Тестирование Django приложений

Калашников Андрей Дмитриевич

Geekbrains - Python - Django

Содержание

1. Об авторе

2. План вебинара

3. Теоритическая часть

4. Практическая часть

5. Общие рекомендации

6. Заключение

Об авторе

Профессиональная деятельность

- Программирование контроллеров
  - Систем навигации
  - Противопожарной защиты
- Специалист по BIM-технологиям
  - Дополнение функционала программ для проектирования (AutoCad, Revit)
  - Расширение функционала офисных систем
- Разработка приложений Django
  - Системы управления сервисами для операторов сотовой связи
  - Системы управления комплексом устройств (маршрутизаторы, сервера)

План вебинара

Обзор предстоящих тем

- Для кого подойдёт вебинар
- Теоритическая часть
  - Виды тестирования
  - Правила тестирования
- Практическая часть
  - Средства тестирования
  - Пакеты для тестирования
  - Поиск ошибок в Django-проектах
  - Тестирование качества кода
- Напутствие

Стоит ли смотреть дальше?

Вебинар будет Вам интересен:

- Если Вы осовоились в Python
- Если Вы только начали работать с Django
- Если Вы хотите получить обзор средств повышающих качество продукта

Теоритическая часть

Введение

- Термин ”тестирование”в рамках данного вебинара мы будем понимание как поиск ошибок в нашем коде, локализация возникших проблем.
- Вебинар подготовлен программистом для начинающих программистов

Почему это важно

- Вы сосздаёте продукт. Продукт должен обеспечивать минимальную гарантированную функциональность.
- Знание средств поможет Вам выявить ошибку быстрее и точнее. Разные средства для разных целей.
- Знание ”как?”предупредить ошибку - как Ваш hard-skill
- Написав документированные тесты вы снизите возможность провала в legacy

Виды тестирования

Важно! Данные виды тестирования актуальны для начинающих программистов. Реально список гораздо шире.

- Ручное тестирование (pentesting)
- Автоматизированное тестирование
- Тестирование интерфейса (UI testing)
- Тестирование качества кода (соблюдение PEP)

Чёрный ящик

Критерии черного ящика описывают тестирование с точки зрения поставленной задачи **без учета** внутреннего устройства программы.

Рассмотрим на примере

http://demo.viewflow.io/materialforms/registration/

Чёрный ящик: длина набора данных - 1 стр.

- Пустой набор (не содержит ни одного элемента)
- Единичный набор (состоит из одного-единственного элемента)
- Слишком короткий набор (если предусмотрена минимально допустимая длина)
- Набор минимально возможной длины (если такая предусмотрена)

Чёрный ящик: длина набора данных - 2 стр.

- Нормальный набор (состоит из нескольких элементов)
- Набор максимально возможной длины (если такая предусмотрена)
- Слишком длинный набор (с длиной больше максимально допустимой)

Разработчики ленивы

Самая обычная форма:
`http://demo.viewflow.io/materialforms/bank/`
Таких форм несколько. Различаются ”вот тем checkbox’ом, третьим снизу”. Выберите наиболее подходящий вариант развития событий:

1. Сказать, что каркас (фреймворк) за вас всё уже сделал
2. Сдать на тестирование профессионалу
3. Пустить в продакшн и фиксить замечания

Правильный ответ

Разработчик ответственен за разработанное решение. Решение не прошедшее тщательной проверки и допускающее возможную поломку - решением не является. При передаче решения можно учесть следующее:

- Для тестировщика: я реализовал функционал и по идее оно должно работать вот так ...
- Для менеджера: я реализовал функционал в соответствии с ТЗ и все крайние случаи отсёк (гарантия, что не сломается)

Практическая часть

О чём этот раздел

Обзор средств поиска ошибок и отладки программ. По каждому можно найти огромное количество материалов и лучшие практики (порой противоположные)

Пример проекта

https://github.com/bartromgens/django-project-template

Режим интерактивного ввода Python

- Сценарии: быстро проверить рузальтат операций, проверить выборку (в случае с работой с базой данных)
- Плюсы: выполнение кода без вреда основному листингу программы, подключение в Django
- Минусы: невозможно переменные доступны только на текущий сеанс, сложно набрать много кода

Функция print()

- Сценарии: вывести значение переменной в момент исполнения кода
- Плюсы: простой способ, можно использовать форматирование строк
- Минусы: потенциальный источник ошибок, неприемлим для ”боя”(Django), сложные структуры распечатываются inline (use pprint instead)

Модуль logging

- Сценарии: вывод сообщений, разделённых по важности с тонкой настройкой
- Плюсы: повсеместное использование на ”бое тонкая настройка, модуль стандартной библиотеки, вывод номера строки
- Минусы: логов много (парсинг)
- Лучшие практики: всегда пишите логи в базу

Дебаггеры - Общее

Виды:

- Консольный
- Встроенный в IDE

Дебаггеры - Консольный

- Сценарии: проверка состояния окружения исполнения
- Плюсы: нет необходимости в графическом окружении, модуль стандартной библиотеки, выручает в критических ситуациях
- Минусы: малая интерактивность, не всегда удобно

Дебаггеры - В IDE - VSCode

- Плюсы: бесплатно, intellisence в дебаг консоли, отладка шаблонов (Django), дебаг-сообщения
- Минусы: не работает с многопоточность (пока)

Дебаггеры - В IDE - Pycharm

- Плюсы: подключается к многопоточным приложениям (Celery), доступен ”из коробки”
- Минусы: нет возможности отладки шаблонов Django (в CE версии)

Модуль pickle

- Сценарий: исполняется длительный процесс или процесс со множеством элементов и необходимо узнать состояние объекта на момент исполнения
- Плюсы: отложенная работа с объектами без прерывания работы основного приложения
- Минусы: при изменении свойств или методов объекта - возможно некорректное поведение при извлечении объекта, нельзя ”законсервировать”все объекты (например открытые соединения или файлы)

Автоматизированное тестирование

- coverage (https://pypi.org/project/coverage/)
- unitest or pytest
- django tests
- selenium

Тестирование качества кода

- Линтеры: pep8, pylint, prospector
- Форматтеры: isort, black

Ваш код (возможно) будут сопровождать. Единообразие кода способствует его читаемости.

Общие рекомендации

Немного советов на будущее

- Избегайте лишних сущностей
  - Внешних зависимотей
  - Классов
- Приступая к задаче - прочувствуйте бизнес-логику (используйте псевдокод)

Заключение

СПАСИБО ЗА ВНИМАНИЕ!