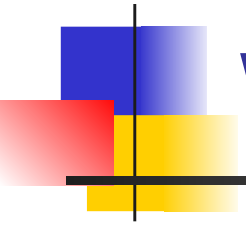


Web Application Testing in Python

James Brucker

Guidance

What to Test?

- Logic
- Flow Control
- Application Flow, e.g. Page Flow
- Configuration

"Don't test Constants" (e.g. HTML template text)

Test your code (not the framework)

What is Being Tested?

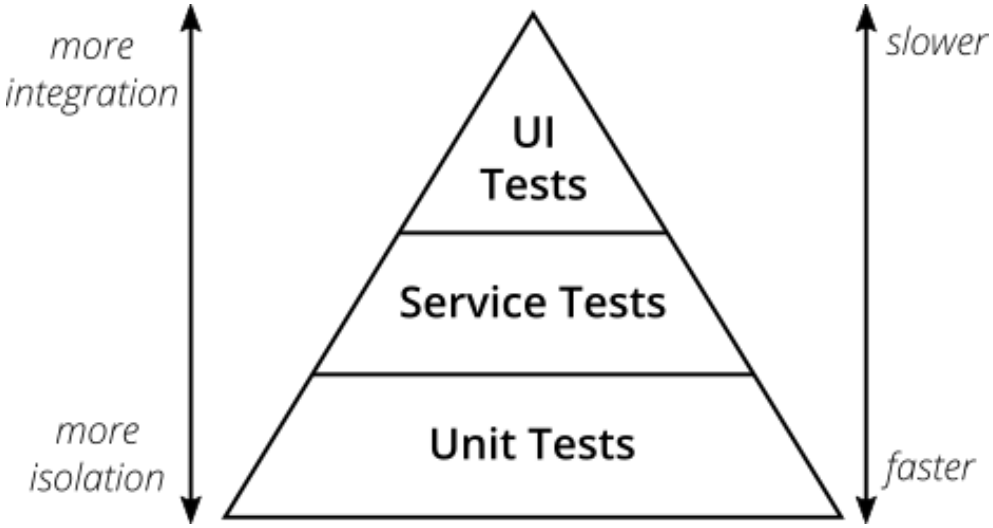
```
import django.test
from polls.models import Question

class QuestionTest(django.test.TestCase):
    def setUp(self):
        Question.objects.create(
                question_text="Question One")
        Question.objects.create(
                question_text="Question Two")

    def test_create_questions(self):
        self.assertEqual(2, Questions.objects.count())
```

What is Being Tested? (cont'd)

```
    def test_question_text(self):
        self.assertTrue(
            any("Question One" in q.question_text
                for q in Questions.objects.all() )
```

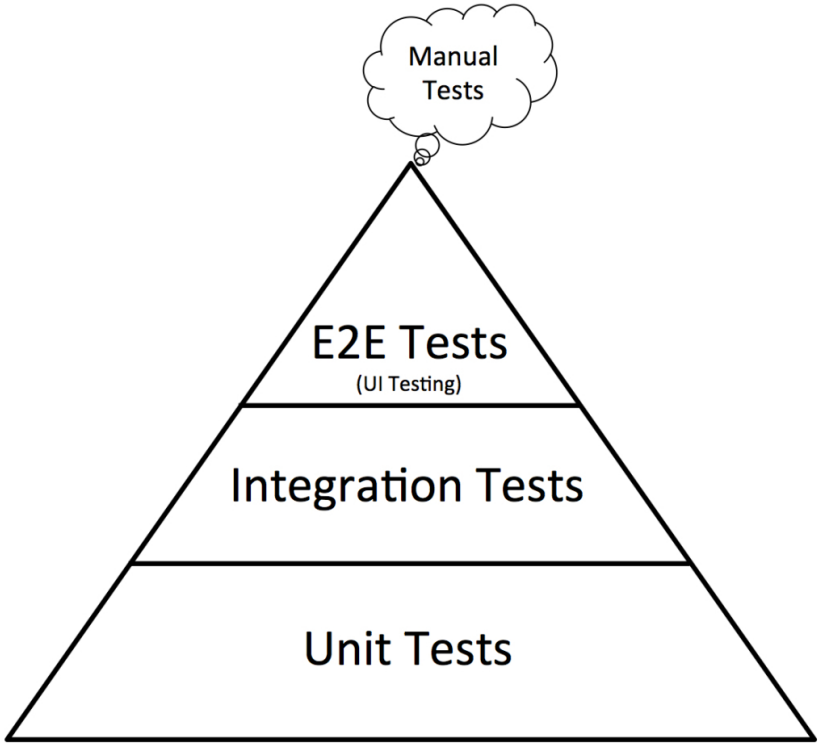
Are Unit Tests Enough?

The Testing Pyramid



Mike Cohen's original Pyramid

"Practical" pyramid



Integration Testing

Test the interaction between components.

- Components belonging to your app.
- Back-end services called by front-end
- External components and web services.

Integration tests often access a "service layer", or your standard URLs.

Integration Testing

`django.test.Client` is useful for testing URLs

```
$ python manage.py shell
>>> from django.test import Client
>>> c = Client()
# Get the /polls/ page. Should contain some polls
>>> response = c.get('/polls/')
# Did it succeed?
>>> response.status_code
200
# Print the html content
>>> response.content
'<html>\n<head>\n<style>...\n<h1>Active Polls</h1>...
```

Integration Testing in Django

django.test.Client tests how the application handles requests.

Can test:

1. invoke urls with query parameters, e.g.

```
# create a new poll

client.post('/polls/', {'text': 'Where is God?'})
```

2. different HTTP methods (POST, PUT, ...)

3. what template was used to create a response

What to Test?

External services - web services

Integration of "apps"

- test all URLs used by "front-end"
- test authorized/unauthorized access
- integration of multiple projects
- database - since Django has built-in ORM, this can be done by unit tests.

E2E Testing Tools

Selenium - widely used, standard tool.

- Django has built-in support.
- can write tests in any language, incl. Python, Java
- Selenium IDE for creating tests in a web browser

Cypress.io - Javascript testing tool. Natively interacts with pages in your application.

- uses Mocha and Chai for writing tests
- tests written in Javascript

Puppeteer - library for controlling a "headless" Chrome browser. Uses Javascript and node.js.

- many uses: page scraping, web crawling, testing

Cypress Example

Simple test:

```
describe('My First Test', function() {
    it('Visits the Kitchen Sink', function() {
        cy.visit('https://example.cypress.io')
    })
})
```

Retrieving page elements:

https://docs.cypress.io/guides/core-concepts/introduction-to-cypress.html#Querying-Elements

Test Recipes:

https://docs.cypress.io/examples/examples/recipes.html

References

The Practical Test Pyramid

https://martinfowler.com/articles/practical-test-pyramid.html

TDD in Python (online book)

Testing the Github Public API

developer.github.com/v3/users/