

УДК 373.167.1:004.42+004.42(075.3)
ББК 32.973.721
П27

П27 Персиваль Г.

Python. Разработка на основе тестирования. / пер. с англ. Логунов А. В. – М.: ДМК Пресс, 2018. – 622 с.: ил.

ISBN 978-5-97060-594-3

Книга демонстрирует преимущества методологии разработки на основе тестирования (TDD) на языке Python. Вы научитесь писать и выполнять тесты для создания любого фрагмента вашего приложения и затем разрабатывать минимальный объем программного кода, необходимого для прохождения этих тестов. Вы также научитесь работать с различными инструментами и фреймворками, такими как Django, Selenium, Git, jQuery и Mock.

Издание предназначено всем разработчикам, кто уже освоил начальный уровень программирования на Python и хочет перейти на следующий.

УДК 373.167.1:004.42+004.42(075.3)
ББК 32.973.721

Original English language edition published by O'Reilly Media, Inc. Copyright © 2017 Harry Percival. All rights reserved. Russian-language edition copyright © 2017 by ДМК Пресс. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-49195-870-4 (англ.)
ISBN 978-5-97060-594-3 (рус.)

© 2017 Harry Percival. All rights reserved.
© Оформление, перевод на русский язык, издание,
ДМК Пресс, 2018

Оглавление

Предисловие.....	16
Предпосылки и предположения	21
Сопутствующее видео.....	30
Признательности.....	31
Часть I. Основы TDD и Django	33
Глава 1. Настройка Django с использованием функционального теста	34
Повинуйтесь Билли-тестировщику! Ничего не делайте, пока у вас не будет теста	34
Приведение Django в рабочее состояние	37
Запуск репозитория Git	40
Глава 2. Расширение функционального теста при помощи модуля unittest.....	44
Использование функционального теста для определения минимального дееспособного приложения	44
Модуль unittest стандартной библиотеки Python	47
Фиксация	50
Глава 3. Тестирование простой домашней страницы при помощи модульных тестов	52
Первое приложение Django и первый модульный тест.....	53
Модульные тесты и их отличия от функциональных тестов	53
Модульное тестирование в Django.....	55
MVC в Django, URL-адреса и функции представления.....	56
Наконец-то! Мы на самом деле напишем прикладной код!	58
Файл urls.py.....	59
Модульное тестирование представления	62
Цикл «модульный-тест/программный-код».....	64
Глава 4. И что же делать со всеми этими тестами (и рефакторизацией)?	68
Программировать – все равно что поднимать ведро с водой из колодца	69
Использование Selenium для тестирования взаимодействий пользователя	71
Правило «Не тестировать константы» и шаблоны во спасение	74
Перестройка программного кода для использования шаблона	75
Тестовый клиент Django	79
О рефакторизации.....	81
Еще немного о главной странице	83
Резюме: процесс TDD	85
Глава 5. Сохранение вводимых пользователем данных: тестирование базы данных	88

Подключение формы для отправки POST-запроса	89
Обработка POST-запроса на сервере	92
Передача переменных Python для вывода в шаблоне	93
Если клюнуло трижды, рефакторизуй	98
Django ORM и первая модель	100
Первая миграция базы данных	102
Тест продвинулся удивительно далеко	103
Новое поле означает новую миграцию	104
Сохранение POST-запроса в базу данных	105
Переадресация после POST-запроса	108
Лучшие приемы модульного тестирования: каждый тест должен проверять одну единицу кода	109
Генерирование элементов в шаблоне	110
Создание производственной базы данных при помощи команды migrate	112
Резюме	115

Глава 6. Усовершенствование функциональных тестов: обеспечение

изоляция и удаление методов sleep	118
Обеспечение изоляции в функциональных тестах	118
Выполнение только модульных тестов	122
Ремарка: обновление Selenium и Geckodriver	123
О неявных и явных ожиданиях и методе time.sleep	124

Глава 7. Работа в инкрементном режиме

Маломасштабные конструктивные изменения по мере необходимости	130
В первую очередь маломасштабные конструктивные изменения	131
Вам это никогда не понадобится!	131
REST (-овский) подход	132
Инкрементная реализация новой структуры кода на основе TDD	133
Обеспечение теста на наличие регрессии	134
Итеративное движение в сторону новой структуры кода	137
Первый самодостаточный шаг: один новый URL-адрес	139
Новый URL-адрес	140
Новая функция представления	140
Зеленый? Рефакторизуй!	142
Еще один шаг: отдельный шаблон для просмотра списков	143
Третий шаг: URL-адрес для добавления элементов списка	146
Тестовый класс для создания нового списка	146
URL-адрес и представление для создания нового списка	148
Удаление теперь уже избыточного кода и тестов	149
Регрессия! Наведение форм на новый URL-адрес	149
Стиснув зубы: корректировка моделей	151
Связь по внешнему ключу	153
Адаптация остальной части мира к новым моделям	155
Каждому списку – свой URL-адрес	157
Извлечение параметров из URL-адресов	158
Адаптирование new_list к новому миру	160

Функциональные тесты обнаруживают еще одну регрессию	161
Еще одно представление для добавления элементов в существующий список.....	162
Остерегайтесь «жадных» регулярных выражений!.....	163
Последний новый URL-адрес	164
Последнее новое представление	165
Тестирование контекстных объектов отклика напрямую.....	166
Финальная рефакторизация с использованием URL-включений.....	168

Часть II. Непременные условия веб-разработки..... 171

Глава 8. Придание привлекательного вида: макет, стилевое оформление сайта и что тут тестировать172

Что функционально тестируется в макете и стилевом оформлении.....	172
Придание привлекательного вида: использование платформы CSS.....	176
Наследование шаблонов в Django	178
Интеграция платформы Bootstrap	180
Строки и столбцы	180
Статические файлы в Django.....	182
Переход на StaticLiveServerTestCase	183
Использование компонентов Bootstrap для улучшения внешнего вида сайта	184
Класс jumbotron.....	184
Большие поля ввода	185
Стилистическое оформление таблицы	185
Использование собственного CSS	185
О чем мы умолчали: collectstatic и другие статические каталоги.....	187
Несколько вещей, которые не удалось	190

Глава 9. Тестирование развертывания с использованием промежуточного сайта.....191

TDD и опасные зоны развертывания	192
Как всегда, начинайте с теста	194
Получение доменного имени	196
Ручное обеспечение работы сервера для размещения сайта.....	196
Выбор места размещения сайта	197
Запуск сервера.....	197
Учетные записи пользователей, SSH и полномочия.....	198
Инсталляция Nginx	198
Инсталляция Python 3.6.....	200
Конфигурирование доменов для промежуточного и реального серверов	200
Использование ФТ для подтверждения, что домен работает и Nginx выполняется	201
Развертывание исходного кода вручную.....	201
Корректировка расположения базы данных.....	202
Создание Virtualenv вручную и использование requirements.txt.....	204
Простое конфигурирование Nginx	206
Создание базы данных при помощи команды migrate.....	209
Победа! Наше хакерское развертывание работает	210

Глава 10. Переход к развертыванию, готовому к эксплуатации.....	212
Переход на Gunicorn.....	212
Настройка Nginx для раздачи статических файлов	214
Переход на использование сокетов Unix.....	215
Присвоение DEBUG значения False и настройка ALLOWED_HOSTS	216
Применение Systemd для проверки, что Gunicorn запускается на начальной загрузке	217
Сохранение изменений: добавление Gunicorn в файл requirements.txt.....	218
Размышления об автоматизации.....	219
Сохранение шаблонов конфигурационных файлов этапа обеспечения работы	219
Сохранение хода выполнения.....	222
Глава 11. Автоматизация развертывания с помощью Fabric	224
Описание частей сценария Fabric для развертывания	225
Создание структуры каталогов	226
Получение исходного кода из репозитория командой git.....	226
Обновление файла settings.py.....	227
Обновление virtualenv.....	229
Миграция базы данных при необходимости.....	229
Испытание автоматизации.....	230
Развертывание на работающем сайте	231
Конфигурирование Nginx и Gunicorn при помощи sed	234
Маркировка релиза командой git tag	235
Дополнительные материалы для чтения.....	236
Глава 12. Разделение тестов на многочисленные файлы и обобщенный помощник ожидания.....	238
Начало с ФТ валидации данных: предотвращение пустых элементов	238
Пропуск теста.....	239
Разбиение функциональных тестов на несколько файлов.....	241
Выполнение только одного файла с тестами	244
Новый инструмент функционального тестирования: обобщенная вспомогательная функция явного ожидания.....	245
Завершение ФТ.....	249
Рефакторизация модульных тестов на несколько файлов	251
Глава 13. Валидация на уровне базы данных.....	254
Валидация на уровне модели.....	255
Контекстный менеджер self.assertRaises	255
Причуда Django: сохранение модели не выполняет валидацию.....	256
Выведение на поверхность ошибок валидации модели в представлении	257
Проверка, чтобы недопустимые входные данные не сохранялись в базе данных.....	261
Схема Django: обработка POST-запросов в том же представлении, которое генерирует форму	263
Рефакторизация: передача функциональности new_item в view_list	264
Обеспечение валидации модели в view_list	267
Рефакторизация: удаление жестко закодированных URL-адресов.....	269

Ter {% url %} шаблона.....	269
Использование get_absolute_url для переадресаций.....	270
Глава 14. Простая форма	274
Перемещение программной логики валидации из формы	274
Исследование API форм при помощи модульного теста.....	275
Переход на Django ModelForm.....	277
Тестирование и индивидуальная настройка валидации формы.....	278
Использование формы в представлениях.....	281
Использование формы в представлении с GET-запросом.....	281
Глобальный поиск и замена.....	282
Использование формы в представлении, принимающем POST-запросы	285
Адаптация модульных тестов к представлению new_list	285
Использование формы в представлении.....	287
Использование формы для отображения ошибок в шаблоне	287
Использование формы в другом представлении	288
Вспомогательный метод для нескольких коротких тестов	289
Неожиданное преимущество: бесплатная валидация на стороне клиента из HTML5.....	291
Одобряющее похлопывание по спине.....	293
Не потратили ли мы уйму времени впустую?	294
Использование собственного для формы метода save	295
Глава 15. Более развитые формы	298
Еще один ФТ на наличие повторяющихся элементов.....	298
Предотвращение дубликатов на уровне модели	299
Небольшое отступление по поводу упорядочивания Queryset и представлений строковых значений.....	301
Новое написание старого теста модели	304
Некоторые ошибки целостности проявляются при сохранении	306
Экспериментирование с проверкой на наличие повторяющихся элементов на уровне представлений.....	307
Более сложная форма для проверки на наличие повторяющихся значений	308
Использование существующей формы для элемента списка в представлении для списка	310
Итоги: что мы узнали о тестировании Django.....	313
Глава 16 Пробуем окунуться, очень робко, в JavaScript.....	316
Начинаем с ФТ	316
Настройка элементарного исполнителя тестов на JavaScript.....	318
Использование элемента div для jQuery и фикстуры	320
Создание модульного теста на JavaScript для требуемой функциональности	324
Фикстуры, порядок выполнения и глобальное состояние: ключевые проблемы тестирования на JS.....	326
console.log для отладочной распечатки.....	326
Использование функции инициализации для большего контроля над временем выполнения.....	328
Коломбо говорит: стереотипный код для onload и организация пространства	

имен.....	330
Тестирование на Javascript в цикле TDD.....	332
Несколько вещей, которые не удалось сделать.....	332
Глава 17. Развертывание нового программного кода	334
Развертывание на промежуточном сервере	334
Развертывание на реальном сервере	335
Что делать, если вы видите ошибку базы данных.....	335
Итоги: маркировка нового релиза командой git tag	335
Часть III. Основы TDD и Django.....	337
Глава 18. Аутентификация пользователя, импульсное исследование и внедрение его результатов.....	338
Беспарольная аутентификация	338
Разведочное программирование, или Импульсное исследование	339
Открытие ветки для результатов импульсного исследования	340
Авторизация на стороне клиента в пользовательском интерфейсе.....	341
Отправка электронных писем из Django	341
Использование переменных окружения для предотвращения секретов в исходном коде	344
Хранение маркеров в базе данных.....	344
Индивидуализированные модели аутентификации.....	345
Завершение индивидуализированной авторизации в Django	346
Внедрение результатов импульсного исследования.....	351
Возвращение импульсного исходного кода в прежний вид.....	353
Минимальная индивидуализированная модель пользователя	354
Тесты в качестве документирования	357
Модель маркера для соединения электронных адресов с уникальным идентификатором	358
Глава 19. Использование имитаций для тестирования внешних зависимостей или сокращения дублирования	362
Перед началом: получение базовой инфраструктуры	362
Создание имитаций вручную, или Обезьянья заплатка	363
Библиотека Mock	367
Использование unittest.patch.....	368
Продвижение ФТ чуть дальше вперед	371
Тестирование инфраструктуры сообщений Django	371
Добавление сообщений в HTML.....	374
Начало с URL-адреса для входа в систему.....	375
Подтверждение отправки пользователю ссылки с маркером	376
Создание индивидуализированного серверного процессора аутентификации на основе результатов импульсного исследования.....	378
Еще один тест для каждого случая	379
Метод get_user	382

Использование серверного процессора аутентификации в представлении входа в систему.....	384
Еще одна причина использовать имитации: устранение повторов.....	385
Использование mock.return_value.....	388
Установка заплатки на уровне класса.....	390
Момент истины: пройдет ли ФТ?.....	392
Теоретически – работает! Работает ли на практике?.....	394
Завершение ФТ, тестирование выхода из системы.....	396
Глава 20. Тестовые фикстуры и декоратор для явных ожиданий.....	399
Пропуск регистрации в системе путем предварительного создания сеанса.....	399
Проверка работоспособности решения.....	402
Финальный вспомогательный метод явного ожидания: декоратор ожидания.....	405
Глава 21. Отладка на стороне сервера.....	410
Чтобы убедиться в пудинге, надо его попробовать: использование предварительного сервера для отлавливания финальных дефектов.....	410
Настройка журналирования.....	411
Установка секретных переменных окружения на сервере.....	413
Адаптация ФТ для тестирования реальных электронных писем POP3.....	414
Управление тестовой базой данных на промежуточном сервере.....	418
Управляющая команда Django для создания сеансов.....	418
Настройка ФТ для выполнения управляющей команды на сервере.....	420
Использование fabric напрямую из Python.....	421
Резюме: создание сеансов локально по сравнению с промежуточным сервером.....	422
Внедрение программного кода журналирования.....	424
Итоги.....	425
Глава 22. Завершение приложения «Мои списки»: TDD с подходом «снаружи внутрь».....	427
Альтернатива: «изнутри наружу».....	427
Почему «снаружи внутрь» предпочтительнее?.....	428
ФТ для «Моих списков».....	428
Внешний уровень: презентация и шаблоны.....	431
Спуск на один уровень вниз к функциям представления (к контроллеру).....	431
Еще один проход снаружи внутрь.....	433
Быстрая реструктуризация иерархии наследования шаблонов.....	433
Конструирование API при помощи шаблона.....	434
Спуск к следующему уровню: что именно представление передает в шаблон.....	436
Следующее техническое требование из уровня представлений: новые списки должны записывать владельца.....	437
Момент принятия решения: перейти к следующему уровню с неработающим тестом или нет.....	438
Спуск к уровню модели.....	439
Финальный шаг: подача .name API из шаблона.....	441
Глава 23. Изоляция тестов и «слушание своих тестов».....	444

Пересмотр точки принятия решения: уровень представлений зависит от ненаписанного кода моделей	444
Первая попытка использования имитаций для изоляции	446
Использование имитации side_effects для проверки последовательности событий	447
Слушайте свои тесты: уродливые тесты сигнализируют о необходимости рефакторизации	449
Написание тестов по-новому для представления, которое будет полностью изолировано	450
Держите рядом старый комплект интегрированных тестов в качестве проверки на токсичность	450
Новый комплект тестов с полной изоляцией	451
Мышление с точки зрения взаимодействующих объектов	452
Спуск вниз на уровень форм	457
Продолжайте слушать свои тесты: удаление программного кода ORM из нашего приложения	458
Наконец, спуск вниз на уровень моделей	462
Назад к представлениям	464
Момент истины (и риски имитации)	466
Точка зрения о взаимодействиях между уровнями как о контрактах	467
Идентификация неявных контрактов	468
Исправление недосмотра	469
Еще один тест	471
Наведение порядка: что следует убрать из комплекта интегрированных тестов	472
Удаление избыточного кода на уровне форм	472
Удаление старой реализации представления	473
Удаление избыточного кода на уровне форм	474
Выводы: когда писать изолированные тесты, а когда – интегрированные	475
Пусть вычислительная сложность будет вашим гидом	476
Следует ли делать оба типа тестов?	477
Вперед!	477
Глава 24. Непрерывная интеграция	479
Инсталляция сервера Jenkins	479
Конфигурирование сервера Jenkins	481
Первоначальная разблокировка	481
Набор плагинов на первое время	481
Конфигурирование пользователя-администратора	482
Добавление плагинов	483
Указание серверу Jenkins, где искать Python 3 и Xvfb	483
Завершение с HTTPS	484
Настройка проекта	484
Первая сборка!	485
Установка виртуального дисплея, чтобы ФТ выполнялись бездисплейно	487
Взятие снимков экрана	489
Если сомневаетесь – встряхните тайм-аут!	492
Выполнение тестов JavaScript QUnit на Jenkins вместе с PhantomJS	494
Установка node	494

Добавление шагов сборки в Jenkins	495
Больше возможностей с CI-сервером	497
Глава 25. Социально зачимый кусок, шаблон проектирования	
«Страница» и упражнение для читателя.....	499
ФТ с многочисленными пользователями и addCleanup	499
Страничный шаблон проектирования	501
Расширение ФТ до второго пользователя и страница «Мои списки».....	504
Упражнение для читателя.....	506
Глава 26. Быстрые тесты, медленные тесты и горячий поля	509
Тезис: модульные тесты сверхбыстры и к тому же хороши.....	511
Более быстрые тесты означают более быструю разработку.....	511
Священное состояние потока.....	511
Медленные тесты не выполняются часто, что приводит к плохому коду.....	512
Теперь у нас все в порядке, но интегрированные тесты со временем становятся медленнее	512
Не верьте мне	512
И модульные тесты управляют хорошей структурой кода.....	512
Проблемы с «чистыми» модульными тестами	512
Изолированные тесты труднее читать и писать	512
Изолированные тесты не тестируют интеграцию автоматически	513
Модульные тесты редко отлавливают неожиданные дефекты	513
Тесты с имитациями могут стать близко привязанными к реализации.....	513
Но все эти проблемы могут быть преодолены	514
Синтез: что мы хотим от всех наших тестов?.....	514
Правильность	514
Чистый код, удобный в сопровождении	514
Продуктивный поток операций.....	515
Оценивайте свои тесты относительно преимуществ, которые вы хотите от них получить	515
Архитектурные решения	516
Порты и адаптеры/шестиугольная/чистая архитектура	516
Функциональное ядро, императивная оболочка	517
Заключение	518
Дополнительные материалы для чтения.....	518
Повинуйтесь Билли-тестировщику!	521
Тестировать очень тяжело.....	521
Держите свои сборки CI-сервера на зеленом уровне.....	521
Гордитесь своими тестами так же, как своим программным кодом.....	522
Не забудьте дать на чай персоналу заведения.....	522
Не пропадайте!	522
Приложение A. PythonAnywhere	523
Выполнение сеансов Firefox Selenium при помощи Xvfb.....	523
Настройка Django как веб-приложение PythonAnywhere.....	525
Очистка папки /tmp.....	526

Снимки экрана	526
Глава о развертывании	526
Приложение В. Представления на основе классов в Django.....	528
Обобщенные представления на основе классов	528
Домашняя страница как FormView.....	529
Использование form_valid для индивидуализации CreateView	530
Более сложное представление для обработки просмотра и добавления к списку	533
Сравните старую верию с новой.....	536
Лучшие приемы модульного тестирования обобщенных представлений на основе классов.....	537
Приложение С. Обеспечение работы серверной среды при помощи Ansible.....	539
Установка системных пакетов и Nginx.....	539
Конфигурирование Gunicorn и использование обработчиков для перезапуска служб	541
Что делать дальше	542
Приложение D. Тестирование миграций базы данных.....	544
Попытка развертывания на промежуточном сервере.....	544
Выполнение тестовой миграции локально.....	545
Вставка миграции данных.....	546
Совместное тестирование новых миграций	547
Выводы	548
Приложение Е. Разработка на основе поведения (BDD).....	550
Что такое BDD?.....	550
Базовые служебные операции	551
Написание ФТ как компонента при помощи синтаксиса языка Gherkin.....	552
Программирование шаговых функций	553
Определение первого шага	555
Эквиваленты setUp и tearDown в environment.py.....	555
Еще один прогон.....	556
Извлечение параметров в шагах	556
BDD по сравнению с ФТ с локальными комментариями	559
BDD способствует написанию структурированного тестового кода.....	561
Страничный шаблон проектирования как альтернатива	561
BDD может быть менее выразительным, чем локальные комментарии	563
Будут ли непрограммисты писать тесты?	563
Некоторые предварительные выводы	564
Приложение F. Создание REST API: JSON, Ajax и имитирование на JavaScript.....	565
Наш подход для этого раздела	565
Выбор подхода к тестированию	566
Организация базовой конвейерной передачи.....	566
Получение фактического отклика.....	568

Добавление POST-метода.....	569
Тестирование клиентского Ajax при помощи Sinon.js.....	570
Соединение всего в шаблоне, чтобы убедиться, что это реально работает.....	574
Реализация Ajax POST-запроса, включая маркер CSRF.....	576
Имитация в JavaScript.....	578
Валидация данных. Упражнение для читателя.....	582
Приложение G. Django-Rest-Framework.....	587
Инсталляция.....	587
Сериализаторы (на самом деле – объекты ModelSerializer).....	588
Объекты Viewset (на самом деле – объекты ModelViewSet) и объекты Router.....	589
Другой URL для элемента с POST-запросом.....	592
Адаптирование на стороне клиента.....	593
Что дает инфраструктура Django-Rest-Framework.....	595
Приложение H. Шпаргалка.....	599
Начальная настройка проекта.....	599
Основной поток операций TDD.....	599
Выход за пределы тестирования только на сервере разработки.....	600
Общие приемы тестирования.....	601
Лучшие приемы на основе Selenium / функциональных тестов.....	601
«Снаружи внутрь», изоляция тестов против интегрированных тестов и имитация.....	602
Приложение I. Что делать дальше.....	603
Уведомления – на сайте и по электронной почте.....	603
Переходите на Postgres.....	604
Выполняйте тесты относительно разных браузеров.....	604
Тесты на коды состояния 404 и 500.....	604
Сайт администратора Django.....	604
Напишите несколько тестов защищенности.....	605
Тест на мягкую деградацию.....	605
Кэширование и тестирование и производительности.....	605
MVC-инфраструктуры для JavaScript.....	605
Async и протокол Websocket.....	606
Перейдите на использование py.test.....	606
Попробуйте coverage.py.....	606
Шифрование на стороне клиента.....	606
Здесь место для ваших предложений.....	607
Приложение J. Примеры исходного кода.....	608
Полный список ссылок для каждой главы.....	608
Использование Git для проверки вашего прогресса.....	610
Скачивание ZIP-файла для главы.....	611
Не позволяйте этому превращаться в костыль!.....	611
Предметный указатель.....	612