Gruppe Antira - Daniel Szygula, Fabian Löffler, Luke Mikat, Marlon Pantazescu, Mike Sommer - SoSe 2021

Inhaltsverzeichnis

1. Was ist Django?

Python Web Framework

Initial release:	21 July 2005 (about 15-16 years ago)
Original Authors:	Adrian Holovaty, Simon Willison
Developer:	Django Software Foundation (Juni 2008)
Newest stable release:	3.2.3 (since May 13, 2021)
Planned releases:	4.0 (Dec 2021) - 5.0 (Dec 2023)

1. Was ist ein Framework?

Deutsch: Rahmenstruktur

Liefert unterstützende Development Tools

- Web Ressourcen
- Services
- API’s
- Datenbanken
- Libraries
- Pre-build Templates
- Session Management Tools

Schnelleres Programmieren durch Code Recycling

1. Framework für nicht-Programmierer

Beispiel:

Holzrahmenbau
(Steckhäuser)

1. Was ist ein Framework?

“Ein Framework ist eine semi-vollständige Applikation. Es stellt für Applikationen eine wiederverwendbare, gemeinsame Struktur zur Verfügung.
Die Entwickler bauen das Framework in ihre eigene Applikation ein und erweitern es derart, dass es ihren spezifischen Anforderungen entspricht.
Frameworks unterscheiden sich von Toolkits dahingehend, dass sie eine kohärente Struktur zur Verfügung stellen, anstatt einer einfachen Menge von Hilfsklassen.“

- Ralph E. Johnson, Brian Foote: “Designing Reusable Classes” im "Journal of Object-Oriented Programming" (1988)

1. Framework Varianten: Server & Client seitig

Clients: Chrome, Firefox, Safari, etc.

Web Server: Apache, NGINX, etc.

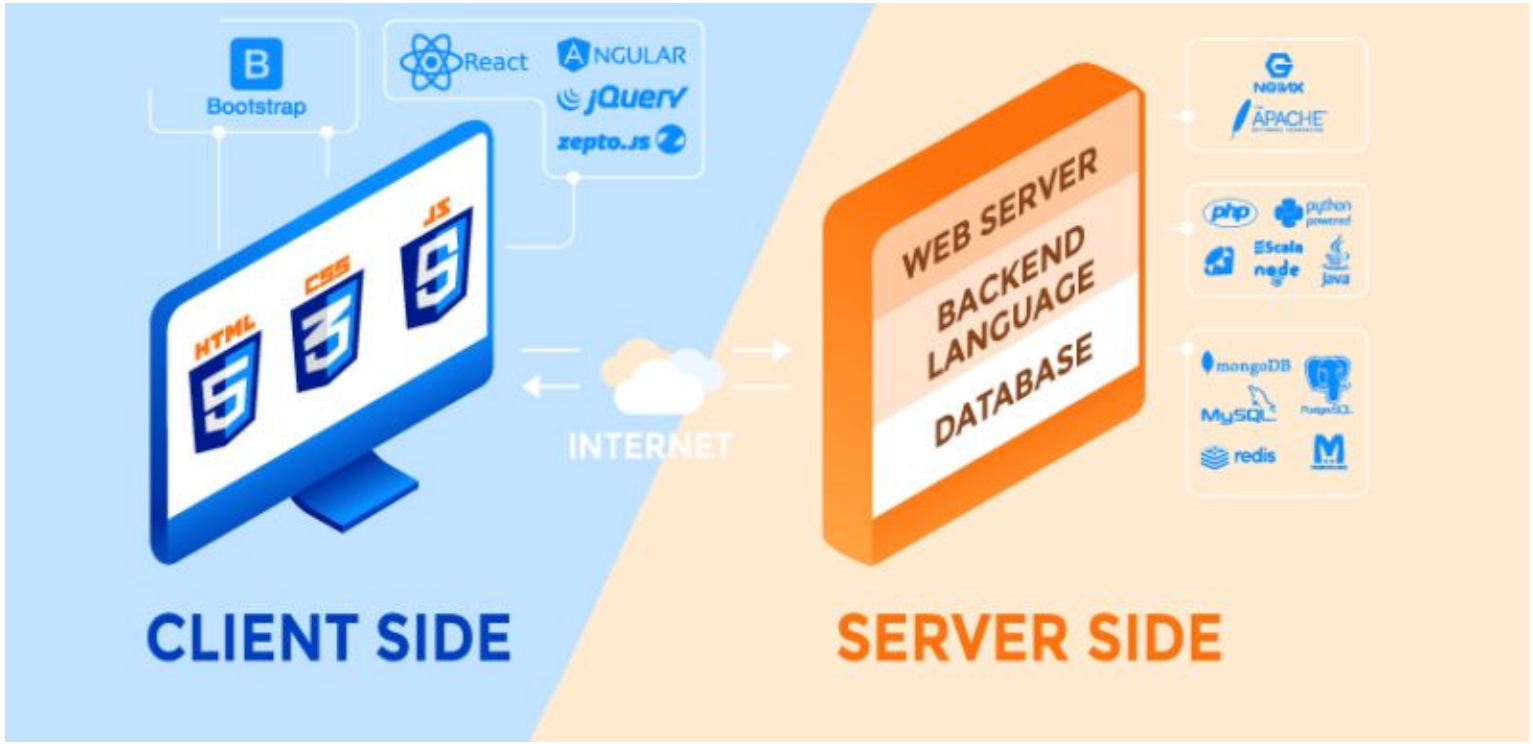
Developer bezeichnen die beiden Scripting-Techniken auch als

Frontend (Client-side Scripting) &

Backend (Server-side Scripting)

Django ist ein Server seitiges (Backend) Framework

Frontend / Backend



Client-side Scripting (Frontend)

Scripts laufen im Browser

Anfragen werden vom User an den Client gestellt

Daten werden im Client hinterlegt

Sprachen: HTML, CSS, JavaScript

Server-side Scripting (Backend)

Scripts laufen auf dem Server

Anfragen werden vom Client an den Server gestellt

Ruft Informationen und Daten vom Server ab

Daten werden auf dem Web Server gespeichert

Sprachen: PHP, Python, Java, Ruby, ASP.NET, JavaScript

2. Erste Schritte mit Django

- **Erstellen einer virtuellen Umgebung**

  $ python -m venv ./venv

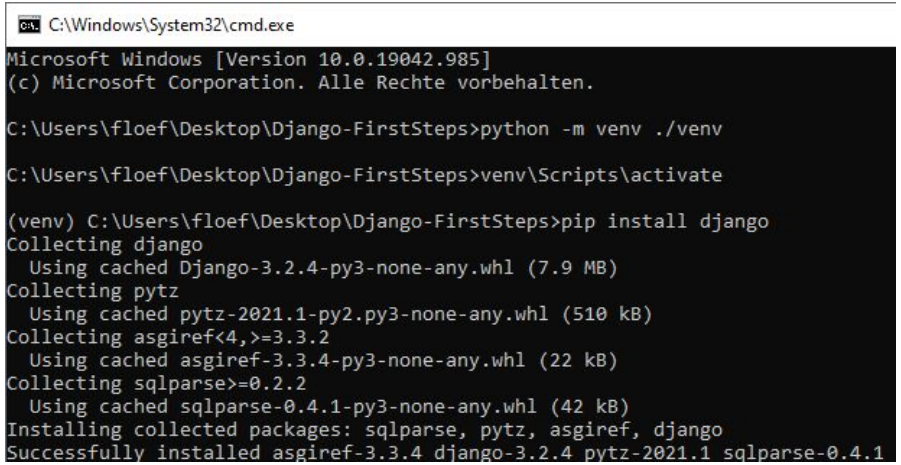
- **virtuelle Umgebung aktivieren**

  Linux: $ source ./venv/bin/activate

  Windows: $ venv\Scripts\activate

- **Django installieren**

  $ pip install django



2. Erste Schritte mit Django

- **Wie lege ich ein Projekt an?**

  $ django-admin startproject projectname

- **Wie starte ich den Server?**

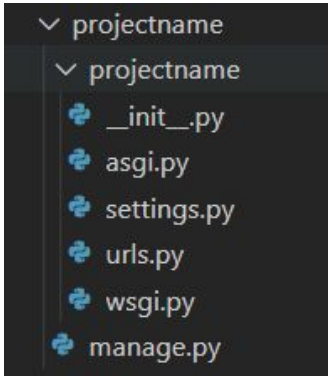
  $ python manage.py runserver

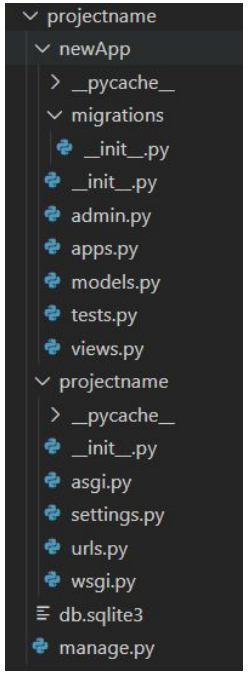
  (Anmerkung: Nach Eingabe der vorher angegeben Schritte, kann an dieser Stelle der Server schon gestartet und ist im Browser unter http://127.0.0.1:8000/ zu erreichen.)

- **Weiter Apps im Projekt erstellen**

  $ python manage.py startapp newApp

Projektstruktur:



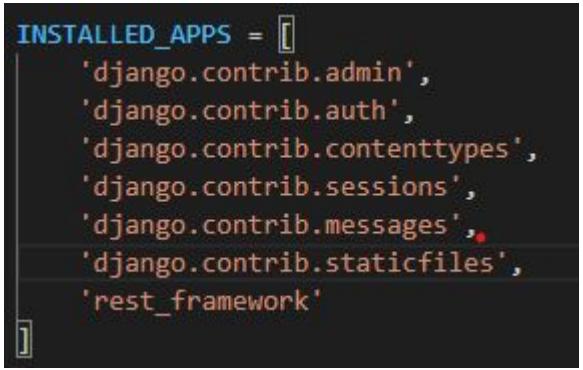


2. Erste Schritte mit Django

- **Wie werden Packages in installiert?**

  $ pip install djangorestframework

- Packages müssen danach noch in settings.py eingetragen werden



2. Django Packages

- **Django Rest Framework**

  - Ist ein Package zum erstellen von Rest APIs

  - große Hilfe beim umsetzen CRUD Operationen

  - darin halten sind:
    - Oauth1 und OAuth2 Protokolle
    - Serialisierung von ORM und non-ORM Daten

2. Django Packages

- **Django CORS Headers**

  - CORS = Cross Origin Resource Sharing

  - Zusätzliche HTTP-Header um Browser mitzuteilen das Webanwendung auf anderer Domain läuft

  - Diesen Headern kann eine Zugriffsberechtigung übergeben

  - Zusätzlich kann außerdem angeben werden welche Methoden erlaubt sind

2. Django Packages

- **Django Debug Toolbar**

  Zeigt die Informationen des aktuellen Request:

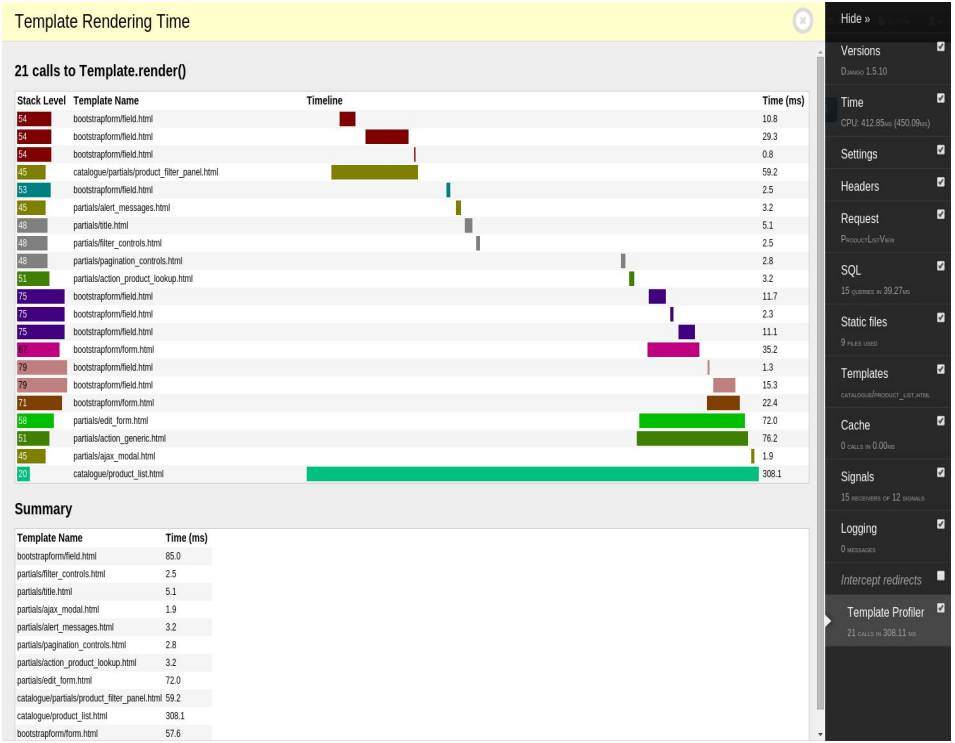
  - Abfragedauer

  - Request Variablen (GET, POST, Cookie und Session informationen)

  - Alle SQL-queries mit diversen Erläuterungen

  - Django version

  - HTTP Headers



2. Django Packages

- **Django Allauth**

  lokale sowie Authentifizierung über Drittanbieter, wie E-Mail Bestätigung, Facebook, Google

  zusätzliche Verwaltung von:
  - Login
  - Logout
  - Passwort ändern
  - Username ändern
  - Account deaktivieren

3. Performance / Optimierung

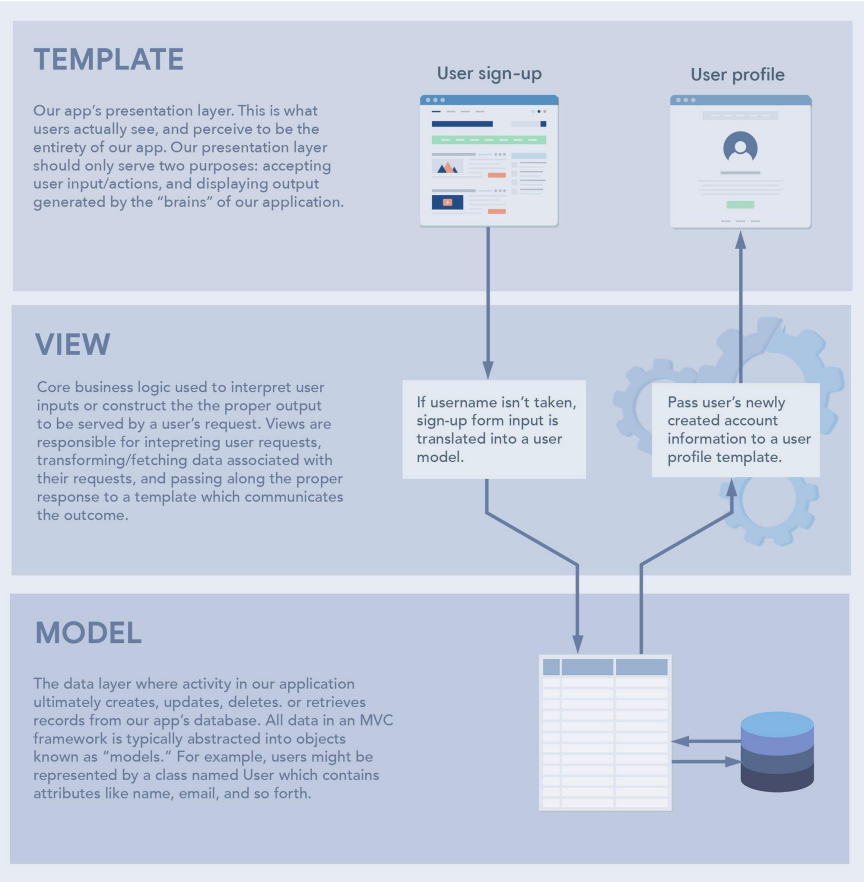
- **Performance von Web Frameworks**
- die Qualität des objektiv wahrgenommenen Verhaltens einer Anwendung
- Ladezeiten, Nutzbarkeit, Interaktivität, Leistungsmessungen

- **Optimierung der Performance**
- Die Anwendung soll so effizient wie möglich genutzt werden können
- Das Verhalten soll dabei nur minimal beeinflusst werden
- wichtige Faktoren - Latenz, Anwendungsgröße, Anzahl DOM-Knoten, Anzahl Requests und wie mit diesen umgegangen wird, CPU-Leistung

3. Performance in Django

- **Model-View-Template in Django**
- sehr ähnlich zu dem bekannten Model-View-Controller Konzept
- lose gekoppelte Komponenten
- Django übernimmt die Aufgabe des Controllers

- Das Template und die View werden an eine URL gebunden und Django liefert die Ausgabe an den Nutzer



Django Model-View-Template Funktionsweise

3. Performance in Django

- **Templates**

- Django Template Language (DTL)
- kann alle text-basierten Formate generieren (HTML, XML, JS, CSV, ...)
- Syntax

Variable - **{{ variable }}**

Filter - **{{ namelower }}**

Tag - **{% tag %}**

Comment - **{#Comment#}**

```
{% extends "base_generic.html" %}

{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Django Template Beispiel geschrieben in DTL

3. Performance in Django

S.NO.	Model View Controller (MVC)	Model View Template (MVT)
1.	MVC has controller that drives both Model and View.	MVT has Views for recieving HTTP request and returning HTTP response.
2.	View tells how the user data will be presented.	Templates are used in MVT for that purpose.
3.	In MVC, we have to write all the control specific code.	Controller part is managed by the framework itself.
4.	Highly coupled	Loosely coupled
5.	Modifications are difficult	Modifications are easy
6.	Suitable for development of large applications but not for small applications.	Suitable both small and large applications.

Unterschiede zwischen dem MVC und MVT Pattern

3. Performance Optimierung in Django

- **Caching** - welche Vorteile hat es ?
- Datenbank Caching, File-System Caching, Lokales Caching
- Lazy loading von Querysets
- QuerySets sind Listen von Objekten, die es uns ermöglichen Daten aus der Datenbank zu lesen, zu filtern und zu sortieren
- Cached Templates (Cache Template Loader)

```
>>> q = Entry.objects.filter(headline__startswith="What")
>>> q = q.filter(pub_date__lte=datetime.date.today())
>>> q = q.exclude(body_text__icontains="food")
>>> print(q)
```

Beispiel Code QuerySets : Zugriff auf die Datenbank findet erst bei der Auswertung des Queries statt

3. Performance Optimierung in Django

- **Code Optimierung**
- restrukturieren
- Profiler verwenden
- Module von Drittanbietern verändern oder ersetzen

```
In [3]: load_ext line_profiler

In [4]: lprun -f get_books_by_library get_books_by_library()
Timer unit: 1e-06 s

Total time: 47.977 s
File: <ipython-input-1-5cb238008f5e>
Function: get_books_by_library at line 1

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     1                                           def get_books_by_library():
     2         1        312.0    312.0      0.0      libraries = Library.objects.all()
     3         1          1.0      1.0      0.0      result = {}
     4      1001      30298.0     30.3      0.1      for library in libraries:
     5      1000     566487.0    566.5      1.2          books_in_library = library.book_set.all()
     6      1000   47379897.0  47379.9     98.8          result[library.id] = list(books_in_library)
     7
     8         1          0.0      0.0      0.0      return result

In [5]:
```

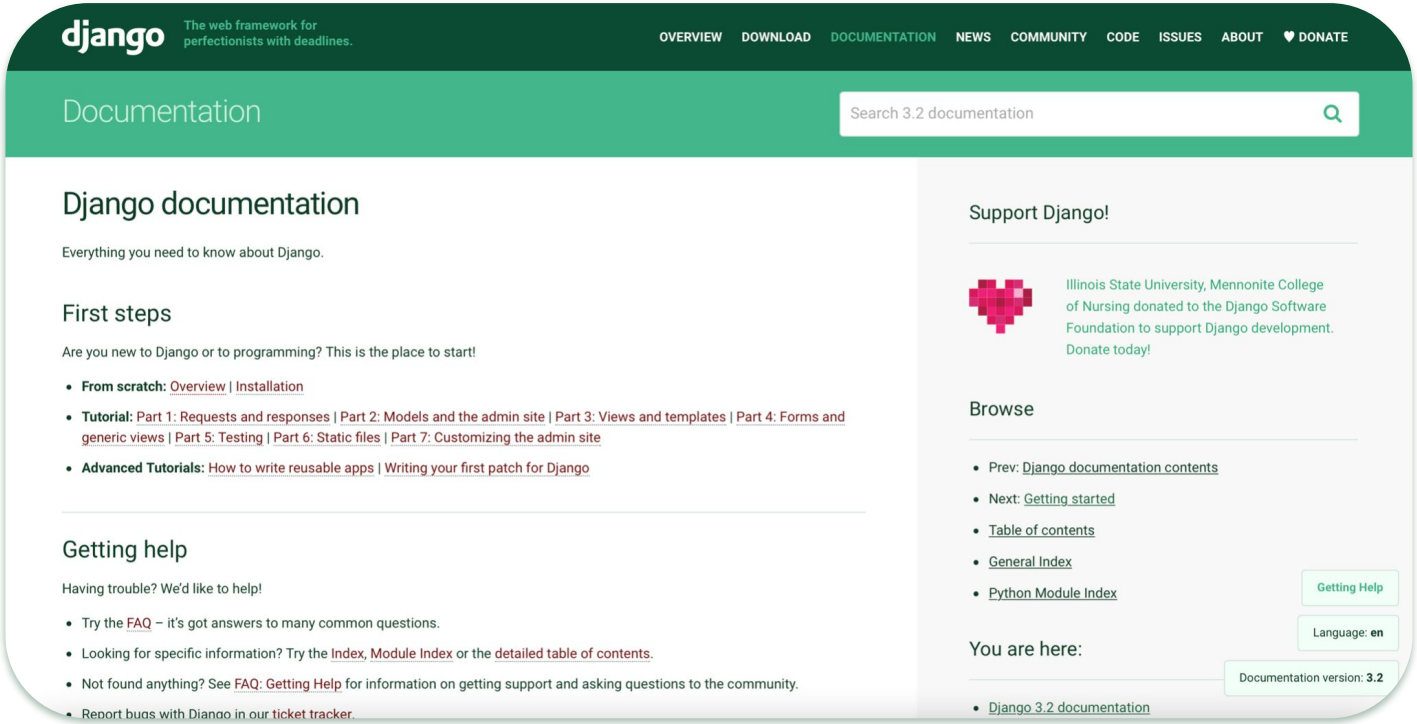
Code Profiler - Übersicht der benötigten Zeit pro Codezeile

3. Dokumentation von Django

- äußerst ausführliche Aufführung der Funktionen und deren Einsatzbereiche
- sehr große Community
- aktueller Stand
- Step-By-Step Tutorials für neue Python/Django Programmierer
- Tipps zur Optimierung der Performance

- **Organisation der Dokumentation**
- Tutorials
- Topic guides
- Reference guides
- How-to-guides

3. Dokumentation von Django



https://docs.djangoproject.com/en/3.2/

4. Vergleichbare Web Frameworks

- Teil des Industriestandard
- benötigt Jinja2 und Werkzeug
- stellt keine Komponenten zur Verfügung für die Lösungen existieren
- entsprechende Bibliotheken lassen sich leicht integrieren
- mitgelieferter Webserver

+	--
Minimalistisch ohne einbußen	Nicht Asynchron-freundlich
Viele Ressourcen online Verfügbar	Um ein größeres Projekt umzusetzen sind Vorkenntnisse erforderlich
Extrem leicht einen schnellen Prototypen zu erstellen	Threadlocals und Globals werden überall verwendet

- keine Abhängigkeiten
- minimalistisch und leicht zu nutzen
- mitgelieferter Webserver
- Teil der Basisbibliothek von Python
- Besser für kleinere Projekte geeignet

+

Flexibel

Braucht keine Installation

Asynchron-freundlich

--

kleine Community

Sehr schwer Projekte mit mehr als 1000 Zeilen Code zu entwickeln

- Full-Stack-Framework
- Multi-Datenbank unterstützung
- Horizontale Daten Partitionierung
- Javascript Toolkit-unterstützung
- Unterstützung von vielen Datenaustausch Formaten

+

Fängt als Microframework an und endet als Fullstack Lösung

Unterstützt horizontale Daten Partionierung

--

Die Große Erweiterbarkeit kann überwältigen

WEB2PY

- Fokus auf Datensicherheit
- keine Abhängigkeiten
- Vollständiges Framework
- sehr steile Lernkurve
- Abwärtskompatibilität
- integrierte Web-Oberfläche

+

Dokumentation in Form eines Buches geschrieben

User Unterstützung

einfach zu erweitern

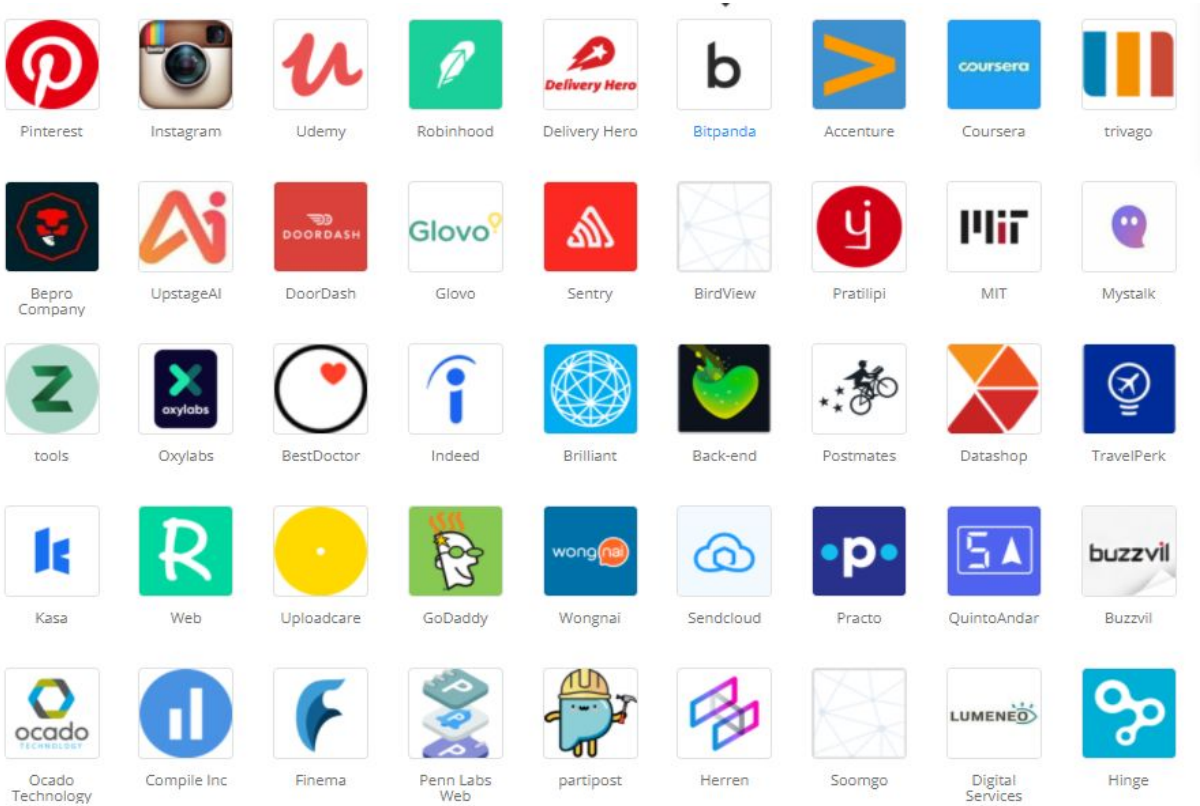
--

Die Web IDE ist keine vollständige IDE

- verlässlicher WSGI threadpooled webserver
- flexibles Plugin System
- Eingebaute Werkzeuge für Datensicherheit
- kann mehrere http Server auf mehreren Ports gleichzeitig in Betrieb nehmen
- eingebaute profiling, coverage und Test Unterstützung

+	--
Robuster Konfigurations-Mechanismen	keine gute Dokumentation
Hat Produktionsfertigen Server	
Hilft dir deinen Code zu strukturieren	

5. Welche Unternehmen nutzen Django?



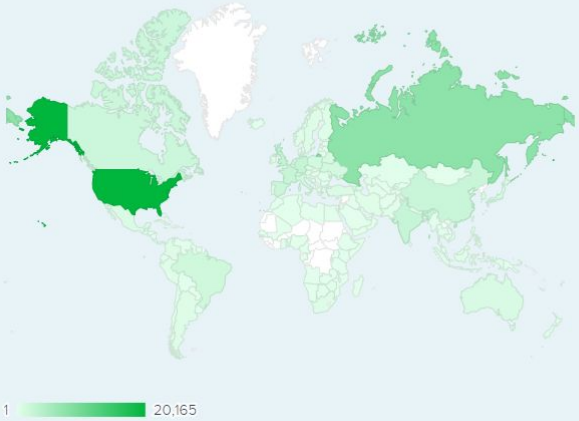
Stackshare.io

Daten und Fakten

- 94,907 Webseiten laut SimilarTech

Geography

Django usage by websites across the globe

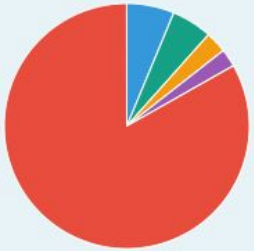


Leading Countries

Country	Websites
United States	20,165
Russia	7,642
United Kingdom	2,908
Germany	2,641
France	2,594
India	2,536
China	2,384
Spain	2,227
Canada	2,002
Brazil	1,802
Rest of the World	26,832

Top Industry Verticals

Industry verticals where Django is being used



Industry Vertical	Popularity
Science & Education	6.27%
Computers Electronics & T...	5.35%
Food & Drink	2.81%
Arts & Entertainment	2.37%
Others	83.20%

Die wohl bekanntesten Namen die Django nutzen:

Disqus (Gründung 2007)

ist ein comment hosting Service

- genutzt von 75% der Webseiten, welche ein Drittanbieterprogramm nutzten (stand 2011)

- ca. 1,7 Million Kommentare jeden Tag
- ca. 66 Millionen tägliche Nutzer
- ca. 15 Milliarden Seitenaufrufe pro Monat

Bekannte Seiten die Disqus nutzen:

- genius.com
- worldstar.com
- myshopify.com

Dropbox (Gründung 2007)

ist ein file hosting Service

- von ca 25.000 Webseiten genutzt
- 700 Millionen registrierte Nutzer

Nach GoogleDrive, Apple iCloud und OneDrive meist genutzer file hosting Service

- beinhaltet 400 Milliarden hochgeladene Datensätze (stand 2018)

Bekannte Seiten die Dropbox nutzen:

Instagram (Gründung 2010)

- ca. 95 Million Fotos jeden Tag
- ca. 4 Milliarden Likes jeden Tag
- ca. 300 Millionen tägliche Nutzer

Stand 2016

- Instagram bietet derzeit die größte Umsetzung einer Anwendung mit Django

- “We had been able to get to a few hundred million users with our Python/Django stack, so we decided we would continue. Also significant in the decision was that our engineers love Python. It’s a reason people want to come to work for us.”

  - **Hui Ding - Former Head of Instagram**

Instagram stack

6. Informationsquellen

- https://flask.palletsprojects.com/en/2.0.x/
- https://bottlepy.org/docs/dev/
- https://turbogears.org
- http://www.web2py.com
- https://cherrypy.org
- https://towardsdatascience.com/5-cool-alternatives-to-django-and-flask-for-deploying-endpoints-5c99a066696
- https://docs.djangoproject.com/en/3.2/
- https://docs.djangoproject.com/en/3.2/topics/performance/
- https://www.geeksforgeeks.org/top-10-django-apps-and-why-companies-are-using-it/
- https://www.quora.com/How-does-Instagram-use-Django
- https://data-flair.training/blogs/django-features/
- https://data-flair.training/blogs/django-caching/
- https://medium.com/shecodeafrica/understanding-the-mvc-pattern-in-django-edda05b9f43f
- https://developer.mozilla.org/en-US/docs/Learn/Performance/What_is_web_performance
- https://www.brandwatch.com/de/blog/instagram-statistiken/
- https://www.tutorialspoint.com/django/django_overview.html
- https://www.slant.co
- https://www.geeksforgeeks.org/top-10-django-apps-and-why-companies-are-using-it/
- https://allfacebook.de/instagram/instagram-nutzer-deutschland
- https://www.similartech.com/technologies/
- https://backlinko.com/dropbox-users
- https://expandedramblings.com/index.php/dropbox-statistics/
- https://www.youtube.com/watch?v=q_fzdjiLbPQ&list=PLyGqLFNKX_QnzcHneNFM-FeAjq-4FXwGo&index=3
- https://en.wikipedia.org/wiki/Django_(web_framework)
- https://de.wikipedia.org/wiki/Framework#cite_note-1
- https://www.ionos.de/digitalguide/websites/web-entwicklung/server-side-und-client-side-scripting-die-unterschiede/
- https://www.youtube.com/watch?v=V1aQyoRy91k
- https://pediaa.com/difference-between-client-side-scripting-and-server-side-scripting/

Bildquellen

- https://de.wikipedia.org/wiki/Datei:Instagram_logo_2016.svg
- https://hackersandslackers.com/creating-django-views/
- https://www.similartech.com/technologies/django
- https://stackshare.io/django
- https://docs.djangoproject.com/en/3.2/
- https://raw.githubusercontent.com/node13h/django-debug-toolbar-template-profiler/master/screenshot.png
- https://www.tabel-gmbh.de/haus-bauen/vorteil-holzbau/
- https://en.wikipedia.org/wiki/File:Django_Reinhardt_(Gottlieb_07301).jpg
- https://www.techwebspace.com/server-side-vs-client-side-programming-languages/

Noch Fragen?