Engineer

Django

D ein Python-Framework zum Aufbau von Webanwendungen

Was es ist

Django ist ein in Python geschriebenes Webframework: Eine Anfrage trifft ein, die URL-Konfiguration leitet sie an eine Funktion oder Klasse namens View weiter, und diese liefert eine Antwort zurück, meist aus einer Vorlage gerendertes HTML oder JSON. Um diesen Kern herum bringt es die Teile mit, die die meisten datenbankgestützten Websites brauchen: einen objektrelationalen Mapper, Schemamigrationen, Formulare, Authentifizierung mit Berechtigungen und eine automatisch erzeugte Administrationsoberfläche, die seine Dokumentation zu seinen beliebtesten Funktionen zählt. Das Projekt schreibt sich mit großem D; das installierte und importierte Paket wird dagegen kleingeschrieben: `django` .

Es entstand in einer Nachrichtenredaktion. Im Herbst 2003 wechselten Adrian Holovaty und Simon Willison, Entwickler bei World Online, der Webabteilung einer Zeitung in Lawrence, Kansas, für ihre unter journalistischem Termindruck gebauten Websites von PHP zu Python. Beim Aufbau von Websites wie Lawrence.com lösten sie ein allgemeines Framework aus der Arbeit heraus, verfeinerten es zwei Jahre lang, und im Sommer 2005 veröffentlichte World Online es als Open Source. Es ist nach dem Jazz-Manouche-Gitarristen Django Reinhardt benannt, wird JANG-oh mit stummem D ausgesprochen und unter der BSD-Lizenz mit 3 Klauseln verbreitet.

Unter der A.B-Nummerierung erschienen Funktionsversionen ungefähr alle acht Monate. Nur ausgewählte Versionen mit Langzeitunterstützung erhielten drei Jahre Support; 5.2 LTS wird bis April 2028 unterstützt. Django 6.2, eine für April 2027 vorgesehene LTS-Version, ist die letzte nach dieser Regel. Ab Django 2028 tragen Funktionsversionen das Jahr im Namen, erscheinen jeden Januar und erhalten alle dieselbe dreijährige Unterstützung.

Die FAQ des Projekts schlagen die Bezeichnung MTV-Framework für Model, Template und View vor: Das Model definiert die Daten, die View entscheidet, welche Daten der Nutzer sieht, und das Template legt das Aussehen fest. Der Controller ist in dieser Lesart das Framework selbst, also der Mechanismus, der jede Anfrage zur richtigen View schickt. Models sind Python-Klassen; Abfragen darauf heißen QuerySets und werden verzögert ausgewertet. Ein QuerySet lässt sich aufbauen, filtern und weiterreichen, ohne die Datenbank anzufassen. Sie wird erst abgefragt, wenn etwas es auswertet, etwa eine Schleife. Modeländerungen werden zu Migrationen: `makemigrations` verpackt jede Änderung in eine Datei, die die Dokumentation mit einem Commit in einer Versionsverwaltung für das Schema vergleicht. `migrate` wendet sie an. Verknüpfte Objekte werden bei Bedarf durch einzelne Abfragen geladen, sofern das QuerySet sie nicht vorab anfordert.

```
# 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()])
```

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Django?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Der Teil, den Nutzer nie sehen: wo die Daten liegen, welche Regeln gelten und wie andere Systeme darauf zugreifen dürfen.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 6 ETL/ELT-Pipelines entworfen und betrieben — 6 ETL-Pipelines über drei Datenbankengines hinweg

6 ETL/ELT-Pipelines entworfen und betrieben Kontakt aufnehmen →

Warum es wichtig ist

Djangos Bedeutung liegt in seinen Voreinstellungen. Seine QuerySets bauen SQL auf und halten die Parameter vom Abfragetext getrennt, sodass vom Nutzer gelieferte Werte durch den Datenbanktreiber maskiert werden. Seine Templates maskieren die in HTML gefährlichsten Zeichen. Der CSRF-Schutz prüft bei jedem

POST ein Geheimnis, und Passwörter werden mit PBKDF2 über SHA-256 gespeichert, einem Verfahren zur Schlüsselstreckung, das laut Dokumentation von NIST empfohlen wird. Ein Team erhält diese Schutzmechanismen mit gewöhnlichem Django-Code, ohne sie eigens anfordern zu müssen.

Dieselben Voreinstellungen prägen das Risiko. Jede lässt sich mit einer Zeile aushebeln, etwa einer direkten SQL-Abfrage, `mark_safe`, `csrf_exempt` oder einem auf dem Liveserver verbliebenen `DEBUG = True`. Jede solche Zeile wirkt im Review harmlos. Das ORM geht einen ähnlichen Kompromiss bei der Leistung ein: Eine View, deren Abfragezahl mit der Seitenlänge wächst, ist schnell geschrieben, und der Python-Code liest sich in beiden Fällen gleich.

Unsere Empfehlung lautet, Django zu wählen, wenn die Daten die Anwendung prägen: Datensätze, die Menschen anlegen und bearbeiten, Konten mit Berechtigungen und Mitarbeiter, die eine Administrationsoberfläche für all das brauchen. Halten Sie jedes Projekt auf einer unterstützten Version, wählen Sie während der verbleibenden A.B-Versionen eine LTS und notieren Sie das Supportende neben der Version. Führen Sie `manage.py check --deploy` in der Pipeline gegen die Produktionseinstellungen aus, halten Sie Geheimschlüssel und Datenbankpasswort aus dem Repository heraus und lesen Sie für jede wichtige Seite das vom ORM erzeugte SQL. Für eine Website, die der Kunde hauptsächlich durch das Bearbeiten von Seiten pflegen wird, passen ein Content-Management-System oder statische Seiten besser als ein Framework.

Wie es in der Praxis aussieht

Von 2015 bis 2022 umfasste die Arbeit als Webentwicklerin Django-Webanwendungen rund um eine IPTV-Plattform: die Teile eines rund um die Uhr betriebenen Dienstes, auf die Menschen tatsächlich klickten. Dazu gehörten neue Funktionen, Fehlerkorrekturen sowie kontinuierliche Arbeit an Leistung und Stabilität im gesamten Team. Bei einem solchen Dienst ist laut Seite Stabilität die Grenze, die Funktionen einhalten müssen. Die Anwendungen gewannen weiter an Fähigkeiten und blieben dabei für interne Teams und Kunden zuverlässig.

Zuvor gehörte von 2003 bis 2014 der Bau von hundert Webanwendungen für Kunden mit sehr unterschiedlichen Wünschen zur Arbeit als Full-Stack-Webentwicklerin. Wo ein Kunde eine individuelle Lösung brauchte, waren die Werkzeuge HTML/HTML5, CSS/SCSS und Django. Wo er etwas Standardmäßigeres brauchte, das er auch selbst verwalten konnte, waren WordPress oder Joomla die schnellere, sinnvollere Antwort. Die Seite beschreibt

die Fähigkeit, beides zu unterscheiden. Dazu gehört, einem Kunden eine unnötige Individualentwicklung auszureden oder ihm zu einer nötigen zu raten. Genau diese Grenze zieht unsere Empfehlung.

Die Technologiestack-Seite hält beides fest. Ihre Webschicht setzt zuerst auf statische Seiten, mit Django und PHP dort, wo eine Anwendung zu betreiben ist. Ihre Sprachenliste behandelt SQL als vollwertige Sprache statt als etwas, das ein ORM erzeugt. Denn dort liegt der größte Teil der Arbeit an der Performance, für die wir hinzugezogen werden. Für eine Django-Anwendung heißt das: Das ORM schreibt die Abfragen, und jemand liest sie weiterhin.

Wo es schiefgeht

- **Eine Schleife führt pro Zeile eine Abfrage aus.** Im standardmäßigen Abrufmodus löst das Lesen eines vom QuerySet nicht geladenen Fremdschlüssels eine Abfrage für dieses eine Objekt aus. Eine Liste mit hundert Büchern samt Autoren kostet so 101 Abfragen. Templates machen das schwerer sichtbar, weil sie Methoden wie `all` ohne Klammern aufrufen. Nutzen Sie `select_related()` für Fremdschlüssel und Eins-zu-eins-Felder sowie `prefetch_related()` für Viele-zu-viele-Felder und umgekehrte Fremdschlüsselbeziehungen. Ab Django 6.1 bündelt der Modus `FETCH_PEERS` diese Zugriffe in einer zusätzlichen Abfrage. `FETCH_RAISE` macht in Code, der schnell bleiben muss, eine ungeplante Abfrage zu einer Ausnahme.
- **Der Debugmodus erreicht Produktion.** Die Bereitstellungscheckliste verbietet `DEBUG` in Produktion, weil seine Fehlerseiten Quellcodeauszüge, lokale Variablen, Einstellungen und verwendete Bibliotheken preisgeben. Mit `DEBUG = False` bedient Django überhaupt keine Anfragen, bis `ALLOWED_HOSTS` einen passenden Wert enthält. Ein Platzhalter dort verlangt eine eigene Prüfung des Host-Headers. Führen Sie `manage.py check --deploy` gegen die Datei mit Produktionseinstellungen aus; der Befehl automatisiert einen Teil der Checkliste.
- **Der Geheimschlüssel liegt im Repository.** `SECRET_KEY` schützt Sitzungen bei den meisten Sitzungsbackends, jedes Passwortzurücksetzungstoken und alle kryptografischen Signaturen ohne eigens genannten anderen Schlüssel. Die Einstellungsreferenz warnt, dass ein bekannter Schlüssel Rechteausweitung und entfernte Codeausführung ermöglichen kann. Laden Sie ihn aus der Umgebung oder einer Datei, halten Sie ihn aus der Versionsverwaltung heraus, rotieren Sie ihn mit `SECRET_KEY_FALLBACKS` und entfernen Sie alte Schlüssel zügig aus dieser Liste.

- **Eine sichere Voreinstellung wird für eine View abgeschaltet.** QuerySets sind parametrisiert. Direkte SQL-Abfragen, `extra()` und `RawSQL` überlassen die Maskierung aber dem Entwickler. Templates maskieren HTML, doch `mark_safe`, der Filter `safe` und abgeschaltetes Autoescaping geben auch diese Aufgabe zurück. `csrf_exempt` entfernt die Prüfung einer View vollständig. Jede solche Änderung verdient eine ausdrücklich genannte Begründung im Review. Von Nutzern gespeichertes HTML sollte sowohl bei der Eingabe als auch bei der Ausgabe bereinigt werden.
- **Eine Migration bleibt auf halbem Weg stehen.** Auf PostgreSQL und SQLite laufen standardmäßig alle Operationen einer Migration in einer Transaktion. MySQL hat dagegen keine Transaktionen um Schemaänderungen. Eine teilweise fehlgeschlagene Migration hinterlässt deshalb Änderungen, die von Hand entwirrt werden müssen. Lesen Sie, was sich nach Ansicht von `makemigrations` geändert hat, prüfen Sie das SQL mit `sqlmigrate`, bevor es in Produktion gelangt, und committen Sie jede Migration zusammen mit ihrer Modeländerung.

Weiterführende Lektüre

- Django-Dokumentation — die Dokumentation der aktuellen stabilen Version vom Tutorial bis zur Referenz.
- Django: Bereitstellungscheckliste — die vor dem Livegang zu ändernden Einstellungen, teilweise durch `check --deploy` automatisiert.
- Django: Datenbankzugriffe optimieren — QuerySet-Auswertung, Abruf verknüpfter Objekte und Gewohnheiten für eine gleichbleibende Abfragezahl.
- Django: Migrationen — der Migrationsablauf, `sqlmigrate` und der Umgang jeder Datenbank-Engine mit Schemaänderungen.
- Sicherheit in Django — wovor das Framework standardmäßig schützt und wo dieser Schutz endet.
- Django herunterladen — die unterstützten Versionsreihen samt Terminen für regulären und erweiterten Support.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Django?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/django/