Statische Blogs mit Pelican erstellen

Axel Kielhorn

Betriebssystemwerkzeuge

Diese Werkzeuge sind nicht zwingend erforderlich, erleichtern aber die Arbeit durch Automatisierung.

Mac OS

- `X-Code` mit Kommandozeilenwerkzeugen für `make` (Für Pelican 4 nicht mehr erforderlich)

Linux

- `make` (Für Pelican 4 nicht mehr erforderlich)

Windows

- `msys` eine komplette GNU Umgebung
- `GOW` Gnu on Windows, eine minimale GNU Umgebung
- `git` für Windows, das bringt eine `bash` mit und macht das Leben so einfacher. (Funktioniert mit GOW zusammen.)
- Für Pelican 4 nicht mehr erforderlich.

Python installieren

Download von Python.org. `Pelican` funktioniert mit Python 3.6 oder höher. Für eine neue Installation sollte das aktuelle Python (3.9) verwendet werden.

Alternative Anaconda, ist aber für ein Blog Overkill. (Siehe Anaconda Anleitung.) Evtl. stattdessen `Miniconda`.

Zusätzliche Module

- `pip` Bei neueren Python Versionen bereits enthalten.
- `virtualenv` Eine Virtualisierungsumgebung für Python. Alles weitere wird in der Virtualenv installiert. (`venv` bei Anaconda) Ab Python 3.5 wird `venv` empfohlen, es ist Bestandteil der Python Installation.
- `pelican` Das Blogsystem.
- `markdown` Die Blogs werden in Markdown geschrieben.
- `typogrify` Typografische Verbesserungen. (Empfehlenswert, aber nicht zwingend erforderlich.)
- `fabric` (optional) Eine Make Alternative. Für Pelican 3.
- `invoke` (optional) für Pelican 4
- `livereload` zum automatischen aktualisieren im lokalen Webserver.
- `ghp-import` (optional) Importiert die Ausgabe in ein Github Repository.
- `s3cmd` (optional) zur Unterstützung von Amazon S3.
- `webassets` und `cssmin` werden von einigen plugins benötigt, nur bei Bedarf installieren.

Virtualenv installieren

Mit `python -m pip install virtualenv` wird die Unterstützung für virtuelle Umgebungen installiert.

`python -m virtualenv ENV` erstellt eine neue virtuelle Umgebung im Verzeichnis `ENV`. Dieses darf danach nicht mehr umbenannt oder verschoben werden.

`python -m venv ENV` bietet die gleiche Funktion ab Python 3.5.

Die Umgebung wird mit `source bin/activate` aktiviert und mit `deactivate` deaktiviert.

Unter Windows mit `command.com` wird die Umgebung mit `Scripts\activate.bat` aktiviert. Bei Verwendung der `bash` aus dem `git` Paket mit `source Scripts\activate`. Der Pfad zur virtuellen Umgebung darf keine Leerzeichen enthalten.

Pelican installieren

Zuerst wird die virtuelle Umgebung aktiviert. Dann wird in dieser Umgebung das Blogsystem mit `pip install pelican markdown typogrify` installiert.

Wenn auf dem Zielsystem kein `make` zur Verfügung steht, kann man stattdessen `fabric` (Pelican 3) bzw. `invoke` (Pelican 4) installieren. Das funktioniert ähnlich wir `make` benötigt aber nur eine Python Installation und die ist ja für `pelican` bereits vorhanden.

In Thonny

- `Pelican` und `markdown` Modul installieren
- Verzeichnis auswählen
- `Extra System Terminal öffnen`
- `pelican-quickstart` eingeben und Fragen beantworten

Weitere Software (optinal)

- `git` Versionsverwaltung für `ghp-import`
- `Vim` als Editor
  - `pathogen` als Modulverwaltung
  - `vim-pandoc` und `vim-pandoc-syntax` zur Markdown Unterstützung
- `pandoc` um Markdown in andere Formate zu konvertieren (`odt`, `docx`, `latex`)

Updates

Python benutzt das Programm `pip` zur Paketverwaltung. Diese wird mit

```
pip install --upgrade pip
```

aktualisiert. Der Befehl

```
pip list -o
```

listet alle veralteten Pakete auf, diese können dann mit

```
pip install <Paketname> --upgrade
```

aktualisiert werden.

Arbeiten mit GitHub

Es ist eine gute Idee, Dokumente in eine Versionsverwaltung einzuchecken, so lassen sich alte Version und Veränderungen leicht wieder rekonstruieren.

GitHub ist nur ein Anbieter, die hier wiedergegebene Anleitung sollte sich aber leicht für andere Anbieter anpassen lassen. Seit 2020 sind auch die privaten Repositories by `github` kostenlos. Neben `gihub` gibt es noch `gitlab` und `bitbucket`, die ebenfalls in der Grundversion kostenlos sind..

Um mehrere Projekte von einem Rechner betreuen zu können, werden projektabhängige Konfigurationen benutzt.

In der Datei `~/.ssh/config` ist ein Eintrag in der Form

```
Host github-projekt
  User git
  HostName github.com
  IdentityFile ~/.ssh/id-projekt_rsa
```

erforderlich. Dieser legt für das Projekt einen projektspezifischen Key fest. Dieser Schlüssel wird mit dem Befehl

```
ssh-keygen -t rsa -b 2048 -C "name@domain.de" -f id-projekt_rsa
```

erzeugt. Beim Erstellen wird nach einer "passphrase" gefragt, diese bleibt erst mal leer.

Anschließend muss der Schlüssel dem `ssh-agent` mitgeteilt werden:

```
ssh-add -K ~/.ssh/id-projekt_rsa
```

Damit ist der lokale Teil beendet. Der Schlüssel `id-projekt_rsa.pub` wird jetzt noch der remote Seite (GitHub) mitgeteilt. Unter `Personel settings` wird im Unterpunkt `SSH and GPG keys` über den Knopf `New SSH key` ein Dialog geöffnet, in dem der Inhalt der `pub` Datei eingefügt wird.

Nun kann in der Web-Oberfläche bei `GitHub` ein neues Repository `src` angelegt werden. Dieses wird dann mit

```
git clone git@github-projekt:githubuser/src.git
```

auf den lokalen Rechner kopiert.

Anschließend müssen in der Datei `.git/config` noch Name und E-Mail Adresse ergänzt werden:

```
[user]
        name = Name
        email = Name@domain.de
```

Erste Schritte

`pelican-quickstart` erstellt interaktiv eine Basisinstallation.

Neue Artikel entstehen in `content` und sollten folgende Metadaten haben:

```
Title: Mein Super Titel
Date: 2010-12-03 10:20
Modified: 2010-12-05 19:30
Category: Python
Tags: pelican, publishing
Slug: super-artikel
Authors: Mein Name, Meine Muse
Summary: Ist mein Blog nicht Super?
```

Mit dem Befehl

```
pelican -o output content
```

wird die Seite erzeugt.

Das ganze lässt sich mit `invoke build` automatisieren.

Im Verzeichnis `output` kann dann der in Python integrierte Webserver aufgerufen werden. Einfacher geht es mit `invoke serve`.

Im Verzeichnis `content` können Unterverzeichnisse angelegt werden, diese werden auf der Webseite als unterschiedliche Kategorien angezeigt. Eine besondere Rolle spielt hier die Verzeichnisse `pages` und `images`. Diese werden nicht in den normalen Blog Verlauf integriert, sie können für statische Inhalte (z. B. für Impressum, Kontaktdaten, Bilder) verwendet werden.

Die Ordnernamen lassen sich für deutsche Nutzer anpassen.

Publizieren

Die gesamte Webseite kann in einer Dropbox erstellt werden, das `output` Verzeichnis wird dann freigegeben und über eine Domainumleitung (z. B. nic.de.vu, Droppages, freenom oder Site44) freigegeben.

Seit Sommer 2016 funktioniert `nic.de.vu` nicht mehr,

Seit Herbst 2016 liefert Dropbox keine HTML Dateien mehr aus. Damit funktioniert die oben beschriebene Methode nicht mehr.

Für weniger als 5 € gibt es bei allen Webhostern ein Einstiegspacket, in der Regel mit einer eigenen Domain-Adresse.

Alternativ kann das Projekt, oder nur der Ausgabeordner auch auf Github abgelegt werden. Hier hilft `ghp-import`, das die Seiten automatisch in ein github up Repository überträgt.

```
$ pelican content -o output -s pelicanconf.py
$ ghp-import output
$ git push origin gh-pages
```

Das `Makefile` bietet das Target `github` zum Automatisieren des Exports.

Anpassungen

Spracheinstellungen

Standardmäßig werden englische Ordnernamen verwendet. Diese lassen sich aber einfach in der `pelicaconf.py` umstellen:

```
DEFAULT_LANG = u'de'
DEFAULT_CATEGORY = 'Verschiedenes'
STATIC_PATHS = ['Bilder']
PAGE_PATHS = ['Seiten']
```

Erweiterungen

`pelican-plugins` und `pelican-themes` erweitern das System.

Installation mit `git clone --recursive https://github.com/getpelican/pelican-plugins` bzw. `git clone --recursive https://github.com/getpelican/pelican-themes`.

Beim aktualisieren muss die Rekursion explizit aufgerufen werden: `git pull --recurse-submodules`

Alternativ mit `Sourcetree` die `git` Archive verwalten.

Schreiben

Am besten legt man sich eine Musterdatei an, die man dann als Vorlage für die Artikel benutzt. Diese Datei sollte die oben beschriebenen Daten enthalten. `title`, `tags`, `date` und `author` sind dabei zwingend erforderlich, der Rest ist optional.

Nach einer Leerzeile beginnt der eigentliche Text des Artikels. Hierbei können die bekannten Markdown Befehle verwendet werden.

Links zu externen Quellen werden in der Form:

```
[Linktext](http://example.com/seite.html)
```

angegeben.

Lokale Links sind Sonderfälle, sie werden über relative URLs eingebunden:

Link auf eine lokale Datei:

```
[Alternativtext]({static}/images/Pelican_installieren.pdf)
```

Zum Einbinden eines Bildes wird ein `!` vorangestellt.

```
![Alternativtext]({static}/images/Dateiname.jpg)
```

`{static}` ist dabei eine Variable die automatisch auf die Startadresse des Blogs gesätzt wird.

Anleitungen

Pelican Dokumentation

- Pelican Anleitung
- Fabric
- Invoke
- git
- Virtualenv
- Installationsvideo (englisch)