Engineer

Django

D ett Python-ramverk för att bygga webbapplikationer

Vad det är

Django är ett webbramverk skrivet i Python. Ett anrop kommer in, URL-konfigurationen skickar det vidare till en funktion eller klass som kallas vy, och vyn returnerar ett svar, oftast HTML som har renderats från en mall eller JSON. Runt den kärnan levereras de delar som de flesta webbplatser med en databas bakom behöver, bl.a. en objektrelationell mappare, migreringar av databasschemat, formulär, autentisering med behörigheter och ett automatiskt genererat administrationsgränssnitt som den egna dokumentationen kallar en av de mest populära funktionerna. Projektet skrivs med stort D, medan paketet som installeras och importeras heter `django` med litet d.

Ramverket kommer från en nyhetsredaktion. Hösten 2003 gick Adrian Holovaty och Simon Willison, utvecklare på World Online, webbavdelningen på en tidning i Lawrence i Kansas, över från PHP till Python för de webbplatser de byggde under journalistikens deadlines. Medan de byggde webbplatser som Lawrence.com lyfte de ut ett generellt ramverk ur arbetet, förfinade det i två år, och sommaren 2005 släppte World Online det som öppen källkod. Det är uppkallat efter Django Reinhardt, gitarristen inom jazz manouche, uttalas ungefär ”JANG-oh” med stumt D och distribueras under BSD-licensen med tre klausuler.

Under numreringen A.B kom funktionsreleaser ungefär var åttonde månad, och bara utsedda releaser med långtidsstöd, LTS, fick tre års stöd; 5.2 LTS stöds till april 2028. Django 6.2, en LTS-release som väntas i april 2027, blir den sista under den policyn. Från och med Django 2028 namnges funktionsreleaserna efter sitt år, kommer varje januari och får alla samma stödperiod på tre år.

Projektets FAQ föreslår att Django kallas ett MTV-ramverk, för model, template och view: modellen definierar datan, vyn avgör vilken data användaren får se och mallen avgör hur den ser ut. Controllern är i den läsningen ramverket självt,

maskineriet som skickar varje anrop till rätt vy. Modeller är Python-klasser, och databasfrågor mot dem är QuerySets, som är lata: en QuerySet kan byggas, filtreras och skickas runt utan att röra databasen, som frågas först när något utvärderar den, t.ex. en loop. Ändringar i modellerna blir migreringar. `makemigrations` packar varje ändring i en fil som dokumentationen jämför med en commit i ett versionshanteringssystem för databasschemat, och `migrate` tillämpar den. Relaterade objekt läses in vid behov, en databasfråga i taget, om inte QuerySet-objektet ber om dem i förväg.

```
# 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()])
```

Använd begreppet i ett beslut

Använd ”Vad är Django?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Den del användarna aldrig ser: där data lagras, vilka reglerna är och hur andra system får kommunicera med den.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Utformade och drev 6 ETL/ELT-pipelines — 6 ETL-pipelines över tre databasmotorer

Utformade och drev 6 ETL/ELT-pipelines Hör av er →

Varför det spelar roll

Django spelar roll genom det som ramverket gör till standard. QuerySets bygger SQL med parametrarna åtskilda från frågetexten, så värden som en användare skickar in escapas av databasdrivrutinen. Mallarna escapar de tecken som är farligast i HTML. Skyddet mot CSRF kontrollerar en hemlighet vid varje POST, och lösenord lagras med PBKDF2 över SHA-256, en metod för nyckelförlängning som enligt dokumentationen rekommenderas av NIST. Ett team som skriver vanlig Django-kod får de skydden utan att be om dem.

Samma standardval bestämmer också hur risken ser ut. Vart och ett kan upphävas med en enda rad, som en rå databasfråga, ett `mark_safe`, ett `csrf_exempt` eller `DEBUG = True` som lämnats på i den skarpa servern, och var och en av de raderna ser harmlös ut i en kodgranskning. ORM:en gör en liknande avvägning när det gäller prestanda: det är lätt att skriva en vy där antalet databasfrågor växer med sidans längd, och Python-koden ser likadan ut i båda fallen.

Vår rekommendation är att välja Django när applikationen formas av sin data: poster som människor skapar och redigerar, konton med behörigheter och personal som behöver en administrationsvy över alltihop. Håll varje projekt på en release som stöds, välj en LTS-release så länge A.B-releaserna finns kvar och skriv datumet då stödet upphör bredvid versionen. Kör `manage.py check --deploy` mot produktionsinställningarna i pipelinen, håll den hemliga nyckeln och lösenordet till databasen borta från repot och läs den SQL som ORM:en skriver för varje sida som betyder något. För en webbplats som kunden mest kommer att redigera som sidor passar ett publiceringssystem eller statiska sidor bättre än ett ramverk.

Så ser det ut i praktiken

Från 2015 till 2022, i rollen som webbutvecklare, ingick arbete med webbapplikationer i Django runt en IPTV-plattform, de delar av en tjänst i drift dygnet runt som människor faktiskt klickade på: nya funktioner, buggrättningar och ett stadigt arbete med prestanda och stabilitet, gjort tillsammans med hela teamet. På en sådan tjänst, säger sidan, är stabiliteten det villkor som funktionerna måste respektera, och applikationerna fick hela tiden nya förmågor medan de förblev tillförlitliga för både interna team och kunder.

Tidigare, från 2003 till 2014, som fullstackutvecklare för webben, byggdes hundra webbapplikationer åt kunder som ville ha väldigt olika saker. Där en kund behövde något skräddarsytt blev verktygen HTML/HTML5, CSS/SCSS och Django; där kunden behövde något mer standardmässigt som den också kunde sköta själv var WordPress eller Joomla det snabbare och förnuftigare svaret. Sidan beskriver kunnandet som att veta vilket som var vilket, och att ibland få en kund att avstå från ett specialbygge den inte behövde, eller att övertyga den om ett som den faktiskt behövde. Det är samma gräns som vår rekommendation drar.

Sidan om teknikstacken håller kvar båda halvorna. Dess webbskikt är statiskt först, med Django och PHP där det finns en applikation att leverera, och dess lista över språk behandlar SQL som ett fullvärdigt språk snarare än något en ORM

genererar, eftersom det mesta av det prestandaarbete vi kallas in för finns där. För en applikation i Django betyder det att ORM:en skriver databasfrågorna och att någon fortfarande läser dem.

Var det går fel

- **En loop kör en databasfråga per rad.** I standardläget för hämtning kör en läsning av en främmande nyckel som QuerySet-objektet inte läste in en databasfråga för just det objektet, så en lista med hundra böcker och deras författare kostar 101 databasfrågor. Mallar gör det svårare att se, eftersom de anropar metoder som `all` utan parenteser. Använd `select_related()` för främmande nycklar och ett-till-ett-fält och `prefetch_related()` för många-till-många-fält och omvända främmande nycklar. I Django 6.1 och senare samlar läget `FETCH_PEERS` de uppslagen i en enda extra databasfråga, och `FETCH_RAISE` gör en oplanerad databasfråga till ett undantag i kod som måste förbli snabb.
- **Felsökningsläget når produktionen.** Checklistan för driftsättning säger att `DEBUG` aldrig ska slås på i produktion, eftersom felsidorna läcker utdrag ur källkoden, lokala variabler, inställningar och de bibliotek som används. Med `DEBUG = False` svarar Django på anrop först när `ALLOWED_HOSTS` innehåller ett lämpligt värde, och ett jokertecken där innebär att du själv måste validera Host-headern. Kör `manage.py check --deploy` mot filen med produktionsinställningar; kommandot automatiserar en del av checklistan.
- **Den hemliga nyckeln ligger i repot.** `SECRET_KEY` skyddar sessioner i de flesta sessionsbackends, varje token för återställning av lösenord och all kryptografisk signering som inte anger någon annan nyckel, och referensen för inställningarna varnar för att en känd nyckel kan leda till eskalering av privilegier och fjärrkörning av kod. Läs in den från miljön eller en fil, håll den utanför versionshanteringen, rotera den via `SECRET_KEY_FALLBACKS` och ta snabbt bort gamla nycklar från den listan.
- **Ett säkert standardval stängs av för en enda vy.** QuerySets är parametriserade, men råa databasfrågor, `extra()` och `RawSQL` lämnar escapningen åt utvecklaren; mallar escapar HTML, men `mark_safe`, filtret `safe` och avstängd autoescape lämnar också tillbaka det jobbet; `csrf_exempt` tar bort kontrollen från en vy helt och hållet. Vart och ett förtjänar ett uttalat skäl i kodgranskningen, och HTML som användare lagrar bör saneras både vid inmatning och vid utmatning.
- **En databasmigrering stannar halvvägs.** På PostgreSQL och SQLite körs varje operation i en migrering som standard inom en och samma transaktion, men MySQL kör ändringar i schemat utanför transaktioner, så en migrering som fallerar på vägen lämnar ändringar efter sig som måste redas ut för hand. Läs vad `makemigrations` tror har ändrats, granska SQL-koden med `sqlmigrate` innan den når produktion och committa varje migrering tillsammans med sin ändring i modellen.

Läs mer

- Django documentation — dokumentationen för den nuvarande stabila releasen, från handledningen till referensen.
- Django: Deployment checklist — inställningarna som ska ändras före lansering, varav en del automatiseras av `check --deploy`.
- Django: Database access optimization — utvärdering av QuerySets, hämtning av relaterade objekt och de vanor som håller antalet databasfrågor jämnt.
- Django: Migrations — arbetsflödet för migreringar, `sqlmigrate` och hur varje databasbackend hanterar ändringar i schemat.
- Security in Django — vad ramverket skyddar mot som standard och var det skyddet tar slut.
- Download Django — de releaseserier som stöds, med datum för ordinarie och förlängt stöd.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är Django?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/django/