Engineer

Django

D et Python-framework til at bygge webapplikationer

Hvad det er

Django er et webframework skrevet i Python: en forespørgsel kommer ind, URL-konfigurationen sender den videre til en funktion eller klasse, der kaldes et view, og viewet returnerer et svar, som regel HTML fra en skabelon eller JSON. Omkring den kerne følger de dele med, som de fleste databasedrevne sites har brug for, blandt andet en objektrelationel mapper (ORM), skemamigrationer, formularer, autentificering med rettigheder og en automatisk genereret administrationsflade, som dokumentationen selv kalder en af Djangos mest populære funktioner. Projektet skrives med stort D, mens pakken, man installerer og importerer, hedder `django` med småt.

Det begyndte på en avisredaktion. I efteråret 2003 gik Adrian Holovaty og Simon Willison, udviklere hos World Online, webafdelingen på en avis i Lawrence i Kansas, fra PHP til Python på de sites, de byggede under journalistiske deadlines. Mens de byggede sites som Lawrence.com, trak de et generelt framework ud af arbejdet og finpudsede det i to år, og i sommeren 2005 frigav World Online det som open source. Det er opkaldt efter jazz manouche-guitaristen Django Reinhardt, udtales JANG-oh med stumt D og distribueres under 3-clause BSD-licensen.

Under A.B-nummereringen kom der nye feature-udgivelser cirka hver ottende måned, og kun udpegede udgivelser med long-term support (LTS) fik tre års support; 5.2 LTS understøttes til april 2028. Django 6.2, en LTS-udgivelse, der er planlagt til april 2027, er den sidste under den ordning. Fra Django 2028 navngives feature-udgivelser efter årstallet, de udkommer hver januar, og hver af dem får den samme treårige supportperiode.

Projektets FAQ foreslår at kalde Django et MTV-framework, for model, template og view: modellen definerer data, viewet bestemmer, hvilke data brugeren ser, og skabelonen bestemmer, hvordan de ser ud. Controlleren er i den læsning selve frameworket, det maskineri, der sender hver forespørgsel til det rigtige view. Modeller er Python-klasser, og forespørgsler mod dem er QuerySets, som er dovne: et QuerySet kan bygges, filtreres og sendes rundt uden at røre databasen, som først bliver spurgt, når noget evaluerer det, for eksempel en løkke. Ændringer i modellerne bliver til migrationer: `makemigrations` pakker hver ændring i en fil, som dokumentationen sammenligner med et commit i et versionsstyringssystem for skemaet, og `migrate` anvender den. Relaterede objekter hentes efter behov, én forespørgsel ad gangen, medmindre QuerySet'et beder om dem på forhånd.

```
# Book has a ForeignKey to Author.
# 1 query for the books, then 1 per book for its author.
for book in Book.objects.all():
    print(book.author.name)

# 1 query: an SQL join loads each author with its book.
for book in Book.objects.select_related("author"):
    print(book.author.name)

# 2 queries: the authors, then all their books in one batch.
for author in Author.objects.prefetch_related("book_set"):
    print(author.name, [b.title for b in author.book_set.all()])
```

Brug begrebet i en beslutning

Brug »Hvad er Django?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Den del, brugerne aldrig ser: hvor data ligger, hvilke regler der gælder, og hvordan andre systemer må tale med den.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har designet og drevet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tre databasemotorer

Har designet og drevet 6 ETL/ELT-pipelines Kontakt os →

Hvorfor det betyder noget

Django betyder noget på grund af det, frameworket gør til standard. Dets QuerySets bygger SQL med parametrene holdt adskilt fra selve forespørgselsteksten, så værdier fra brugeren escapes af databasedriveren. Skabelonerne escaper de tegn, der er farligst i HTML. CSRF-beskyttelsen tjekker en hemmelighed i hver POST-forespørgsel, og adgangskoder gemmes med

PBKDF2 over SHA-256, en metode til nøglestrækning, som dokumentationen nævner er anbefalet af NIST. Et team, der skriver almindelig Django-kode, får de beskyttelser uden at bede om dem.

De samme standarder bestemmer, hvor risikoen ligger. Hver af dem kan fjernes med én linje, for eksempel en rå forespørgsel, et `mark_safe`, et `csrf_exempt` eller `DEBUG = True`, der er blevet stående på den kørende server, og hver af de linjer ser harmløs ud i et review. ORM'en laver en lignende byttehandel med performance: det er let at skrive et view, hvis antal forespørgsler vokser med sidens længde, og Python-koden ser ens ud i begge tilfælde.

Vores anbefaling er at vælge Django, når applikationen er formet af sine data: poster, som folk opretter og redigerer, konti med rettigheder og medarbejdere, der har brug for en administrationsflade over det hele. Hold hvert projekt på en understøttet udgivelse, vælg en LTS, så længe A.B-udgivelserne varer, og skriv supportens slutdato ved siden af versionen. Kør `manage.py check --deploy` mod produktionsindstillingerne i byggepipelinen, hold den hemmelige nøgle og databasens adgangskode ude af repositoriet, og læs den SQL, ORM'en skriver, for hver side, der betyder noget. Til et site, som kunden mest skal redigere som sider, passer et indholdsstyringssystem eller statiske sider bedre end et framework.

Sådan ser det ud i praksis

Fra 2015 til 2022, som webudvikler, omfattede arbejdet Django-webapplikationer omkring en IPTV-platform, de dele af en tjeneste, der kører døgnet rundt, som folk faktisk klikkede på: nye funktioner, fejlrettelser og vedholdende arbejde med performance og stabilitet, gjort på tværs af teamet. På en sådan tjeneste, skriver siden, er stabilitet den begrænsning, funktionerne må respektere, og applikationerne fik flere funktioner og forblev samtidig pålidelige for både de interne teams og kunderne.

Tidligere, fra 2003 til 2014, som full stack-webudvikler, blev hundrede webapplikationer bygget til kunder, der ville have vidt forskellige ting. Hvor en kunde havde brug for noget skræddersyet, var værktøjerne HTML/HTML5, CSS/SCSS og Django; hvor de havde brug for noget mere standard, som de også selv kunne håndtere, var WordPress eller Joomla det hurtigere og mere fornuftige svar. Siden kalder færdigheden at vide, hvad der var hvad, også når det betød at tale en kunde fra et skræddersyet build, de ikke havde brug for, eller til et, de havde. Det er den samme grænse, vores anbefaling trækker.

Vores teknologivalg holder fast i begge halvdele. Weblaget er statisk først, med Django og PHP, hvor der er en applikation at servere, og listen over sprog behandler SQL som et fuldgyldigt sprog og ikke noget, en ORM genererer, fordi det meste af det performancearbejde, vi hentes ind til, ligger dér. For en Django-applikation betyder det, at ORM'en skriver forespørgslerne, og at nogen stadig læser dem.

Hvor det går galt

- **En løkke kører én forespørgsel pr. række.** I standardtilstanden for hentning udløser opslag i en fremmednøgle, som QuerySet'et ikke har hentet, en forespørgsel for netop det ene objekt, så en liste med hundrede bøger og deres forfattere koster 101 forespørgsler. Skabeloner gør det sværere at se, fordi de kalder metoder som `all` uden parenteser. Brug `select_related()` til fremmednøgler og en-til-en-felter og `prefetch_related()` til mange-til-mange-felter og omvendte fremmednøgler. Fra Django 6.1 samler tilstanden `FETCH_PEERS` de opslag i én ekstra forespørgsel, og `FETCH_RAISE` gør en uplanlagt forespørgsel til en undtagelse i kode, der skal forblive hurtig.
- **Debugtilstanden når produktion.** Udrulningstjeklisten siger, at `DEBUG` aldrig må slås til i produktion, fordi fejlsiderne så lækker uddrag af kildekoden, lokale variabler, indstillinger og de biblioteker, der bruges. Med `DEBUG = False` betjener Django slet ikke forespørgsler, før `ALLOWED_HOSTS` har en passende værdi, og et jokertegn dér betyder, at man selv skal validere Host-headeren. Kør `manage.py check --deploy` mod produktionens indstillingsfil; den automatiserer en del af tjeklisten.
- **Den hemmelige nøgle ligger i repositoriet.** `SECRET_KEY` beskytter sessioner under de fleste sessionsbackends, alle tokens til nulstilling af adgangskoder og al kryptografisk signering, der ikke angiver en anden nøgle, og indstillingsreferencen advarer om, at en kendt nøgle kan føre til rettighedseskalering og fjernkørsel af kode. Indlæs den fra miljøet eller en fil, hold den ude af versionsstyringen, rotér den via `SECRET_KEY_FALLBACKS`, og fjern gamle nøgler fra den liste i god tid.
- **En sikker standard slås fra for et enkelt view.** QuerySets er parametriserede, men rå forespørgsler, `extra()` og `RawSQL` overlader escaping til udvikleren; skabeloner escaper HTML, men `mark_safe`, filteret `safe` og slukket autoescape giver også den opgave tilbage; `csrf_exempt` fjerner tjekket helt fra et view. Hver af dem fortjener en begrundelse i reviewet, og HTML, som brugere gemmer, bør renses både ved input og ved output.

- **En migration stopper halvvejs.** På PostgreSQL og SQLite kører alle operationer i en migration som standard i én transaktion, men MySQL har ingen transaktioner omkring skemaændringer, så en migration, der fejler midtvejs, efterlader ændringer, der skal pilles ud i hånden. Læs, hvad `makemigrations` mener, der er ændret, tjek SQL'en med `sqlmigrate`, før den når produktion, og commit hver migration sammen med dens modelændring.

Læs mere

- Django documentation — dokumentationen til den aktuelle stabile udgivelse, fra tutorialen til referencen.
- Django: Deployment checklist — de indstillinger, der skal ændres før lancering, hvoraf `check --deploy` automatiserer en del.
- Django: Database access optimization — evaluering af QuerySets, hentning af relaterede objekter og de vaner, der holder antallet af forespørgsler nede.
- Django: Migrations — arbejdsgangen med migrationer, `sqlmigrate`, og hvordan hver databasebackend håndterer skemaændringer.
- Security in Django — hvad frameworket beskytter mod som standard, og hvor beskyttelsen holder op.
- Download Django — de understøttede udgivelsesserier med datoer for almindelig og udvidet support.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er Django?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/django/