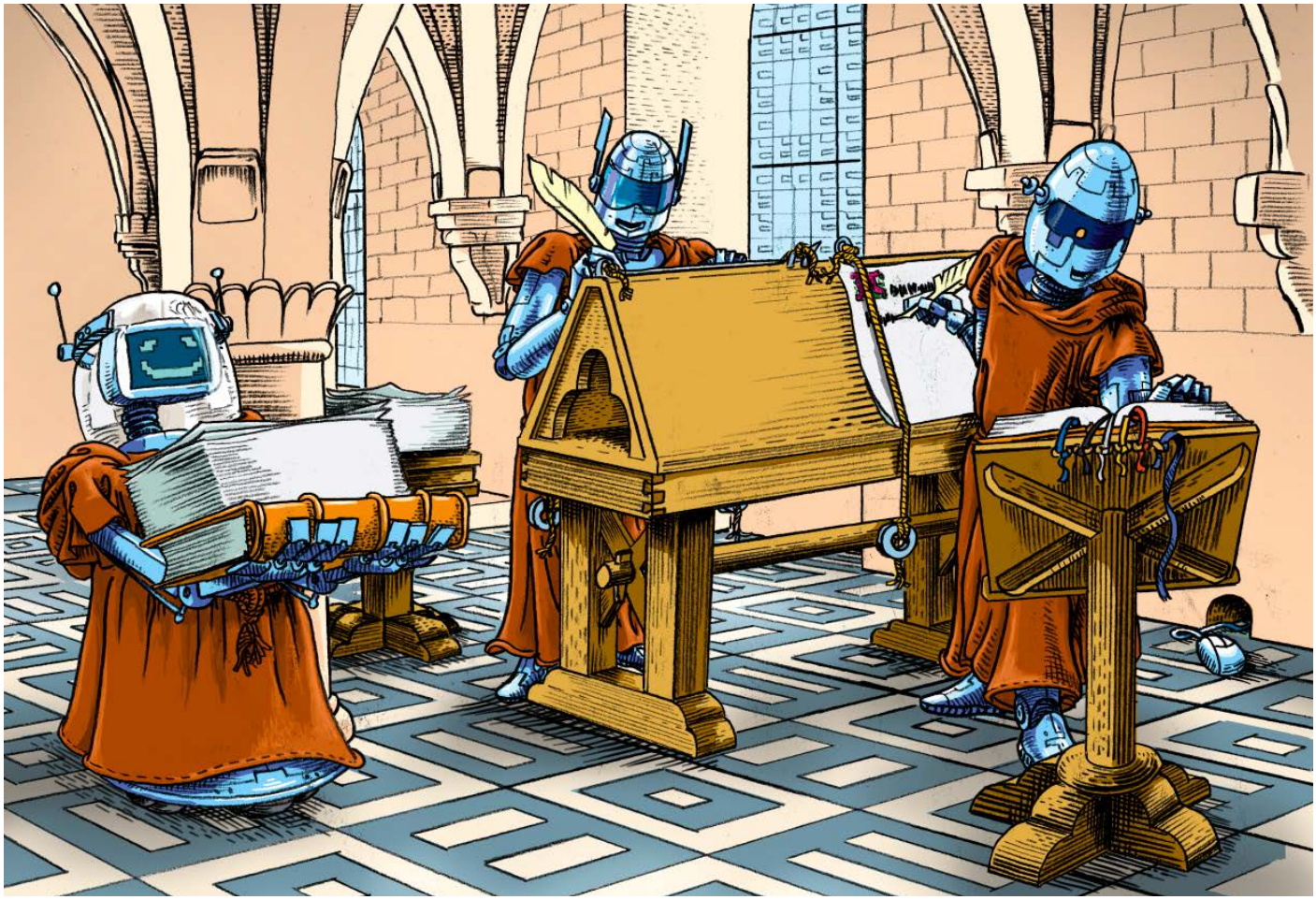


Bild: Rudolf A. Blaha

Doku-Freuden

Ansprechende Online-Dokumentationen mit MkDocs

MkDocs unterstützt Softwareentwickler und Administratoren beim Dokumentieren und kümmert sich darum, dass die Inhalte ansprechend präsentiert werden. Das erhöht die Motivation, Inhalte zu verfassen – und sie zu lesen.

Von Jan Mahn

Eine gute Dokumentation ist mehr als nur lästiges Beiwerk. Administratoren sollten das Wissen über ihr Netzwerk für Kollegen und die Nachwelt festhalten und Softwareentwickler tun ihren Kunden einen großen Gefallen, wenn sie – am besten auf einer stets aktualisierten Website – eine ausführliche Dokumentation anbieten. Und selbst fürs Heimnetz, fürs Smart Home und die Bastelprojekte lohnt ein solches Handbuch.

Schön auf Knopfdruck

Die Software MkDocs gehört zur Gruppe der statischen Seitengeneratoren – ein Schritt zurück zu den Ursprüngen des World Wide Webs. Autoren verfassen die Texte in einer leicht zu lernenden Auszeichnungssprache, in diesem Fall Markdown. MkDocs rendert aus diesen Texten und einer Vorlage fertige HTML-Seiten mit Menü, die man auf einem Webserver ausliefert.

MkDocs-Dokumentationen begegnet man recht häufig, sowohl bei der Dokumentation von Open-Source-, als auch bei kommerzieller Software. Nicht immer erkennt man sie auf den ersten Blick – je nach verwendetem Theme sehen die Ergebnisse sehr unterschiedlich aus. Über ct.de/ygyg finden Sie einige Beispiele von größeren Softwareprojekten. Viele nutzen das Material-Theme, das Googles Designsprache Material umsetzt. Unter den MkDocs-Themes ist es das mit Abstand ausgereifteste und sehr gut personalisierbar.

Einrichten

Ein eigenes MkDocs-Projekt ist schnell aufgesetzt und mit wenig Arbeit wird aus einem Markdown-Dokument eine hübsche Dokumentation im Browser. Ein Vorteil von statischen Seitengeneratoren ist es, dass man das Ausgangsmaterial, also Texte und Bilder, mit einer Versionsverwaltung wie Git genau wie Programmcode verwalten und versionieren kann – anders als zum Beispiel die Daten einer WordPress-Installation, die zur Hälfte in einer SQL-Datenbank und zur anderen Hälfte in Daten-Ordnern stecken.

Am besten erzeugen Sie schon für die ersten Versuche mit MkDocs ein Git-Repository – als Vorlage können Sie das Beispiel-Repository zu diesem Artikel auschecken (siehe ct.de/ygyg).

Damit MkDocs eine Dokumentation rendern und ein Menü erzeugen kann, braucht der Generator zunächst eine Konfigurationsdatei im YAML-Format mit dem Namen mkdocs.yml, die zum Beispiel so aussieht:

```
site_name: Beispiel-Doku
nav:
 - index.md
 - getting_started.md
 - legal/imprint.md
theme: material
```

Sie muss mindestens den `site_name`, das `theme` und einen Eintrag unter `nav` enthalten. Im Ordner, in dem die Konfiguration liegt, brauchen Sie einen Ordner mit dem Namen docs. In diesem landen alle Markdown-Dokumente mit Ihren Inhalten – bei der Gestaltung der Ordnerstruktur unterhalb von docs sind Sie recht frei. Die Ordner- und Dateinamen (ohne die Endung .md) werden beim Generieren zur URL der Seite verarbeitet. Das Impressum, das aus der Datei docs/legal/imprint.md entsteht, findet man später etwa unter „https://example.org/legal/imprint“. Eine Markdown-Datei muss nicht zwangsläufig in der Navigation eingebaut sein, damit MkDocs sie rendert. Alle Dateien im Ordner docs werden verarbeitet.

Legen Sie jetzt die Markdown-Datei index.md an und füllen Sie diese mit etwas Inhalt, etwa:

```
# Willkommen
Dies ist eine Dokumentation.
![Ein Screenshot](screen.png)
Zum [Impressum](legal/imprint.md).
```

In der letzten Zeile sehen Sie einen internen Link, der beim Generieren automatisch zu einem vollständigen HTML-Link umgewandelt wird. MkDocs ist in der Lage, den Titel einer Seite anhand der ersten Überschrift erster Ebene (in Markdown mit # eingeleitet) auszulesen. In der Navigation erscheint die Seite also als „Willkommen“. Möchte man diesen Titel im Menü überschreiben, ergänzt man den alternativen Titel in der mkdocs.yml:

```
nav:
 - "Herzlich Willkommen": index.md
```

Einsetzen

MkDocs ist eine Python-Anwendung, vom Python-Code sieht man als Anwender aber nichts, wenn man nicht gerade selbst Plugins entwickeln möchte. Wer sich in der Python-Welt zu Hause fühlt und sowohl Python als auch den Paketmanager Pip installiert hat, kann die Software als Paket beziehen:

```
pip install mkdocs
```

Wenn Sie, wie im Beispiel, das beliebte Material-Theme nutzen möchten, laden Sie auch dieses über pip:

```
pip install mkdocs-material
```

Anschließend navigieren Sie auf der Kommandozeile in den Ordner, in dem die Datei mkdocs.yml liegt und starten den Entwicklungsserver:

```
mkdocs serve
```

Nach etwa einer Minute erreichen Sie die Dokumentation unter der Adresse http://localhost:8000 im Browser. Der Entwicklungsserver bemerkt, wenn Sie eine Änderung vorgenommen haben und lädt automatisch die Seite neu. So kann man bequem die Dokumentation verfassen und die Ergebnisse fast in Echtzeit verfolgen.

Der Entwicklungsserver ist aber nicht dafür gedacht, die Seite auszuliefern. Stattdessen soll die Python-Anwendungen einen Ordner mit fertigen HTML-Dokumenten erzeugen, die man dann mit einem Webserver ausliefern kann. Den Generator starten Sie im Projektordner mit:

```
mkdocs build
```

Das Ergebnis landet im Ordner site. Wenn Sie ein klassisches Webhosting-Paket gebucht haben, auf das Sie zum Beispiel per SFTP Ihre Inhalte kopieren, können Sie diesen Ordner dort per Hand (auf der Kommandozeile oder mit Werkzeugen wie FileZilla) ablegen oder ein Skript schreiben, das die Doku lokal baut und dort hochlädt.

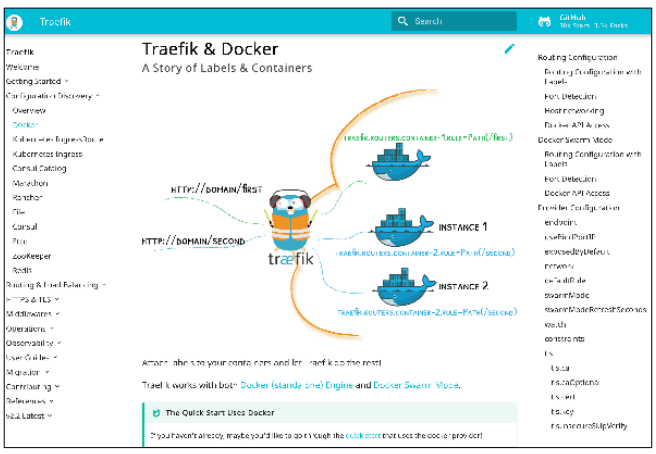
Eine Skriptsprache wie PHP, Python oder Perl ist auf dem Webserver nicht nötig – sogar die eingebaute Suche kommt ganz ohne aus. Beim Bauvorgang erzeugt MkDocs einen Suchindex und löst die Suche Client-seitig per JavaScript. Auch wenn Sie nur eine Seite geändert haben, sollten Sie daher immer den ganzen Ordner site auf den Server kopieren.

Standard aufbohren

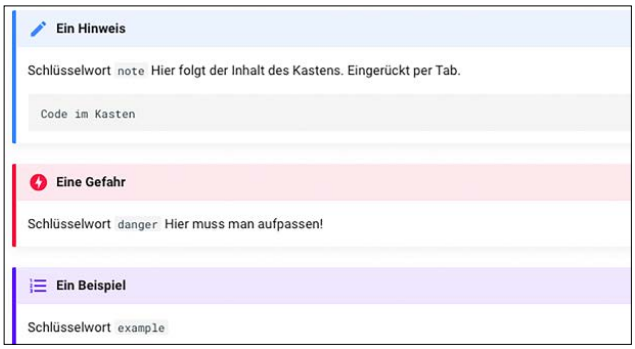
Bilder, Links, Tabellen und Überschriften und Listen und Listing-Abschnitte für Programmcode erstellen Sie mit handelsüblichem Markdown. Eine kleine Übersicht der gängigsten Formatierungen:

````
# Überschrift erster Ebene
## Überschrift 2. Ebene
```
function someCode(){
}
```
* Eine Aufzählung
* mit zwei Punkten
Der Text ist *kursiv*, dieser **fett**
|Kopfzeile|Kopfzeile|
|---|---|
|Zelle|Zelle|
````

Darüber hinaus ist es hilfreich, Markdown mit sogenannten Extensions zu erweitern. In Dokumentationen kommt es recht häufig vor, dass man Abschnitte besonders hervorheben möchte – etwa eindringliche Warnungen oder gut gemeinte Hinweise.



MkDocs ist bei großen Softwareprojekten im Einsatz. Die Dokumentation des HTTP-Routers Træfik zeigt, was mit der Software möglich ist.



Mit der Markdown-Erweiterung Admonition erzeugen Sie Textkästen, um Inhalte vom Fließtext abzuheben.

Mit der Erweiterung `admonition` erzeugen Sie Kästen, die im Fließtext auffallen. Die Erweiterung `superfences` kümmert sich darum, dass Sie in diesen Kästen Markdown-Formatierungen nutzen können. `details` macht die Kästen auf Wunsch ein- und ausklappbar. Die nötigen Pakete werden bereits mit dem Material-Paket heruntergeladen. Sie aktivieren die Erweiterungen durch eine kleine Ergänzung in der Datei mkdocs.yml:

```
markdown_extensions:
 - admonition
 - pymdownx.superfences
 - pymdownx.details
```

Sind die Erweiterungen aktiviert, erzeugen Sie einen Kasten in einer Markdown-Datei mit:

```
!!! note „Ein Hinweis“
 Hier folgt der Inhalt des Kastens.
 Eingerückt per Tab.
 ```
 Code im Kasten
 ```
```

Oben sehen Sie verschiedene Textkästen in ihrer ganzen Schönheit. Neben dem Schlüsselwort `note` für harmlose Hinweise unterstützt die Erweiterung auch `abstract`, `info`, `tip`, `success`, `question`, `warning`, `failure`, `danger`, `bug` und `quote`. Damit sind zum Beispiel auch FAQ-Bereiche oder Schulungsunterlagen für Mitarbeiter mit MkDocs kein Problem. Soll der Kasten ein- und ausklappbar sein, ersetzen Sie die `!!!` durch `???`.

Aller Wahrscheinlichkeit nach werden Sie mit den eingestellten Farben für Links und die Kopfleiste noch nicht zufrieden sein – als Unternehmen muss die Dokumentation zum Corporate Design passen. Dafür müssen Sie aber nicht von Grund auf ein neues Template entwickeln. Das Material-Design können Sie mit zusätzlichem CSS erweitern. Ersetzen Sie in der mkdocs.yml zunächst die Zeile `theme: material` durch folgenden Block:

```
theme:
 name: material
 language: de
 font: false
 favicon: favicon.png
 logo: logo.png
```

Mit `language: de` zeigt MkDocs die Navigationselemente auf Deutsch an. `font: false` verhindert das Einbinden von externen Schriftarten direkt von Google Fonts – der Datenschutzbeauftragte wird es Ihnen danken. Auch das Logo und das Favicon können Sie hier konfigurieren. Die Bilddateien gehören wie die Markdown-Texte in den Ordner docs oder einen Unterordner. Die vollständige Dokumentation des Material-Themes mit weiteren Optionen finden Sie über ct.de/ygyg.

Mit `extra_css` binden Sie eine zusätzliche CSS-Datei ein. In dieser können Sie mit sogenannten Custom Properties die Farben an Ihre Wünsche anpassen, zum Beispiel für eine rote Kopfleiste mit grauem Text:

```
:root {
 --md-primary-fg-color: #B51F10;
 --md-primary-fg-color--light: ...;
 --md-primary-fg-color--dark: ...;
 --md-primary-bg-color: #ccc;
 --md-primary-bg-color--light: #eee;
}
```

Die gesamte CSS-Datei finden Sie im Beispielprojekt über ct.de/ygyg.

Einfacher verpacken

Wer kein Python installiert hat, mit mehreren Entwicklern zusammen an einer Doku arbeiten möchte oder möglichst viel automatisieren will, kann sich mit Docker-Containern das Leben leichter machen – sowohl für die Entwicklung als auch für die Auslieferung der Seite [1]. Steckt man die gesamte Doku in einen Container, kann man sicher sein, dass die Software auf allen Entwickler-Maschinen und auf dem Server identische Bedingungen vorfindet – identische Abhängigkeiten, MkDocs-Plug-ins und Python-Versionen zum Beispiel.

Für Entwicklung und Auslieferung brauchen Sie zwei unterschiedliche Dockerfiles. In der Entwicklungsumgebung soll das Neuladen bei Änderungen weiter funktionieren. Folgendes Dockerfile richtet eine Arbeitsumgebung im Container ein. Alle Dateien finden Sie im Beispielprojekt zum Artikel bei GitHub. Die Datei bekommt den Namen Dockerfile-dev:

```
FROM python:3-alpine
RUN apk add build-base
COPY ./mkdocs/ /mkdocs/
WORKDIR /mkdocs/
RUN pip install -
-upgrade pip && ↴
↳pip install mkdocs mkdocs-material
EXPOSE 8080
CMD ["mkdocs", "serve"]
```

Damit die Zusammenstellung funktioniert, müssen Sie das Repository etwas umstrukturieren: Auf der obersten Eben legen Sie den Ordner container an. In diesen kommt das oben stehende Dockerfile mit dem Dateinamen Dockerfile-dev. Außerdem ein Ordner mkdocs, der die oben angelegte Datei mkdocs.yml sowie den Ordner docs mit allen Inhalten enthält.

Um den Container für Entwicklungszwecke zu starten, sollten Sie sich eine kleine Docker-Compose-Datei mit dem Namen compose-dev.yml auf der obersten Ebene anlegen:

```
version: "3.7"
services:
 docs:
 build:
 context: ./container
 dockerfile: Dockerfile-dev
 ports:
  - 8080:8080
 volumes:
  - ./container/mkdocs/:/mkdocs
```

Navigieren Sie auf der Kommandozeile in den Projektordner und fahren Sie die Entwicklungsumgebung hoch:

```
docker-compose -f \
docker-compose-dev.yml up
```