Engineer

Django

D фреймворк Python для вебзастосунків

Що це таке

Django — це вебфреймворк, написаний на Python: надходить запит, конфігурація URL спрямовує його до функції або класу, який називається view, і view повертає відповідь, зазвичай HTML, зібраний із шаблону, або JSON. Навколо цього ядра постачаються частини, потрібні більшості сайтів із базою даних, серед них об'єктно-реляційне відображення (ORM), міграції схеми, форми, автентифікація з правами доступу та автоматично згенерований інтерфейс адміністратора, який власна документація Django називає однією з його найпопулярніших можливостей. Назва проєкту пишеться з великої літери, а пакет, який встановлюють та імпортують, — це `django` з малої.

Django виник у газетній редакції. Восени 2003 року Адріан Головатий і Саймон Віллісон, розробники World Online, вебпідрозділу газети в місті Лоуренс, штат Канзас, перейшли з PHP на Python для сайтів, які робили під журналістські дедлайни. Створюючи такі сайти, як Lawrence.com, вони виокремили з цієї роботи загальний фреймворк і два роки його вдосконалювали, а влітку 2005 року World Online відкрила його код. Назву дано на честь Джанго Рейнхардта, гітариста стилю джаз-мануш; англійською вона вимовляється як JANG-oh, без окремого звука D, а поширюється фреймворк за ліцензією BSD із трьома пунктами.

За нумерацією A.B функціональні випуски виходили приблизно раз на вісім місяців, і лише визначені випуски з довгостроковою підтримкою (LTS) отримували три роки підтримки; 5.2 LTS підтримується до квітня 2028 року. Django 6.2, випуск LTS, запланований на квітень 2027 року, — останній за цією схемою. Починаючи з Django 2028, функціональні випуски називатимуться за роком, виходитимуть щосічня, і кожен отримуватиме ті самі три роки підтримки.

У FAQ проєкту пропонують називати Django фреймворком MTV, від model, template і view: модель описує дані, view вирішує, які дані бачить користувач, а шаблон — як вони виглядають. Контролером у такому прочитанні є сам фреймворк, механізм, що надсилає кожен запит до потрібного view. Моделі — це класи Python, а запити до них — QuerySet, і вони ліниві: QuerySet можна створювати, фільтрувати й передавати далі, не звертаючись до бази даних, а запит до неї виконується лише тоді, коли QuerySet обчислюють, наприклад у циклі. Зміни моделей перетворюються на міграції: `makemigrations` пакує кожну зміну у файл, який документація порівнює з комітом у системі контролю версій для схеми, а `migrate` застосовує його. Пов’язані об’єкти завантажуються на вимогу, по одному запиту, якщо QuerySet не попросив їх заздалегідь.

```
# Book має зовнішній ключ ForeignKey до Author.
# 1 запит для книжок, далі по 1 на кожну книжку для її автора.
for book in Book.objects.all():
    print(book.author.name)

# 1 запит: SQL-з’єднання завантажує кожного автора разом із книжкою.
for book in Book.objects.select_related("author"):
    print(book.author.name)

# 2 запити: автори, потім усі їхні книжки разом.
for author in Author.objects.prefetch_related("book_set"):
    print(author.name, [b.title for b in author.book_set.all()])
```

Застосуйте поняття до рішення

Застосуйте «Що таке Django?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Та частина, якої користувач не бачить: де зберігаються дані, які діють правила і як інші системи можуть із нею взаємодіяти.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Спроєктувала й підтримувала 6 пайплайнів ETL/ELT — 6 пайплайнів ETL на трьох рушіях баз даних

Спроєктувала й підтримувала 6 пайплайнів ETL/ELT Зв'язатися з нами →

Чому це важливо

Django важливий тим, що він робить поведінкою за замовчуванням. QuerySet будує SQL, тримаючи параметри окремо від тексту запиту, тож значення від користувача екранує драйвер бази даних. Шаблони екранують символи, найнебезпечніші в HTML. Захист від CSRF перевіряє секрет у кожному POST-запиті, а паролі зберігаються за допомогою PBKDF2 із SHA-

256 — методу розтягування, який, як зазначає документація, рекомендує NIST. Команда, що пише звичайний код на Django, отримує цей захист, навіть не просячи про нього.

Ті самі типові налаштування визначають і те, де лежить ризик. Кожне з них скасовується одним рядком, як-от сирий запит, `mark_safe`, `csrf_exempt` чи `DEBUG = True`, що лишився на робочому сервері, і кожен такий рядок на рев'ю виглядає безневинно. ORM робить подібний обмін і з продуктивністю: легко написати view, у якому кількість запитів росте разом із довжиною сторінки, а код на Python при цьому виглядає однаково.

Наша рекомендація — обирати Django, коли застосунок визначають його дані: записи, які люди створюють і редагують, облікові записи з правами доступу і працівники, яким потрібен інтерфейс адміністратора над усім цим. Тримайте кожен проєкт на підтримуваному випуску, обирайте LTS, поки існують випуски A.B, і записуйте дату кінця підтримки поруч із версією. Запускайте `manage.py check --deploy` у конвеєрі з продакшн-налаштуваннями, тримайте секретний ключ і пароль бази даних поза репозиторієм і читайте SQL, який пише ORM, для кожної важливої сторінки. Для сайту, який клієнт здебільшого редагуватиме як сторінки, система керування вмістом або статичні сторінки підходять краще за фреймворк.

Як це виглядає на практиці

У 2015–2022 роках, на посаді веброзробниці, робота охоплювала вебзастосунки на Django навколо IPTV-платформи — ту частину цілодобового сервісу, на яку люди реально натискали: нові функції, виправлення помилок і постійну роботу над продуктивністю та стабільністю разом з усією командою. У такому сервісі, як пише сторінка, стабільність — це обмеження, якому функції мусять підпорядковуватися, і застосунки набували нових можливостей, лишаючись надійними і для внутрішніх команд, і для клієнтів.

Раніше, у 2003–2014 роках, на посаді фулстек-веброзробниці було створено сто вебзастосунків для клієнтів, які хотіли дуже різних речей. Там, де клієнту потрібне було щось індивідуальне, у хід ішли HTML/HTML5, CSS/SCSS і Django; там, де потрібне було щось стандартніше, з чим клієнт міг упоратися сам, швидшою й розумнішою відповіддю ставали WordPress або Joomla. Сторінка

називає майстерністю вміння розрізняти ці випадки, зокрема відрадити клієнта від індивідуальної розробки, яка йому не потрібна, або переконати в тій, яка потрібна. Цю саму межу проводить і наша рекомендація.

Сторінка технологій тримається обох половин цього. Вебрівень там насамперед статичний, а Django і PHP використовуються там, де є застосунок, який треба обслуговувати; у переліку мов SQL — повноцінна мова, а не те, що генерує ORM, бо саме там здебільшого й лежить робота з продуктивністю, заради якої нас запрошують. Для застосунку на Django це означає, що запити пише ORM, а читати їх однаково має людина.

Де виникають проблеми

- **Цикл виконує один запит на рядок.** У режимі вибірки за замовчуванням звернення до зовнішнього ключа, який QuerySet не завантажив, запускає запит для одного цього об'єкта, тож список зі ста книжок з їхніми авторами коштує 101 запит. У шаблонах це помітити ще важче, бо вони викликають методи на кшталт `all` без дужок. Використовуйте `select_related()` для зовнішніх ключів і полів «один до одного», а `prefetch_related()` — для полів «багато до багатьох» і зворотних зовнішніх ключів. Починаючи з Django 6.1, режим `FETCH_PEERS` об'єднує ці звернення в один додатковий запит, а `FETCH_RAISE` перетворює незапланований запит на виняток у коді, який мусить лишатися швидким.
- **Режим налагодження потрапляє в продакшн.** Контрольний список розгортання каже ніколи не вмикати `DEBUG` у продакшні, бо тоді сторінки помилок розкривають фрагменти вихідного коду, локальні змінні, налаштування й бібліотеки, що використовуються. З `DEBUG = False` Django взагалі не обслуговує запити, доки `ALLOWED_HOSTS` не має належного значення, а символ підстановки там означає, що заголовок Host доведеться перевіряти самостійно. Запускайте `manage.py check --deploy` з файлом продакшн-налаштувань: він автоматизує частину цього списку.
- **Секретний ключ лежить у репозиторії.** `SECRET_KEY` захищає сесії в більшості серверних сховищ сесій, усі токени для скидання пароля і весь криптографічний підпис, для якого не вказано іншого ключа, а довідник налаштувань попереджає, що відомий ключ може призвести до підвищення привілеїв і віддаленого виконання коду. Завантажуйте його зі змінної середовища або з файлу, не тримайте в системі контролю версій, змінюйте через `SECRET_KEY_FALLBACKS` і вчасно прибирайте старі ключі з цього списку.

- **Безпечне типове налаштування вимикають для одного view.** QuerySet параметризовані, але сирі запити, `extra()` і `RawSQL` перекладають екранування на розробника; шаблони екранують HTML, але `mark_safe`, фільтр `safe` і вимкнений autoescape теж повертають цю роботу людині; `csrf_exempt` повністю знімає перевірку з view. Кожен такий випадок заслуговує на записану причину на рев'ю, а HTML, який зберігають користувачі, варто очищати і на вході, і на виході.
- **Міграція зупиняється на півдорозі.** У PostgreSQL і SQLite усі операції міграції за замовчуванням виконуються в одній транзакції, але MySQL не підтримує транзакцій навколо змін схеми, тож міграція, що впала посередині, лишає зміни, які доведеться розбирати вручну. Читайте, що саме `makemigrations` вважає зміненим, перевіряйте SQL через `sqlmigrate`, перш ніж він потрапить у продакшн, і комітьте кожну міграцію разом зі зміною моделі.

Що почитати далі

- Django documentation — документація до поточного стабільного випуску, від туторіалу до довідника.
- Django: Deployment checklist — налаштування, які треба змінити перед запуском, частину яких автоматизує `check --deploy`.
- Django: Database access optimization — обчислення QuerySet, завантаження пов'язаних об'єктів і звички, що стримують кількість запитів.
- Django: Migrations — робочий процес міграцій, `sqlmigrate` і те, як кожен адаптер бази даних обробляє зміни схеми.
- Security in Django — від чого фреймворк захищає за замовчуванням і де цей захист закінчується.
- Download Django — підтримувані серії випусків із датами основної й розширеної підтримки.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке Django?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/django/