
Table of Contents

Preface.....	xv
Prerequisites and Assumptions.....	xxi
Companion Video.....	xxxii
Acknowledgments.....	xxxiii
Part I. The Basics of TDD and Django	
1. Getting Django Set Up Using a Functional Test.....	3
Obey the Testing Goat! Do Nothing Until You Have a Test	3
Getting Django Up and Running	6
Starting a Git Repository	8
2. Extending Our Functional Test Using the unittest Module.....	13
Using a Functional Test to Scope Out a Minimum Viable App	14
The Python Standard Library's unittest Module	16
Commit	19
3. Testing a Simple Home Page with Unit Tests.....	21
Our First Django App, and Our First Unit Test	22
Unit Tests, and How They Differ from Functional Tests	22
Unit Testing in Django	23
Django's MVC, URLs, and View Functions	25
At Last! We Actually Write Some Application Code!	26
urls.py	28

Unit Testing a View	30
The Unit-Test/Code Cycle	32
4. What Are We Doing with All These Tests? (And, Refactoring).....	37
Programming Is Like Pulling a Bucket of Water Up from a Well	38
Using Selenium to Test User Interactions	40
The “Don’t Test Constants” Rule, and Templates to the Rescue	43
Refactoring to Use a Template	43
The Django Test Client	47
On Refactoring	49
A Little More of Our Front Page	50
Recap: The TDD Process	52
5. Saving User Input: Testing the Database.	55
Wiring Up Our Form to Send a POST Request	55
Processing a POST Request on the Server	59
Passing Python Variables to Be Rendered in the Template	60
Three Strikes and Refactor	64
The Django ORM and Our First Model	66
Our First Database Migration	68
The Test Gets Surprisingly Far	69
A New Field Means a New Migration	69
Saving the POST to the Database	70
Redirect After a POST	74
Better Unit Testing Practice: Each Test Should Test One Thing	75
Rendering Items in the Template	75
Creating Our Production Database with migrate	78
Recap	80
6. Improving Functional Tests: Ensuring Isolation and Removing Voodoo Sleeps.	83
Ensuring Test Isolation in Functional Tests	83
Running Just the Unit Tests	87
Aside: Upgrading Selenium and Geckodriver	88
On Implicit and Explicit Waits, and Voodoo time.sleeps	89
7. Working Incrementally.....	95
Small Design When Necessary	95
Not Big Design Up Front	95
YAGNI!	96
REST (ish)	96
Implementing the New Design Incrementally Using TDD	97
Ensuring We Have a Regression Test	99

Iterating Towards the New Design	101
Taking a First, Self-Contained Step: One New URL	103
A New URL	103
A New View Function	104
Green? Refactor	106
Another Small Step: A Separate Template for Viewing Lists	107
A Third Small Step: A URL for Adding List Items	109
A Test Class for New List Creation	109
A URL and View for New List Creation	110
Removing Now-Redundant Code and Tests	111
A Regression! Pointing Our Forms at the New URL	112
Biting the Bullet: Adjusting Our Models	113
A Foreign Key Relationship	115
Adjusting the Rest of the World to Our New Models	117
Each List Should Have Its Own URL	119
Capturing Parameters from URLs	120
Adjusting new_list to the New World	121
The Functional Tests Detect Another Regression	123
One More View to Handle Adding Items to an Existing List	124
Beware of Greedy Regular Expressions!	125
The Last New URL	125
The Last New View	126
Testing the Response Context Objects Directly	127
A Final Refactor Using URL includes	129

Part II. Web Development *Sine Qua Nons*

8. Prettification: Layout and Styling, and What to Test About It.	135
What to Functionally Test About Layout and Style	135
Prettification: Using a CSS Framework	139
Django Template Inheritance	141
Integrating Bootstrap	142
Rows and Columns	143
Static Files in Django	144
Switching to StaticLiveServerTestCase	145
Using Bootstrap Components to Improve the Look of the Site	146
Jumbotron!	146
Large Inputs	147
Table Styling	147
Using Our Own CSS	147
What We Glossed Over: collectstatic and Other Static Directories	149

A Few Things That Didn't Make It	152
9. Testing Deployment Using a Staging Site.	155
TDD and the Danger Areas of Deployment	156
As Always, Start with a Test	157
Getting a Domain Name	160
Manually Provisioning a Server to Host Our Site	160
Choosing Where to Host Our Site	160
Spinning Up a Server	161
User Accounts, SSH, and Privileges	162
Installing Nginx	162
Installing Python 3.6	163
Configuring Domains for Staging and Live	163
Using the FT to Confirm the Domain Works and Nginx Is Running	164
Deploying Our Code Manually	164
Adjusting the Database Location	165
Creating a Virtualenv Manually, and Using requirements.txt	167
Simple Nginx Configuration	168
Creating the Database with migrate	171
Success! Our Hack Deployment Works	173
10. Getting to a Production-Ready Deployment.	175
Switching to Gunicorn	175
Getting Nginx to Serve Static Files	177
Switching to Using Unix Sockets	177
Switching DEBUG to False and Setting ALLOWED_HOSTS	178
Using Systemd to Make Sure Gunicorn Starts on Boot	179
Saving Our Changes: Adding Gunicorn to Our requirements.txt	180
Thinking About Automating	181
Saving Templates for Our Provisioning Config Files	181
Saving Our Progress	184
11. Automating Deployment with Fabric.	187
Breakdown of a Fabric Script for Our Deployment	188
Creating the Directory Structure	189
Pulling Down Our Source Code with Git	189
Updating settings.py	190
Updating the Virtualenv	192
Migrating the Database If Necessary	192
Trying It Out	193
Deploying to Live	194
Nginx and Gunicorn Config Using sed	196

Git Tag the Release	199
Further Reading	199
12. Splitting Our Tests into Multiple Files, and a Generic Wait Helper.....	201
Start on a Validation FT: Preventing Blank Items	201
Skipping a Test	202
Splitting Functional Tests Out into Many Files	203
Running a Single Test File	206
A New Functional Test Tool: A Generic Explicit Wait Helper	207
Finishing Off the FT	211
Refactoring Unit Tests into Several Files	213
13. Validation at the Database Layer.....	217
Model-Layer Validation	218
The self.assertRaises Context Manager	218
A Django Quirk: Model Save Doesn't Run Validation	219
Surfacing Model Validation Errors in the View	220
Checking That Invalid Input Isn't Saved to the Database	223
Django Pattern: Processing POST Requests in the Same View as Renders the Form	225
Refactor: Transferring the new_item Functionality into view_list	226
Enforcing Model Validation in view_list	229
Refactor: Removing Hardcoded URLs	231
The {% url %} Template Tag	231
Using get_absolute_url for Redirects	232
14. A Simple Form.....	235
Moving Validation Logic into a Form	235
Exploring the Forms API with a Unit Test	236
Switching to a Django ModelForm	238
Testing and Customising Form Validation	239
Using the Form in Our Views	241
Using the Form in a View with a GET Request	241
A Big Find and Replace	242
Using the Form in a View That Takes POST Requests	245
Adapting the Unit Tests for the new_list View	245
Using the Form in the View	246
Using the Form to Display Errors in the Template	247
Using the Form in the Other View	248
A Helper Method for Several Short Tests	248
An Unexpected Benefit: Free Client-Side Validation from HTML5	251
A Pat on the Back	253

But Have We Wasted a Lot of Time?	254
Using the Form's Own Save Method	254
15. More Advanced Forms.....	259
Another FT for Duplicate Items	259
Preventing Duplicates at the Model Layer	260
A Little Digression on Queryset Ordering and String Representations	263
Rewriting the Old Model Test	265
Some Integrity Errors Do Show Up on Save	267
Experimenting with Duplicate Item Validation at the Views Layer	268
A More Complex Form to Handle Uniqueness Validation	269
Using the Existing List Item Form in the List View	271
Wrapping Up: What We've Learned About Testing Django	273
16. Dipping Our Toes, Very Tentatively, into JavaScript.....	277
Starting with an FT	277
Setting Up a Basic JavaScript Test Runner	279
Using jQuery and the Fixtures Div	281
Building a JavaScript Unit Test for Our Desired Functionality	285
Fixtures, Execution Order, and Global State: Key Challenges of JS Testing	287
console.log for Debug Printing	287
Using an Initialize Function for More Control Over Execution Time	289
Columbo Says: Onload Boilerplate and Namespacing	291
JavaScript Testing in the TDD Cycle	292
A Few Things That Didn't Make It	293
17. Deploying Our New Code.....	295
Staging Deploy	295
Live Deploy	296
What to Do If You See a Database Error	296
Wrap-Up: git tag the New Release	296

Part III. More Advanced Topics in Testing

18. User Authentication, Spiking, and De-Spiking.....	301
Passwordless Auth	302
Exploratory Coding, aka "Spiking"	302
Starting a Branch for the Spike	303
Frontend Log in UI	303
Sending Emails from Django	304
Using Environment Variables to Avoid Secrets in Source Code	306

Storing Tokens in the Database	307
Custom Authentication Models	307
Finishing the Custom Django Auth	309
De-spiking	313
Reverting Our Spiked Code	315
A Minimal Custom User Model	316
Tests as Documentation	319
A Token Model to Link Emails with a Unique ID	320
19. Using Mocks to Test External Dependencies or Reduce Duplication.	323
Before We Start: Getting the Basic Plumbing In	323
Mocking Manually, aka Monkeypatching	324
The Python Mock Library	328
Using unittest.patch	328
Getting the FT a Little Further Along	331
Testing the Django Messages Framework	331
Adding Messages to Our HTML	333
Starting on the Login URL	334
Checking That We Send the User a Link with a Token	335
De-spiking Our Custom Authentication Backend	337
1 if = 1 More Test	338
The get_user Method	341
Using Our Auth Backend in the Login View	343
An Alternative Reason to Use Mocks: Reducing Duplication	344
Using mock.return_value	347
Patching at the Class Level	349
The Moment of Truth: Will the FT Pass?	351
It Works in Theory! Does It Work in Practice?	353
Finishing Off Our FT, Testing Logout	354
20. Test Fixtures and a Decorator for Explicit Waits.	359
Skipping the Login Process by Pre-creating a Session	360
Checking That It Works	362
Our Final Explicit Wait Helper: A Wait Decorator	364
21. Server-Side Debugging.	369
The Proof Is in the Pudding: Using Staging to Catch Final Bugs	369
Setting Up Logging	370
Setting Secret Environment Variables on the Server	372
Adapting Our FT to Be Able to Test Real Emails via POP3	372
Managing the Test Database on Staging	376
A Django Management Command to Create Sessions	376

Getting the FT to Run the Management Command on the Server	378
Using Fabric Directly from Python	379
Recap: Creating Sessions Locally Versus Staging	380
Baking In Our Logging Code	382
Wrap-Up	382
22. Finishing “My Lists”: Outside-In TDD.	385
The Alternative: “Inside-Out”	385
Why Prefer “Outside-In”?	386
The FT for “My Lists”	386
The Outside Layer: Presentation and Templates	389
Moving Down One Layer to View Functions (the Controller)	390
Another Pass, Outside-In	391
A Quick Restructure of the Template Inheritance Hierarchy	391
Designing Our API Using the Template	392
Moving Down to the Next Layer: What the View Passes to the Template	393
The Next “Requirement” from the Views Layer: New Lists Should Record Owner	394
A Decision Point: Whether to Proceed to the Next Layer with a Failing Test	395
Moving Down to the Model Layer	396
Final Step: Feeding Through the .name API from the Template	398
23. Test Isolation, and “Listening to Your Tests”	401
Revisiting Our Decision Point: The Views Layer Depends on Unwritten Models Code	401
A First Attempt at Using Mocks for Isolation	402
Using Mock side_effects to Check the Sequence of Events	404
Listen to Your Tests: Ugly Tests Signal a Need to Refactor	406
Rewriting Our Tests for the View to Be Fully Isolated	406
Keep the Old Integrated Test Suite Around as a Sanity Check	406
A New Test Suite with Full Isolation	407
Thinking in Terms of Collaborators	407
Moving Down to the Forms Layer	412
Keep Listening to Your Tests: Removing ORM Code from Our Application	413
Finally, Moving Down to the Models Layer	416
Back to Views	419
The Moment of Truth (and the Risks of Mocking)	420
Thinking of Interactions Between Layers as “Contracts”	421
Identifying Implicit Contracts	422
Fixing the Oversight	424
One More Test	425

Tidy Up: What to Keep from Our Integrated Test Suite	426
Removing Redundant Code at the Forms Layer	426
Removing the Old Implementation of the View	427
Removing Redundant Code at the Forms Layer	429
Conclusions: When to Write Isolated Versus Integrated Tests	429
Let Complexity Be Your Guide	430
Should You Do Both?	431
Onwards!	431
24. Continuous Integration (CI).....	433
Installing Jenkins	433
Configuring Jenkins	434
Initial Unlock	435
Suggested Plugins for Now	435
Configuring the Admin User	435
Adding Plugins	437
Telling Jenkins Where to Find Python 3 and Xvfb	437
Finishing Off with HTTPS	438
Setting Up Our Project	438
First Build!	439
Setting Up a Virtual Display So the FTs Can Run Headless	441
Taking Screenshots	443
If in Doubt, Try Bumping the Timeout!	447
Running Our QUnit JavaScript Tests in Jenkins with PhantomJS	448
Installing node	448
Adding the Build Steps to Jenkins	449
More Things to Do with a CI Server	451
25. The Token Social Bit, the Page Pattern, and an Exercise for the Reader.....	453
An FT with Multiple Users, and addCleanup	453
The Page Pattern	455
Extend the FT to a Second User, and the “My Lists” Page	458
An Exercise for the Reader	459
26. Fast Tests, Slow Tests, and Hot Lava.....	463
Thesis: Unit Tests Are Superfast and Good Besides That	464
Faster Tests Mean Faster Development	465
The Holy Flow State	465
Slow Tests Don’t Get Run as Often, Which Causes Bad Code	465
We’re Fine Now, but Integrated Tests Get Slower Over Time	465
Don’t Take It from Me	466
And Unit Tests Drive Good Design	466

The Problems with “Pure” Unit Tests	466
Isolated Tests Can Be Harder to Read and Write	466
Isolated Tests Don’t Automatically Test Integration	466
Unit Tests Seldom Catch Unexpected Bugs	466
Mocky Tests Can Become Closely Tied to Implementation	467
But All These Problems Can Be Overcome	467
Synthesis: What Do We Want from Our Tests, Anyway?	467
Correctness	467
Clean, Maintainable Code	467
Productive Workflow	468
Evaluate Your Tests Against the Benefits You Want from Them	468
Architectural Solutions	469
Ports and Adapters/Hexagonal/Clean Architecture	469
Functional Core, Imperative Shell	470
Conclusion	470
Further Reading	471
Obey the Testing Goat!.....	473
A. PythonAnywhere.....	475
B. Django Class-Based Views.....	479
C. Provisioning with Ansible.....	491
D. Testing Database Migrations.....	497
E. Behaviour-Driven Development (BDD).....	503
F. Building a REST API: JSON, Ajax, and Mocking with JavaScript.....	519
G. Django-Rest-Framework.....	541
H. Cheat Sheet.....	553
I. What to Do Next.....	557
J. Source Code Examples.....	561
Bibliography.....	565
Index.....	567