

T3 PYT 6AE  
WD 4884

# Two Scoops of Django

*Best Practices For Django 1.6*

**Daniel Greenfeld**  
**Audrey Roy**



ary

No.  
signout

4

# Contents

<b>Dedication</b>	<b>iii</b>
<b>List of Figures</b>	<b>xxi</b>
<b>List of Tables</b>	<b>xxiv</b>
<b>Authors' Notes</b>	<b>xxvii</b>
About the Dedication . . . . .	xxvii
A Few Words From Daniel Greenfeld . . . . .	xxviii
A Few Words From Audrey Roy . . . . .	xxix
<b>Introduction</b>	<b>xxxi</b>
A Word About Our Recommendations . . . . .	xxxi
Why Two Scoops of Django? . . . . .	xxxii
Before You Begin . . . . .	xxxiii
This book is intended for and Python 2.7.x/3.3.x . . . . .	xxxiii
Each Chapter Stands On Its Own . . . . .	xxxiii
Conventions Used in This Book . . . . .	xxxiv
Core Concepts . . . . .	xxxv
Keep It Simple, Stupid . . . . .	xxxv
Fat Models, Helper Modules, Thin Views, Stupid Templates . . . . .	xxxvi
Start With Django By Default . . . . .	xxxvi
Be Familiar with Django's Design Philosophies . . . . .	xxxvi
The Twelve Factor App . . . . .	xxxvii
Our Writing Concepts . . . . .	xxxvii
Provide the Best Material . . . . .	xxxvii
Stand on the Shoulders of Giants . . . . .	xxxvii
Listen to Our Readers and Reviewers . . . . .	xxxviii

---

Publish Errata . . . . .	xxxviii
<b>1 Coding Style</b>	<b>1</b>
1.1 The Importance of Making Your Code Readable . . . . .	1
1.2 PEP 8 . . . . .	2
1.2.1 The 79 Character Limit . . . . .	2
1.3 The Word on Imports . . . . .	3
1.4 Use Explicit Relative Imports . . . . .	4
1.5 Avoid Using Import * . . . . .	7
1.6 Django Coding Style Guidelines . . . . .	9
1.7 Never Code to the IDE (or Text Editor) . . . . .	9
1.8 Summary . . . . .	9
<b>2 The Optimal Django Environment Setup</b>	<b>11</b>
2.1 Use the Same Database Engine Everywhere . . . . .	11
2.1.1 Fixtures Are Not a Magic Solution . . . . .	11
2.1.2 You Can't Examine an Exact Copy of Production Data Locally . . . . .	12
2.1.3 Different Databases Have Different Field Types/Constraints . . . . .	12
2.2 Use Pip and Virtualenv . . . . .	13
2.3 Install Django and Other Dependencies via Pip . . . . .	15
2.4 Use a Version Control System . . . . .	16
2.5 Consider Using Vagrant for Development Setup . . . . .	16
2.6 Summary . . . . .	17
<b>3 How to Lay Out Django Projects</b>	<b>19</b>
3.1 Django 1.6's Default Project Layout . . . . .	19
3.2 Our Preferred Project Layout . . . . .	20
3.2.1 Top Level: Repository Root . . . . .	21
3.2.2 Second Level: Django Project Root . . . . .	21
3.2.3 Third Level: Configuration Root . . . . .	21
3.3 Sample Project Layout . . . . .	22
3.4 What About the Virtualenv? . . . . .	25
3.5 Using a Startproject Template to Generate Our Layout . . . . .	26
3.6 Other Alternatives . . . . .	27
3.7 Summary . . . . .	27
<b>4 Fundamentals of Django App Design</b>	<b>29</b>

4.1	The Golden Rule of Django App Design . . . . .	30
4.1.1	A Practical Example of Apps in a Project . . . . .	31
4.2	What to Name Your Django Apps . . . . .	32
4.3	When in Doubt, Keep Apps Small . . . . .	33
4.4	Summary . . . . .	33
<b>5</b>	<b>Settings and Requirements Files</b>	<b>35</b>
5.1	Avoid Non-Versioned Local Settings . . . . .	36
5.2	Using Multiple Settings Files . . . . .	37
5.2.1	A Development Settings Example . . . . .	40
5.2.2	Multiple Development Settings . . . . .	41
5.3	Separate Configuration from Code . . . . .	42
5.3.1	A Caution Before Using Environment Variables for Secrets . . . . .	43
5.3.2	How to Set Environment Variables Locally . . . . .	44
5.3.3	How to Set Environment Variables in Production . . . . .	45
5.3.4	Handling Missing Secret Key Exceptions . . . . .	46
5.4	When You Can't Use Environment Variables . . . . .	48
5.4.1	Using JSON Files . . . . .	48
5.4.2	Using Config, YAML, and XML File Formats . . . . .	50
5.5	Using Multiple Requirements Files . . . . .	50
5.5.1	Installing From Multiple Requirements Files . . . . .	51
5.5.2	Using Multiple Requirements Files With Platforms as a Service (PaaS) . . . . .	52
5.6	Handling File Paths in Settings . . . . .	52
5.7	Summary . . . . .	55
<b>6</b>	<b>Database/Model Best Practices</b>	<b>57</b>
6.1	Basics . . . . .	58
6.1.1	Break Up Apps With Too Many Models . . . . .	58
6.1.2	Don't Drop Down to Raw SQL Until It's Necessary . . . . .	58
6.1.3	Add Indexes as Needed . . . . .	59
6.1.4	Be Careful With Model Inheritance . . . . .	60
6.1.5	Model Inheritance in Practice: The TimeStampedModel . . . . .	62
6.1.6	Use South for Migrations . . . . .	63
6.2	Django Model Design . . . . .	65
6.2.1	Start Normalized . . . . .	65
6.2.2	Cache Before Denormalizing . . . . .	65

6.2.3	Denormalize Only if Absolutely Needed . . . . .	66
6.2.4	When to Use Null and Blank . . . . .	66
6.2.5	When to Use BinaryField . . . . .	68
6.3	Model Managers . . . . .	69
6.4	Transactions . . . . .	71
6.4.1	Wrapping Each HTTP Request In a Transaction . . . . .	72
6.4.2	Explicit Transaction Declaration . . . . .	74
6.4.3	django.http.StreamingHttpResponse and Transactions . . . . .	75
6.4.4	Transactions in MySQL . . . . .	76
6.4.5	Django ORM Transaction Resources . . . . .	76
6.5	Summary . . . . .	76
<b>7</b>	<b>Function- and Class-Based Views</b>	<b>79</b>
7.1	When to Use FBVs or CBVs . . . . .	79
7.2	Keep View Logic Out of URLConfs . . . . .	80
7.3	Stick to Loose Coupling in URLConfs . . . . .	82
7.3.1	What If We Aren't Using CBVs? . . . . .	85
7.4	Use URL Namespaces . . . . .	85
7.4.1	Makes for Shorter, More Obvious and Don't Repeat Yourself URL names . . . . .	86
7.4.2	Increases Interoperability with Third-Party Libraries . . . . .	87
7.4.3	Easier Searches, Upgrades, and Refactors . . . . .	88
7.4.4	Allows for More App and Template Reverse Tricks . . . . .	88
7.5	Try to Keep Business Logic Out of Views . . . . .	88
7.6	Django Views are Functions . . . . .	89
7.6.1	The Simplest Views . . . . .	89
7.7	Summary . . . . .	90
<b>8</b>	<b>Best Practices for Function-Based Views</b>	<b>91</b>
8.1	Advantages of FBVs . . . . .	91
8.2	Passing the HttpRequest Object . . . . .	92
8.3	Decorators are Sweet . . . . .	95
8.3.1	Be Conservative with Decorators . . . . .	97
8.3.2	Additional Resources on Decorators . . . . .	98
8.4	Passing the HttpResponse object . . . . .	98
8.5	Summary . . . . .	98

<b>9</b>	<b>Best Practices for Class-Based Views</b>	<b>99</b>
9.1	Guidelines When Working With CBVs . . . . .	100
9.2	Using Mixins With CBVs . . . . .	100
9.3	Which Django GCBV Should Be Used for What Task? . . . . .	102
9.4	General Tips for Django CBVs . . . . .	103
9.4.1	Constraining Django CBV/GCBV Access to Authenticated Users . . . . .	104
9.4.2	Performing Custom Actions on Views With Valid Forms . . . . .	104
9.4.3	Performing Custom Actions on Views With Invalid Forms . . . . .	105
9.4.4	Using the View Object . . . . .	106
9.5	How GCBVs and Forms Fit Together . . . . .	108
9.5.1	Views + ModelForm Example . . . . .	109
9.5.2	Views + Form Example . . . . .	113
9.6	Using Just <code>django.views.generic.View</code> . . . . .	115
9.7	Additional Resources . . . . .	117
9.8	Summary . . . . .	118
<b>10</b>	<b>Common Patterns for Forms</b>	<b>119</b>
10.1	The Power of Django Forms . . . . .	119
10.2	Pattern 1: Simple ModelForm With Default Validators . . . . .	120
10.3	Pattern 2: Custom Form Field Validators in ModelForms . . . . .	121
10.4	Pattern 3: Overriding the Clean Stage of Validation . . . . .	126
10.5	Pattern 4: Hacking Form Fields (2 CBVs, 2 Forms, 1 Model) . . . . .	129
10.6	Pattern 5: Reusable Search Mixin View . . . . .	133
10.7	Summary . . . . .	135
<b>11</b>	<b>More Things to Know About Forms</b>	<b>137</b>
11.1	Use the POST Method in HTML Forms . . . . .	137
11.1.1	Don't Disable Django's CSRF Protection . . . . .	138
11.2	Know How Form Validation Works . . . . .	138
11.2.1	Form Data Is Saved to the Form, Then the Model Instance . . . . .	139
11.3	Summary . . . . .	141
<b>12</b>	<b>Templates: Best Practices</b>	<b>143</b>
12.1	Follow a Minimalist Approach . . . . .	143
12.2	Template Architecture Patterns . . . . .	144
12.2.1	2-Tier Template Architecture Example . . . . .	144

---

12.2.2	3-Tier Template Architecture Example . . . . .	144
12.2.3	Flat Is Better Than Nested . . . . .	145
12.3	Limit Processing in Templates . . . . .	146
12.3.1	Gotcha 1: Aggregation in Templates . . . . .	148
12.3.2	Gotcha 2: Filtering With Conditionals in Templates . . . . .	150
12.3.3	Gotcha 3: Complex Implied Queries in Templates . . . . .	152
12.3.4	Gotcha 4: Hidden CPU Load in Templates . . . . .	154
12.3.5	Gotcha 5: Hidden REST API Calls in Templates . . . . .	154
12.4	Don't Bother Making Your Generated HTML Pretty . . . . .	155
12.5	Exploring Template Inheritance . . . . .	156
12.6	block.super Gives the Power of Control . . . . .	159
12.7	Useful Things to Consider . . . . .	161
12.7.1	Avoid Coupling Styles Too Tightly to Python Code . . . . .	161
12.7.2	Common Conventions . . . . .	162
12.7.3	Location, Location, Location! . . . . .	162
12.7.4	Use Named Context Objects . . . . .	162
12.7.5	Use URL Names Instead of Hardcoded Paths . . . . .	163
12.7.6	Debugging Complex Templates . . . . .	164
12.7.7	Don't Replace the Django Template Engine . . . . .	164
12.8	Error Page Templates . . . . .	164
12.9	Summary . . . . .	165
<b>13</b>	<b>Template Tags and Filters</b> . . . . .	<b>167</b>
13.1	Filters Are Functions . . . . .	167
13.1.1	Filters Are Easy to Test . . . . .	168
13.1.2	Filters, Code Reuse, and Performance . . . . .	168
13.1.3	When to Write Filters . . . . .	169
13.2	Custom Template Tags . . . . .	169
13.2.1	Template Tags Are Harder To Debug . . . . .	169
13.2.2	Template Tags Make Code Reuse Harder . . . . .	169
13.2.3	The Performance Cost of Template Tags . . . . .	169
13.2.4	When to Write Template Tags . . . . .	170
13.3	Naming Your Template Tag Libraries . . . . .	171
13.4	Loading Your Template Tag Modules . . . . .	171
13.4.1	Watch Out for This Crazy Anti-Pattern . . . . .	172
13.5	Summary . . . . .	172

---

<b>14 Building REST APIs</b>	<b>175</b>
14.1 Fundamentals of Basic REST API Design . . . . .	176
14.2 Implementing a Simple JSON API . . . . .	178
14.3 REST API Architecture . . . . .	180
14.3.1 Code for an App Should Remain in the App . . . . .	180
14.3.2 Try to Keep Business Logic Out of API Views . . . . .	180
14.3.3 Grouping API URLs . . . . .	181
14.3.4 Test Your API . . . . .	183
14.3.5 Version Your API . . . . .	183
14.4 Evaluating REST Frameworks . . . . .	183
14.4.1 How Much Boilerplate Do You Want to Write? . . . . .	184
14.4.2 Are Remote Procedure Calls Easy to Implement? . . . . .	184
14.4.3 CBVs or FBVs? . . . . .	184
14.5 Additional Reading . . . . .	185
14.6 Summary . . . . .	185
<b>15 Consuming REST APIs in Templates</b>	<b>187</b>
15.1 Learn How to Debug the Client . . . . .	188
15.2 Consider Using JavaScript-Powered Static Asset Preprocessors . . . . .	188
15.3 Making Content Indexable by Search Engines . . . . .	189
15.3.1 Hand-Craft the sitemap.xml . . . . .	189
15.3.2 Use a Service to Make Your Site Crawlable . . . . .	190
15.3.3 Wait for Search Engines to Figure It Out . . . . .	190
15.4 Real-Time Woes a.k.a. Latency . . . . .	190
15.4.1 Solution: Mask the Latency with Animations . . . . .	191
15.4.2 Solution: Fake Successful Transactions . . . . .	191
15.4.3 Solution: Geographically Based Servers . . . . .	191
15.4.4 Solution: Restrict Users Geographically . . . . .	192
15.5 Avoid the Anti-Patterns . . . . .	192
15.5.1 Building Single Page Apps When Multi-Page Apps Suffice . . . . .	192
15.5.2 Not Writing Tests . . . . .	192
15.5.3 Not Understanding JavaScript Memory Management . . . . .	193
15.5.4 Storing Data in the DOM When It's Not jQuery . . . . .	193
15.6 AJAX and the CSRF Token . . . . .	193
15.6.1 Additional reading . . . . .	195
15.7 Improving JavaScript Skills . . . . .	195



15.7.1	Assessing Skill Levels . . . . .	195
15.7.2	Learn More JavaScript! . . . . .	195
15.8	Follow Javascript Coding Standards . . . . .	195
15.9	Django and Javascript Tutorials . . . . .	196
15.10	Summary . . . . .	196
<b>16</b>	<b>Tradeoffs of Replacing Core Components</b>	<b>197</b>
16.1	The Temptation to Build FrankenDjango . . . . .	198
16.2	Case Study: Replacing the Django Template Engine . . . . .	199
16.2.1	Excuses, Excuses . . . . .	199
16.2.2	What if I'm Hitting the Limits of Templates? . . . . .	199
16.2.3	What About My Unusual Use Case? . . . . .	200
16.3	Non-Relational Databases vs. Relational Databases . . . . .	201
16.3.1	Understand ACID . . . . .	201
16.3.2	Don't Use Non-Relational Databases for Relational Tasks . . . . .	202
16.3.3	Ignore the Hype and Do Your Own Research . . . . .	202
16.3.4	How We Use Non-Relational Databases With Django . . . . .	202
16.4	Summary . . . . .	203
<b>17</b>	<b>Working With the Django Admin</b>	<b>205</b>
17.1	It's Not for End Users . . . . .	206
17.2	Admin Customization vs. New Views . . . . .	206
17.3	Viewing String Representations of Objects . . . . .	206
17.4	Adding Callables to ModelAdmin Classes . . . . .	210
17.5	Don't Use list_editable in Multiuser Environments . . . . .	211
17.6	Django's Admin Documentation Generator . . . . .	212
17.7	Securing the Django Admin and Django Admin Docs . . . . .	213
17.8	Using Custom Skins with the Django Admin . . . . .	213
17.8.1	Evaluation Point: Documentation is Everything . . . . .	214
17.8.2	Write Tests for Any Admin Extensions You Create . . . . .	214
17.9	Summary . . . . .	215
<b>18</b>	<b>Dealing With the User Model</b>	<b>217</b>
18.1	Use Django's Tools for Finding the User Model . . . . .	217
18.1.1	Use settings.AUTH_USER_MODEL for Foreign Keys to User . . . . .	218
18.1.2	Don't Use get_user_model() for Foreign Keys to User . . . . .	218

18.2	Migrating Pre-1.5 User models to 1.5+'s Custom User Models . . . . .	219
18.3	Custom User Fields for Django 1.6 Projects . . . . .	219
18.3.1	Option 1: Subclass AbstractUser . . . . .	220
18.3.2	Option 2: Subclass AbstractBaseUser . . . . .	221
18.3.3	Option 3: Linking Back From a Related Model . . . . .	221
18.4	Summary . . . . .	223
<b>19</b>	<b>Django's Secret Sauce: Third-Party Packages</b> . . . . .	<b>225</b>
19.1	Examples of Third-Party Packages . . . . .	226
19.2	Know About the Python Package Index . . . . .	226
19.3	Know About DjangoPackages.com . . . . .	227
19.4	Know Your Resources . . . . .	227
19.5	Tools for Installing and Managing Packages . . . . .	227
19.6	Package Requirements . . . . .	228
19.7	Wiring Up Django Packages: The Basics . . . . .	228
19.7.1	Step 1: Read the Documentation for the Package . . . . .	228
19.7.2	Step 2: Add Package and Version Number to Your Requirements . . . . .	228
19.7.3	Step 3: Install the Requirements Into Your Virtualenv . . . . .	229
19.7.4	Step 4: Follow the Package's Installation Instructions Exactly . . . . .	230
19.8	Troubleshooting Third-Party Packages . . . . .	230
19.9	Releasing Your Own Django Packages . . . . .	230
19.10	What Makes a Good Package? . . . . .	231
19.10.1	Purpose . . . . .	231
19.10.2	Scope . . . . .	232
19.10.3	Documentation . . . . .	232
19.10.4	Tests . . . . .	232
19.10.5	Activity . . . . .	232
19.10.6	Community . . . . .	233
19.10.7	Modularity . . . . .	233
19.10.8	Availability on PyPI . . . . .	233
19.10.9	Uses the Broadest Requirements Specifiers Possible . . . . .	233
19.10.10	Proper Version Numbers . . . . .	235
19.10.11	Name . . . . .	236
19.10.12	License . . . . .	236
19.10.13	Clarity of Code . . . . .	237
19.10.14	Use URL Namespaces . . . . .	237

---

19.11	Creating Your Own Packages the Easy Way . . . . .	237
19.12	Maintaining Your Open-Source Package . . . . .	238
19.12.1	Give Credit for Pull Requests . . . . .	239
19.12.2	Handling Bad Pull Requests . . . . .	239
19.12.3	Do Formal PyPI Releases . . . . .	240
19.12.4	Create and Deploy Wheels to PyPI . . . . .	241
19.12.5	Upgrade the Package to New Versions of Django . . . . .	242
19.12.6	Follow Good Security Practices . . . . .	242
19.12.7	Provide Sample Base Templates . . . . .	243
19.12.8	Give the Package Away, . . . . .	243
19.13	Additional Reading . . . . .	243
19.14	Summary . . . . .	244
<b>20</b>	<b>Testing Stinks and Is a Waste of Money!</b>	<b>245</b>
20.1	Testing Saves Money, Jobs, and Lives . . . . .	245
20.2	How to Structure Tests . . . . .	246
20.3	How to Write Unit Tests . . . . .	247
20.3.1	Each Test Method Tests One Thing . . . . .	247
20.3.2	For Views, When Possible Use the Request Factory . . . . .	250
20.3.3	Don't Write Tests That Have to Be Tested . . . . .	250
20.3.4	Don't Repeat Yourself Doesn't Apply to Writing Tests . . . . .	250
20.3.5	Don't Rely on Fixtures . . . . .	251
20.3.6	Things That Should Be Tested . . . . .	251
20.3.7	Document the Purpose of Each Test . . . . .	253
20.4	Continuous Integration . . . . .	253
20.5	Who Cares? We Don't Have Time for Tests! . . . . .	253
20.6	The Game of Test Coverage . . . . .	254
20.7	Setting Up the Test Coverage Game . . . . .	254
20.7.1	Step 1: Start Writing Tests . . . . .	254
20.7.2	Step 2: Run Tests and Generate Coverage Report . . . . .	254
20.7.3	Step 3: Generate the report! . . . . .	255
20.8	Playing the Game of Test Coverage . . . . .	256
20.9	Summary . . . . .	256
<b>21</b>	<b>Documentation: Be Obsessed</b>	<b>257</b>
21.1	Use reStructuredText for Python Docs . . . . .	257

---

21.2	Use Sphinx to Generate Documentation From reStructuredText . . . . .	259
21.3	What Docs Should Django Projects Contain? . . . . .	260
21.4	Additional Documentation Resources . . . . .	261
21.5	The Markdown Alternative . . . . .	261
21.6	Wikis and Other Documentation Methods . . . . .	262
21.7	Summary . . . . .	262
<b>22</b>	<b>Finding and Reducing Bottlenecks</b> . . . . .	<b>263</b>
22.1	Should You Even Care? . . . . .	263
22.2	Speed Up Query-Heavy Pages . . . . .	263
22.2.1	Find Excessive Queries With Django Debug Toolbar . . . . .	263
22.2.2	Reduce the Number of Queries . . . . .	264
22.2.3	Speed Up Common Queries . . . . .	265
22.2.4	Switch ATOMIC_REQUESTS to False . . . . .	266
22.3	Get the Most Out of Your Database . . . . .	266
22.3.1	Know What Doesn't Belong in the Database . . . . .	267
22.3.2	Getting the Most Out of PostgreSQL . . . . .	267
22.3.3	Getting the Most Out of MySQL . . . . .	268
22.4	Cache Queries With Memcached or Redis . . . . .	268
22.5	Identify Specific Places to Cache . . . . .	268
22.6	Consider Third-Party Caching Packages . . . . .	269
22.7	Compression and Minification of HTML, CSS, and JavaScript . . . . .	269
22.8	Use Upstream Caching or a Content Delivery Network . . . . .	270
22.9	Other Resources . . . . .	270
22.10	Summary . . . . .	271
<b>23</b>	<b>Security Best Practices</b> . . . . .	<b>273</b>
23.1	Harden Your Servers . . . . .	273
23.2	Know Django's Security Features . . . . .	273
23.3	Turn Off DEBUG Mode in Production . . . . .	274
23.4	Keep Your Secret Keys Secret . . . . .	274
23.5	HTTPS Everywhere . . . . .	274
23.5.1	Use Secure Cookies . . . . .	276
23.5.2	Use HTTP Strict Transport Security (HSTS) . . . . .	276
23.6	Use Django 1.6's Allowed Hosts Validation . . . . .	277
23.7	Always Use CSRF Protection With HTTP Forms That Modify Data . . . . .	277

---

23.7.1	Posting Data via AJAX . . . . .	278
23.8	Prevent Against Cross-Site Scripting (XSS) Attacks . . . . .	279
23.8.1	Use Django Templates Over <code>mark_safe</code> . . . . .	279
23.8.2	Don't Allow Users to Set Individual HTML Tag Attributes . . . . .	279
23.8.3	Use JSON Encoding for Data Consumed by JavaScript . . . . .	279
23.8.4	Additional Reading . . . . .	279
23.9	Defend Against Python Code Injection Attacks . . . . .	280
23.9.1	Python Built-ins That Execute Code . . . . .	280
23.9.2	Python Standard Library Modules That Can Execute Code . . . . .	280
23.9.3	Third-Party Libraries That Can Execute Code . . . . .	281
23.9.4	Be Careful with Cookie-Based Sessions . . . . .	281
23.10	Validate All Incoming Data With Django Forms . . . . .	282
23.11	Disable the Autocomplete On Payment Fields . . . . .	285
23.12	Handle User-Uploaded Files Carefully . . . . .	285
23.12.1	When a CDN Is Not An Option . . . . .	285
23.12.2	Django and User-Uploaded Files . . . . .	286
23.13	Don't Use <code>ModelForms.Meta.exclude</code> . . . . .	287
23.13.1	Mass Assignment Vulnerabilities . . . . .	289
23.14	Don't Use <code>ModelForms.Meta.fields = "__all__"</code> . . . . .	289
23.15	Beware of SQL Injection Attacks . . . . .	290
23.16	Never Store Credit Card Data . . . . .	290
23.17	Secure the Django Admin . . . . .	291
23.17.1	Change the Default Admin URL . . . . .	291
23.17.2	Use <code>django-admin-honeypot</code> . . . . .	291
23.17.3	Only Allow Admin Access via HTTPS . . . . .	291
23.17.4	Limit Admin Access Based on IP . . . . .	292
23.17.5	Use the <code>allow_tags</code> Attribute With Caution . . . . .	292
23.18	Secure the Admin Docs . . . . .	292
23.19	Monitor Your Sites . . . . .	293
23.20	Keep Your Dependencies Up-to-Date . . . . .	293
23.21	Prevent Clickjacking . . . . .	293
23.22	Guard against XML Bombing With <code>defusedxml</code> . . . . .	293
23.23	Give Your Site a Security Checkup . . . . .	294
23.24	Put Up a Vulnerability Reporting Page . . . . .	294
23.25	Have a Plan Ready For When Things Go Wrong . . . . .	294

---

23.25.1	Shut Everything Down or Put It in Read-Only Mode . . . . .	295
23.25.2	Put Up a Static HTML Page . . . . .	295
23.25.3	Back Everything Up . . . . .	296
23.25.4	Email security@djangoproject.com, Even if It's Your Fault . . . . .	296
23.25.5	Start Looking Into the Problem . . . . .	297
23.26	Keep Up-to-Date on General Security Practices . . . . .	297
23.27	Summary . . . . .	298
<b>24</b>	<b>Logging: What's It For, Anyway?</b>	<b>299</b>
24.1	Application Logs vs. Other Logs . . . . .	299
24.2	Why Bother With Logging? . . . . .	300
24.3	When to Use Each Log Level . . . . .	300
24.3.1	Log Catastrophes With CRITICAL . . . . .	301
24.3.2	Log Production Errors With ERROR . . . . .	301
24.3.3	Log Lower-Priority Problems With WARNING . . . . .	302
24.3.4	Log Useful State Information With INFO . . . . .	303
24.3.5	Log Debug-Related Messages to DEBUG . . . . .	303
24.4	Log Tracebacks When Catching Exceptions . . . . .	305
24.5	One Logger Per Module That Uses Logging . . . . .	306
24.6	Log Locally to Rotating Files . . . . .	306
24.7	Other Logging Tips . . . . .	307
24.8	Necessary Reading Material . . . . .	308
24.9	Useful Third-Party Tools . . . . .	308
24.10	Summary . . . . .	308
<b>25</b>	<b>Signals: Use Cases and Avoidance Techniques</b>	<b>309</b>
25.1	When to Use and Avoid Signals . . . . .	309
25.2	Signal Avoidance Techniques . . . . .	310
25.2.1	Using Custom Model Manager Methods Instead of Signals . . . . .	310
25.2.2	Validate Your Model Elsewhere . . . . .	313
25.2.3	Override Your Model's Save or Delete Method Instead . . . . .	313
25.2.4	Use a Helper Function Instead of Signals . . . . .	313
25.3	Summary . . . . .	314
<b>26</b>	<b>What About Those Random Utilities?</b>	<b>315</b>
26.1	Create a Core App for Your Utilities . . . . .	315

---

26.2	Django's Own Swiss Army Knife . . . . .	316
26.2.1	django.contrib.humanize . . . . .	317
26.2.2	django.utils.encoding.force_text(value) . . . . .	317
26.2.3	django.utils.functional.cached_property . . . . .	317
26.2.4	django.utils.html.format_html(format_str, *args, **kwargs) . . . . .	318
26.2.5	django.utils.html.remove_tags(value, tags) . . . . .	318
26.2.6	django.utils.html.strip_tags(value) . . . . .	318
26.2.7	django.utils.six . . . . .	318
26.2.8	django.utils.text.slugify(value) . . . . .	319
26.2.9	django.utils.timezone . . . . .	320
26.2.10	django.utils.translation . . . . .	320
26.3	Exceptions . . . . .	321
26.3.1	django.core.exceptions.ImproperlyConfigured . . . . .	321
26.3.2	django.core.exceptions.ObjectDoesNotExist . . . . .	321
26.3.3	django.core.exceptions.PermissionDenied . . . . .	322
26.4	Serializers and Deserializers . . . . .	323
26.4.1	django.core.serializers.json.DjangoJSONEncoder . . . . .	326
26.4.2	django.core.serializers.pyyaml . . . . .	326
26.4.3	django.core.serializers.xml_serializer . . . . .	326
26.5	Summary . . . . .	327
<b>27</b>	<b>Deployment: Platforms as a Service</b> . . . . .	<b>329</b>
27.1	Evaluating a PaaS . . . . .	330
27.1.1	Compliance . . . . .	330
27.1.2	Pricing . . . . .	331
27.1.3	Uptime . . . . .	331
27.1.4	Staffing . . . . .	332
27.1.5	Scaling . . . . .	333
27.1.6	Documentation . . . . .	333
27.1.7	Performance Degradation . . . . .	333
27.1.8	Geography . . . . .	334
27.1.9	Company Stability . . . . .	334
27.2	Best Practices for Deploying to PaaS . . . . .	335
27.2.1	Aim For Identical Environments . . . . .	335
27.2.2	Automate All the Things! . . . . .	335
27.2.3	Maintain a Staging Instance . . . . .	336

---

27.2.4	Prepare for Disaster With Backups And Rollbacks . . . . .	336
27.2.5	Keep External Backups . . . . .	336
27.3	Summary . . . . .	336
<b>28</b>	<b>Deploying Django Projects</b>	<b>339</b>
28.1	Single-Server For Small Projects . . . . .	339
28.2	Multi-Server For Medium to Large Projects . . . . .	340
28.2.1	Basic Multi-Server Setup . . . . .	340
28.2.2	Advanced Multi-Server Setup . . . . .	341
28.3	WSGI Application Servers . . . . .	343
28.3.1	uWSGI and Django . . . . .	345
28.3.2	Apache and Environment Variables . . . . .	345
28.3.3	Apache and Virtualenv . . . . .	346
28.4	Automated, Repeatable Deployments . . . . .	346
28.4.1	Infrastructure Automation Tools . . . . .	347
28.4.2	SaltStack . . . . .	350
28.4.3	Ansible . . . . .	352
28.5	Summary . . . . .	352
<b>29</b>	<b>Identical Environments: The Holy Grail</b>	<b>353</b>
29.1	The Present: Vagrant . . . . .	354
29.1.1	Advantages of Vagrant . . . . .	354
29.1.2	Disadvantages of Vagrant . . . . .	355
29.1.3	Vagrant Resources . . . . .	355
29.2	Experimental: Docker . . . . .	355
29.2.1	Advantages of Docker . . . . .	355
29.2.2	Warning: Docker Is Under Heavy Development . . . . .	356
29.2.3	Docker Resources . . . . .	356
29.3	Summary . . . . .	356
<b>30</b>	<b>Continuous Integration</b>	<b>357</b>
30.1	Principles of Continuous Integration . . . . .	358
30.1.1	Write Lots of Tests! . . . . .	358
30.1.2	Keeping the Build Fast . . . . .	358
30.2	Tools for Continuously Integrating your Project . . . . .	359
30.2.1	Tox . . . . .	359



30.2.2	Jenkins . . . . .	360
30.3	Continuous Integration as a Service . . . . .	360
30.4	Additional Resources . . . . .	360
30.5	Summary . . . . .	361
<b>31</b>	<b>Where and How to Ask Django Questions</b>	<b>363</b>
31.1	What to Do When You're Stuck . . . . .	363
31.2	How to Ask Great Django Questions in IRC . . . . .	363
31.3	Insider Tip: Be Active in the Community . . . . .	364
31.3.1	10 Easy Ways to Participate . . . . .	364
31.4	Summary . . . . .	366
<b>32</b>	<b>Closing Thoughts</b>	<b>367</b>
<b>Appendix A: Packages Mentioned In This Book</b>		<b>369</b>
<b>Appendix B: Troubleshooting</b>		<b>375</b>
	Identifying the Issue . . . . .	375
	Our Recommended Solutions . . . . .	375
	Check Your Virtualenv Installation . . . . .	376
	Check If Your Virtualenv Has Django 1.6 Installed . . . . .	377
	Check For Other Problems . . . . .	377
<b>Appendix C: Additional Resources</b>		<b>379</b>
	Beginner Python Material . . . . .	379
	Beginner Django Material . . . . .	379
	More Advanced Django Material . . . . .	380
	Useful Python Material . . . . .	381
	JavaScript Resources . . . . .	382
<b>Appendix D: Internationalization and Localization</b>		<b>383</b>
	Start Early . . . . .	383
	Wrap Content Strings with Translation Functions . . . . .	384
	Don't Interpolate Words in Sentences . . . . .	385
	Browser Page Layout . . . . .	388
<b>Appendix E: Settings Alternatives</b>		<b>391</b>
	Twelve Factor-Style Settings . . . . .	391

Multiple Settings with Configuration Objects . . . . .	392
<b>Appendix F: Working with Python 3</b>	<b>395</b>
Most Critical Packages Work with Python 3 . . . . .	395
Use Python 3.3.3 or Later . . . . .	397
Working With Python 2 and 3 . . . . .	397
Resources . . . . .	398
<b>Acknowledgments</b>	<b>399</b>
<b>Index</b>	<b>403</b>