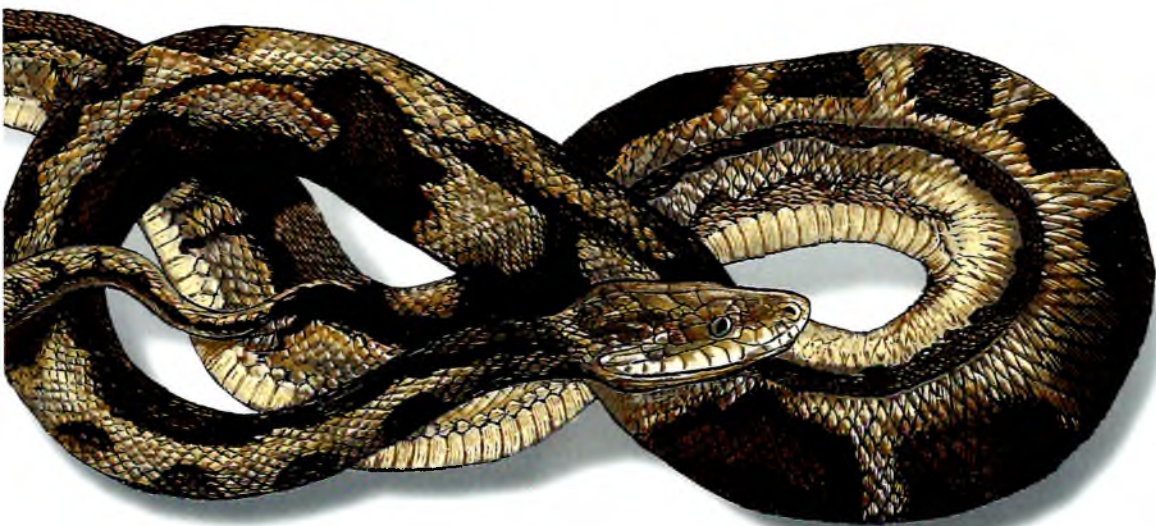


O'REILLY®

Паттерны разработки на Python

TDD, DDD и событийно-ориентированная
архитектура



Гарри Персиваль
Боб Грегори

Краткое содержание

Предисловие	12
Введение	21
Часть I. Создание архитектуры для поддержки моделирования предметной области	
Глава 1. Моделирование предметной области	32
Глава 2. Паттерн «Репозиторий»	53
Глава 3. О связанности и абстракциях	74
Глава 4. Первый вариант использования: API фреймворка Flask и сервисный слой	90
Глава 5. TDD на повышенной и пониженной передачах	109
Глава 6. Паттерн UoW	120
Глава 7. Агрегаты и границы согласованности	136
Часть II. Событийно-управляемая архитектура	
Глава 8. События и шина сообщений	162
Глава 9. Катимся в город на шине сообщений	180
Глава 10. Команды и обработчик команд	200
Глава 11. Событийно-управляемая архитектура: использование событий для интеграции микросервисов	211
Глава 12. Разделение обязанностей команд и запросов	226
Глава 13. Внедрение зависимостей (и начальная загрузка)	246
Эпилог	268
Приложение А. Сводная диаграмма и таблица	290
Приложение Б. Шаблонная структура проекта	292
Приложение В. Замена инфраструктуры: делаем все с помощью CSV	302
Приложение Г. Паттерны «Репозиторий» и UoW с Django	308
Приложение Д. Валидация	318
Об авторах	329
Об обложке	330

Оглавление

Предисловие	12
Управлять сложностью, решая бизнес-задачи	12
Почему Python?	13
TDD, DDD и событийно-управляемая архитектура	14
Для кого эта книга	15
Краткий обзор книги	16
Дополнительные материалы.....	17
Примеры кода и работа с ним.....	17
Условные обозначения	19
Благодарности.....	19
От издательства	20

Введение	21
Почему в проекте что-то идет не так?	21
Инкапсуляции и абстракции	22
Разделение на слои	24
Принцип инверсии зависимостей.....	25
Место для всей бизнес-логики: модель предметной области	27

ЧАСТЬ I. СОЗДАНИЕ АРХИТЕКТУРЫ ДЛЯ ПОДДЕРЖКИ МОДЕЛИРОВАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

Глава 1. Моделирование предметной области	32
Что такое модель предметной области	32
Изучение языка предметной области.....	36
Юнит-тестирование моделей предметных областей	38
Не все должно быть объектом: функция службы предметной области.....	48

Глава 2. Паттерн «Репозиторий»	53
Организация постоянного хранения модели предметной области.....	54
Немного псевдокода: что нам потребуется?	55
Применение принципа инверсии зависимостей для доступа к данным	55
Напоминание: наша модель.....	57
Введение паттерна «Репозиторий»	63
Теперь поддельный репозиторий для тестов создается просто!	69
Что такое порт и что такое адаптер в Python.....	70
Выводы	71
Глава 3. О связанности и абстракциях	74
Абстрагирование состояния способствует тестопригодности	76
Выбор правильной(-ых) абстракции(-й)	79
Реализация выбранных абстракций	81
Выводы	89
Глава 4. Первый вариант использования: API фреймворка Flask и сервисный слой	90
Связываем приложение с реальным миром.....	92
Первый сквозной тест.....	93
Простая реализация	94
Состояния ошибок, требующие проверки базы данных	95
Введение сервисного слоя и использование поддельного репозитория для юнит-теста.....	97
Почему все называется службой?	102
Складываем все в папки, чтобы понять, где что находится	103
Выводы	105
Глава 5. TDD на повышенной и пониженной передачах	109
Как выглядит пирамида тестирования	110
Должны ли тесты слоя предметной области перейти в сервисный слой? ...	110
Какие тесты писать	112
Повышенная и пониженная передачи.....	113
Устранение связей между тестами сервисного слоя и предметной областью.....	113

Дальнейшее улучшение с помощью сквозных тестов	117
Выводы	118
Глава 6. Паттерн UoW	120
Паттерн UoW работает с репозиторием	121
Тестирование UoW интеграционными тестами.....	123
UoW и его контекстный менеджер	124
Использование паттерна UoW в сервисном слое.....	127
Явные тесты для форм поведения по фиксации/откату	128
Явные и неявные фиксации	129
Примеры: использование паттерна UoW для группировки многочисленных операций в атомарную единицу	131
Приведение в порядок интеграционных тестов.....	132
Выводы	133
Глава 7. Агрегаты и границы согласованности	136
Почему бы просто не записать все в электронную таблицу?.....	137
Инварианты, ограничения и согласованность	138
Что такое агрегат	139
Выбор агрегата.....	141
Один агрегат = один репозиторий.....	145
А что насчет производительности?	146
Оптимистичная конкурентность с номерами версий.....	148
Тестирование правил целостности данных	152
Выводы	155
Итоги части I.....	157

ЧАСТЬ II. СОБЫТИЙНО-УПРАВЛЯЕМАЯ АРХИТЕКТУРА

Глава 8. События и шина сообщений	162
Как избежать беспорядка	164
Принцип единственной обязанности	166
Катимся на шине сообщений!	167

Вариант 1: сервисный слой берет события из модели и помещает их в шину сообщений.....	171
Вариант 2: сервисный слой инициирует собственные события.....	172
Вариант 3: UoW публикует события в шине сообщений.....	173
Выводы	177
Глава 9. Катимся в город на шине сообщений.....	180
Новое требование приводит к новой архитектуре.....	182
Рефакторинг функций служб для обработчиков сообщений.....	184
Реализация нового требования.....	191
Тест-драйв нового обработчика.....	192
Необязательно: юнит-тест обработчиков событий в изоляции с помощью поддельной шины сообщений.....	196
Выводы	198
Глава 10. Команды и обработчик команд.....	200
Команды и события	200
Различия в обработке исключений.....	202
События, команды и обработка ошибок.....	204
Синхронное восстановление после ошибок.....	208
Выводы	210
Глава 11. Событийно-управляемая архитектура: использование событий для интеграции микросервисов	211
Распределенный комок грязи, или Мыслить существительными	212
Обработка ошибок в распределенных системах.....	216
Альтернатива: временное устранение связанности при помощи асинхронного обмена сообщениями	218
Использование канала «издатель/подписчик» хранилища Redis для интеграции	219
Тестирование с помощью сквозного теста	219
Внутренние события против внешних.....	224
Выводы	224

Глава 12. Разделение обязанностей команд и запросов.....	226
Модели предметной области для записи.....	226
Большинство пользователей не собираются покупать вашу мебель.....	228
PRG и разделение команд и запросов.....	230
Хватайте свой обед, ребята.....	232
Тестирование представлений CQRS.....	233
«Очевидная» альтернатива 1: использование существующего репозитория.....	234
Модель предметной области не оптимизирована для операций чтения.....	235
«Очевидная» альтернатива № 2: использование ORM.....	236
SELECT N+1 и другие соображения по поводу производительности.....	237
Время прыгать через акулу.....	238
Изменить реализацию модели чтения очень просто.....	242
Выводы.....	244
Глава 13. Внедрение зависимостей (и начальная загрузка).....	246
Неявные зависимости против явных.....	249
Разве явные зависимости не кажутся странными и Java-подобными?.....	250
Подготовка обработчиков: внедрение зависимостей вручную с помощью замыканий и частных применений.....	252
Альтернатива с использованием классов.....	254
Сценарий начальной загрузки.....	255
Шина сообщений получает обработчики во время выполнения.....	258
Использование начальной загрузки в точках входа.....	259
Внедрение зависимостей в тестах.....	260
«Правильное» создание адаптера: рабочий пример.....	262
Выводы.....	266
Эпилог.....	268
И что теперь?.....	268
Как мне добраться туда?.....	268
Разделение запутанных обязанностей.....	269
Определение агрегатов и ограниченных контекстов.....	273

Подход на основе событий для перехода к микросервисам через паттерн «Душителъ»	277
Как убедить стейкхолдеров попробовать что-то новое	281
Вопросы наших научных редакторов, которые мы не включили в основной текст	284
Выстрел в ногу	287
Книги для обязательного прочтения.....	289
Выводы	289
Приложение А. Сводная диаграмма и таблица.....	290
Приложение Б. Шаблонная структура проекта	292
Приложение В. Замена инфраструктуры: делаем все с помощью CSV.....	302
Приложение Г. Паттерны «Репозиторий» и UoW с Django.....	308
Приложение Д. Валидация.....	318
Об авторах	329
Об обложке	330