Das Django Web-Framework dargestellt anhand des Praktischen Beispieles eines Inventarisierungssystemes

Clemens Dautermann

2. Januar 2019 bis 12. Januar 2019

Inhaltsverzeichnis

1 Einleitung

Diese Facharbeit soll einen grundlegenden Überblick über die wichtigsten Funktionen des Django Web-Frameworks geben.
Das Django Web-Framework ist ein größtenteils in Python geschriebenes[1] Framework zum entwickeln von Webservern. Es ist aufgrund seiner ausgeprägten Modularität und der Existenz einer Vielzahl von Datenbanktreibern besonders gut für die Entwicklung von Webservern geeignet, die eine Datenbank erfordern.
Django stellt eine grundlegende Struktur für die Entwicklung zur Verfügung. So zum Beispiel:

- Eine settings.py Die genutzt werden kann um Konfigurationsmöglichkeiten zentral zu bündeln
- Eine library um einfache Zugriffe auf Datenbanken zu tätigen und sogenannte Models um Datenbankobjekte zu verwalten
- Ein Routingsystem um eine einfachere Verwaltung von Urls zu gewährleisten
- Eine Grundstruktur, die Modularität unterstützt und das einfache Installieren oder Entfernen von sogenannten ”Apps” ermöglicht

Es ist also kaum notwendig, jedoch durchaus möglich, als Entwickler noch SQL zu schreiben wenn man mit dem Django Web-Framework entwickelt.

2 Struktur

Ein typischer Django Server ist aus sogenannten ”Apps” aufgebaut. Diese werden entweder vom Entwickler selber geschrieben oder können via pip (dem Python Paket Manager) installiert werden. Ein standard Verzeichnisaufbau ist in Abbildung 2 dargestellt.

2.1 Erstellung

Ein Django Projekt kann mit dem Befehl $ django-admin startproject server initialisiert werden. Dadurch wird folgende Ordnerstruktur erstellt:

```
server
├── manage.py
└── server
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

Abbildung 1: Verzeichnisstruktur, die der $ django-admin startproject server Befehl erzeugt

[1] Offizielle Django GitHub Seite https://github.com/django/django

2.2 manage.py

Die manage.py wird, wie der Name schon sagt, verwendet um den Server zu verwalten. Mit Hilfe der manage.py können beispielsweise Migrierungen an der Datenbank erstellt werden, Datenbanknutzer erstellt werden oder der Testserver zur Entwicklung kann gestartet werden. Die gleiche Funktionalität stellt auch der django-admin Befehl zur Verfügung[2].

2.3 db.sqlite3

In dieser Datei wird die SQL Datenbank gespeichert, die der Server nutzt. Sie wird automatisch erstellt. Es können jedoch auch andere Datenbanken, wie zum Beispiel MongoDB oder eine extern gehostete Datenbank, verwendet werden.

2.4 server/server

2.5 __init__.py

Diese Datei befindet sich im Wurzelverzeichnis jeder App. Sie macht für Python erkennbar, dass es sich bei dem Inhalt dieses Ordners um ein Python Modul handelt. Somit kann die App einfach geteilt und von anderen Nutzern verwendet werden.

2.6 settings.py

```
server
├── manage.py
├── db.sqlite3
├── server
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── app1
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── forms.py
    ├── models.py
    ├── tests.py
    ├── urls.py
    ├── views.py
    └── migrations
        └── 0001_initial.py
```

Abbildung 2: Die typische Verzeichnisstruktur eines Django Servers

[2]Django Dokumentation https://docs.djangoproject.com/en/2.1/ref/django-admin/