTML-Uploader
  - 1) Anzeige Upload-Form
  - 2) Storage
  - 3) Hash-Prüfung
  - 4) Download
  - 5) View-Test
  - 6) API-Test

Alternative Testrunner

- Djangos Testrunner kann einfach getauscht werden
- nose, unittest2

Alternative TestClients

- WebTest (HTML-Parsing)

Performance

- factory_boy
- Keine Fixtures
- Wenig Datenbankzugriffe

Happy Testing!

Fragen?