Engineer

Django

Det Python-rammeverk for å bygge webapplikasjoner

Hva det er

Django er et webrammeverk i Python bygd rundt en enkel syklus: en forespørsel kommer inn, URL-konfigurasjonen sender den til en funksjon eller klasse som kalles et view, og viewet returnerer et svar, som regel HTML gjengitt fra en mal, eller JSON. Rundt den kjernen følger delene de fleste databasebaserte nettsteder trenger, bl.a. en objektrelasjonell mapper (ORM), migreringer for databaseskjemaet, nettskjemaer, autentisering med tilganger og et automatisk generert administrasjonsgrensesnitt som dokumentasjonen selv kaller en av de mest populære funksjonene. Prosjektet skrives med stor D, mens pakken du installerer og importerer, heter `django` med liten bokstav.

Rammeverket ble født i en avisredaksjon. Høsten 2003 gikk Adrian Holovaty og Simon Willison, utviklere i World Online, nettavdelingen til en avis i Lawrence i Kansas, over fra PHP til Python for nettstedene de bygde under journalistiske tidsfrister. Mens de lagde nettsteder som Lawrence.com, trakk de et generelt rammeverk ut av arbeidet og finpusset det i to år, og sommeren 2005 ga World Online det ut som åpen kildekode. Navnet kommer fra jazzgitaristen Django Reinhardt, mester i jazz manouche. Det uttales med stum D, omtrent som engelsk «JANG-oh», og distribueres under BSD-lisensen med tre klausuler.

Under nummereringen A.B kom en ny funksjonsversjon omtrent hver åttende måned, og bare utpekte versjoner med langtidsstøtte, LTS, fikk tre års støtte. 5.2 LTS støttes til april 2028. Django 6.2, en LTS-versjon som kommer i april 2027, er den siste under den ordningen. Fra Django 2028 får funksjonsversjonene navn etter året, kommer hver januar og får alle den samme støtteperioden på tre år.

Prosjektets FAQ foreslår å kalle Django et MTV-rammeverk, for model, template og view. Modellen definerer dataene, viewet avgjør hvilke data brukeren får se, og malen avgjør hvordan de ser ut. Kontrolleren er i den lesningen selve

rammeverket, maskineriet som sender hver forespørsel til riktig view.

Modeller er Python-klasser, og spørringer mot dem er QuerySets. De er late: et QuerySet kan bygges, filtreres og sendes rundt uten å røre databasen, som først får spørsmålet når noe evaluerer QuerySetet, f.eks. en løkke. Endringer i modellene blir migreringer. `makemigrations` pakker hver endring inn i en fil som dokumentasjonen sammenligner med en commit i et versjonskontrollsystem for databaseskjemaet, og `migrate` tar den i bruk. Relaterte objekter lastes ved behov, én spørring om gangen, med mindre QuerySetet ber om dem på forhånd:

```
# Book har en ForeignKey til Author.
# 1 spørring for bøkene, så 1 per bok for forfatteren.
for book in Book.objects.all():
    print(book.author.name)

# 1 spørring: en SQL-join laster hver forfatter sammen med boka.
for book in Book.objects.select_related("author"):
    print(book.author.name)

# 2 spørringer: forfatterne, så alle bøkene deres i én omgang.
for author in Author.objects.prefetch_related("book_set"):
    print(author.name, [b.title for b in author.book_set.all()])
```

Bruk begrepet i en beslutning

Bruk «Hva er Django?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Delen brukerne aldri ser: hvor dataene ligger, hvilke regler som gjelder, og hvordan andre systemer får lov til å snakke med den.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Designet og driftet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tvers av tre databasemotorer

Designet og driftet 6 ETL/ELT-pipelines Ta kontakt →

Hvorfor det er viktig

Django betyr noe på grunn av det rammeverket gjør til standard. QuerySets bygger SQL med parameterne holdt atskilt fra spørringsteksten, så verdier en bruker sender inn, escapes av databasedriveren. Malene escaper de tegnene som er farligst i HTML. CSRF-beskyttelsen sjekker en hemmelighet ved hver POST, og passord lagres med PBKDF2 over SHA-256, en metode for nøkkelstrekking som dokumentasjonen påpeker at NIST anbefaler. Et team som skriver helt vanlig Django-kode, får de beskyttelsene uten å be om dem.

De samme standardene bestemmer også formen på risikoen. Hver av dem kan oppheves med én linje, f.eks. en rå spørring, en `mark_safe`, en `csrf_exempt` eller `DEBUG = True` som blir stående på serveren i drift, og hver av de linjene ser harmløs ut i en kodegjennomgang. ORM-en gjør en lignende avveining med ytelse: det er lett å skrive et view der antallet spørringer vokser med lengden på siden, og Python-koden ser lik ut uansett.

Vår anbefaling er å velge Django når applikasjonen er formet av dataene sine: poster som folk oppretter og redigerer, kontoer med tilganger og ansatte som trenger et administrasjonsbilde over alt sammen. Videre:

- Hold hvert prosjekt på en versjon som fortsatt støttes, velg en LTS-versjon så lenge A.B-versjonene varer, og skriv sluttdatoen for støtten ved siden av versjonsnummeret.
- Kjør `manage.py check --deploy` mot produksjonsinnstillingene i byggepipelinen.
- Hold den hemmelige nøkkelen og databasepassordet ute av repoet.
- Les SQL-en ORM-en skriver, for hver side som betyr noe.

For et nettsted som kunden stort sett skal redigere som sider, passer et publiseringssystem eller statiske sider bedre enn et rammeverk.

Slik ser det ut i praksis

Fra 2015 til 2022, i rollen som webutvikler, omfattet arbeidet Django-webapplikasjoner rundt en IPTV-plattform, de delene av en tjeneste i drift døgnet rundt som folk faktisk klikket på: nye funksjoner, feilrettinger og jevnt arbeid med ytelse og stabilitet, gjort i samarbeid på tvers av teamet. På en slik tjeneste, sier siden, er stabiliteten rammen funksjonene må respektere, og applikasjonene fikk stadig nye egenskaper og holdt seg samtidig pålitelige for både de interne teamene og kundene.

Tidligere, fra 2003 til 2014, i rollen som fullstack-webutvikler, ble hundre webapplikasjoner bygd for kunder som ville ha svært forskjellige ting. Der en kunde trengte noe skreddersydd, ble verktøyene HTML/HTML5, CSS/SCSS og Django; der de trengte noe mer standard som de også kunne administrere selv, var WordPress eller Joomla det raskere og mer fornuftige svaret. Siden kaller ferdigheten å vite hva som var hva, også å få en kunde til å droppe en skreddersydd løsning de ikke trengte, eller til å velge en de trengte. Det er den samme grensen anbefalingen vår trekker.

Teknologisiden holder på begge halvdelene. Weblaget er statisk først, med Django og PHP der det er en applikasjon å levere, og språklisten behandler SQL som et fullverdig språk og ikke noe en ORM genererer, fordi det meste av ytelsesarbeidet vi blir hentet inn for, ligger der. For en Django-applikasjon betyr det at ORM-en skriver spørringene, og at noen fortsatt leser dem.

Hvor det går galt

- **En løkke kjører én spørring per rad.** I standardmodusen for henting fører det til én spørring for akkurat det objektet når koden leser en fremmednøkkel som QuerySetet ikke lastet. En liste med hundre bøker og forfatterne deres koster dermed 101 spørringer. Maler gjør det vanskeligere å se, fordi de kaller metoder som `all` uten parenteser. Bruk `select_related()` for fremmednøkler og en-til-en-felt, og `prefetch_related()` for mange-til-mange-felt og omvendte fremmednøkler. Fra og med Django 6.1 samler modusen `FETCH_PEERS` slike oppslag i én ekstra spørring, og `FETCH_RAISE` gjør en uplanlagt spørring om til et unntak i kode som må holde seg rask.
- **Feilsøkingsmodus når produksjon.** Sjekklisten for produksjonssetting sier at `DEBUG` aldri skal slås på i produksjon, fordi feilsidene lekker utdrag av kildekoden, lokale variabler, innstillinger og bibliotekene som er i bruk. Med `DEBUG = False` betjener Django ingen forespørsler i det hele tatt før `ALLOWED_HOSTS` har en passende verdi, og et jokertegn der betyr at du må validere Host-headeren selv. Kjør `manage.py check --deploy` mot innstillingsfilen for produksjon; kommandoen automatiserer en del av sjekklisten.
- **Den hemmelige nøkkelen ligger i repoet.** `SECRET_KEY` beskytter sesjoner under de fleste sesjonsbackender, hvert token for tilbakestilling av passord og all kryptografisk signering som ikke oppgir en annen nøkkel. Referansen for innstillinger advarer om at en kjent nøkkel kan føre til eskalering av privilegier og fjernkjøring av kode. Last den fra miljøet eller fra en fil, hold den utenfor versjonskontroll, roter den via `SECRET_KEY_FALLBACKS`, og fjern gamle nøkler fra den listen raskt.
- **En trygg standard slås av for ett view.** QuerySets er parameteriserte, men rå spørringer, `extra()` og `RawSQL` overlater escaping til utvikleren. Maler escaper HTML, men `mark_safe`, filteret `safe` og avslått autoescape gir også den jobben tilbake. `csrf_exempt` fjerner sjekken helt fra et view. Hver av dem fortjener en uttalt begrunnelse i kodegjennomgangen, og HTML som brukere lagrer, bør renses både når den tas imot og når den vises.
- **En databasemigrering stopper halvveis.** På PostgreSQL og SQLite kjører hver operasjon i en migrering som standard i én transaksjon, men MySQL har ingen transaksjoner rundt skjemaendringer. En migrering som feiler underveis, etterlater da endringer som må plukkes fra hverandre for hånd. Les hva `makemigrations` mener er endret, sjekk SQL-en med `sqlmigrate` før den når produksjon, og commit hver migrering sammen med modellendringen sin.

Les mer

- Django documentation — dokumentasjonen for gjeldende stabile versjon, fra veiledningen til referansen.
- Django: Deployment checklist — innstillingene som må endres før lansering, og som `check --deploy` delvis automatiserer.
- Django: Database access optimization — evaluering av QuerySets, henting av relaterte objekter og vanene som holder antallet spørringer flatt.
- Django: Migrations — arbeidsflyten for migreringer, `sqlmigrate`, og hvordan hver databasebackend håndterer skjemaendringer.
- Security in Django — hva rammeverket beskytter mot som standard, og hvor den beskyttelsen slutter.
- Download Django — versjonsseriene som støttes, med datoene for ordinær og utvidet støtte.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er Django?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/django/