Statische Websites mit Hugo erstellen

Software Freedom Day
19. September 2015

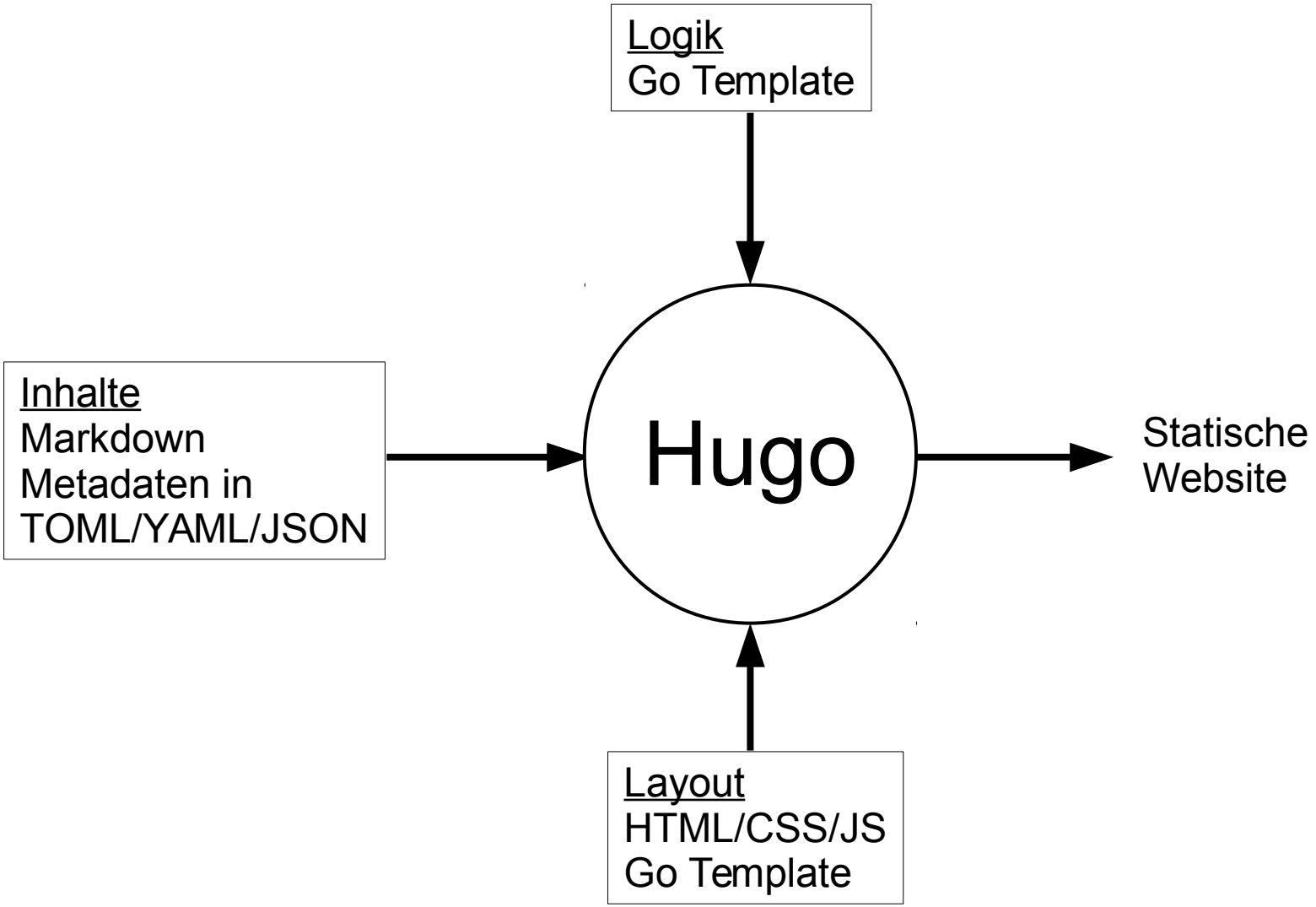
Harald Weidner
hweidner@gmx.net

http://gohugo.io/

Hugo

- Generator für statische Websites
- Trennung von Inhalt, Layout und Logik
- Autor: Steve Francia + >200 Unterstützer
- Geschrieben in Go
- Veröffentlicht unter der Simple Public License (SimPL)

Hugo


Logik
Go Template
Inhalte
Markdown
Metadaten in
TOML/YAML/JSON
Hugo
Statische
Website
Layout
HTML/CSS/JS
Go Template

Hugo – Basistechnologien

- Markdown
  - Syntax für Textauszeichnung
- TOML (oder YAML oder JSON)
  - Syntax für Konfigurationsparameter und Metadaten
- Go Templates
  - Textdateien (i.d.R. HTML) mit Platzhaltern
  - (Begrenzte) Programmiermöglichkeiten

Markdown

- Syntax für Formatierungsauszeichnung von Texten
- Ursprünglich ein Konvertierungstool in Perl
- Angelehnt an die Formatierung textbasierter E-Mails
- Homepage: http://daringfireball.net/projects/markdown/
- Go-Bibliothek: https://github.com/russross/blackfriday

Markdown – Beispiel

```
Was ist Freie Software?
=======================

Freie und Open Source Software
------------------------------

Der Begriff _Freie Software_ (_free software_) wurde Anfang
der 1970'er Jahre von Richard Stallman geprägt. Nach Stallmans
Ansicht sollte Software von jedem benutzt, modifiziert und
weitergegeben werden dürfen.

Die Rechte des Anwenders
------------------------

Gemäß der Definitionen hat der Anwender von FOSS die Rechte,

 * die Software für beliebige Zwecke auszuführen;
 * die Funktionsweise und Interna der Software einzusehen und zu
   untersuchen;
 * die Software zu verändern und an seine persönlichen Bedürfnisse
   anzupassen;
 * die Software an andere Personen weiter zu geben oder öffentlich
   (z.B. auf einer Webseite) anzubieten; und
 * die eigenen Modifikationen weiter zu geben.
```

Markdown – Struktur

```
# Überschrift 1. Ordnung

## Überschrift 2. Ordnung

### Überschrift 3. Ordnung

#### Überschrift 4. Ordnung

##### Überschrift 5. Ordnung

###### Überschrift 6. Ordnung

Überschrift 1. Ordnung
======================

Überschrift 2. Ordnung
----------------------
```

Markdown – Absätze

```
Dies ist der erste Absatz.
Diese Zeile gehört ebenfalls zum ersten Absatz.

Hier beginnt der zweite Absatz.
Ein harter Zeilenumbruch wird durch zwei Leerzeichen_ _
am Ende der Zeile eingefügt.

> Dies ist ein Absatz
> mit zitiertem Text
> > der auch verschachtelt
> > sein kann
> wenn es sein muss.

* eine
* unnummerierte
* Aufzählung

1. eine
1. nummerierte
1. Aufzählung
```

Markdown – Textauszeichnung

```
_Kursivschrift_
__Fettdruck__
___Kursiv und Fett___

*Kursivschrift*
**Fettdruck**
***Kursiv und Fett***


Benutze die `fmt.Prinln()` Funktion:

    func main() {
        fmt.Println("Hello, World!")
    }
```

Markdown – Hyperlinks

```
Hyperlink:

[Linux-Workshop Köln](http://www.uni-koeln.de/themen/linux/ "LiWoK")

Siehe meine [Über mich](/about/) Seite für Einzelheiten.

<http://www.example.com>
<address@example.com>


Bilder:

![Zitrone](/pics/lemmon.jpg "Lemmon")
```

Go Templates

- Template-Mechanismus der Programmiersprache Go.
- Textvorlage mit Platzhaltern, die bei Anwendung ersetzt werden.
- Funktionen:
  - Variablen, arithmetische Operationen
  - Bedingungen
  - Schleifen, Iteration durch Datenstrukturen
  - Eingebaute und selbstdefinierte Funktionen

Go Templates – Beispiele

```
<title>{{.Title}}</title>
This article was posted on {{.Date}}.

{{ if isset .Author }} Written by {{.Author}}. {{ end }}

{{ $address := "Domkloster 4" }}
Come to {{ $address }}.

{{ template "partials/header.html" . }}
Das hier steht zwischen Header und Footer.
{{ template "partials/footer.html" . }}

{{ range $item := array }}
  Artikel: {{ $item }}
{{ end }}
```

Weitere Informationen:
http://gohugo.io/templates/go-templates/
http://golang.org/pkg/text/template/

Metadaten – Formate

- Benötigt für:
  - Meta-Informationen („Front Matter“) für Webseiten
  - Hugo Konfigurationsdatei
- Unterstützte Formate:
  - **TOML** – Tom's Obvious Minimal Language
    https://github.com/toml-lang/toml
  - **YAML** – YAML Ain't Markup Language
    http://yaml.org/
  - **JSON** – JavaScript Object Notation
    http://json.org/

Front Matter in TOML

```
+++
title = "Was ist freie Software?"
date = "2014-04-08"
description = "Definitionen von Freier bzw. Open Source Software"
tags = [ "Freiheit", "Open Source", "Stallmann", "Lizenz" ]
categories = [
  "Freie Software"
  "Linux"
]
draft = false
publishdate = "2015-09-19"
+++
```

Front Matter in YAML

```
---
title: "Was ist freie Software?"
date: "2014-04-08"
description: = "Definitionen von Freier bzw. Open Source Software"
tags: [ "Freiheit", "Open Source", "Stallmann", "Lizenz" ]
categories:
 - "Freie Software"
 - "Linux"
draft: false
publishdate: "2015-09-19"
---
```

Front Matter in JSON

```
{
  "title": "Was ist freie Software?",
  "date": "2014-04-08",
  "description": "Definitionen von Freier bzw. Open Source Software",
  "tags": [ "Freiheit", "Open Source", "Stallmann", "Lizenz" ],
  "categories": [
    "Freie Software",
    "Linux",
  ],
  "draft": "false",
  "publishdate": "2015-09-19"
}
```

Hugo – Installation

- Mit lauffähiger Go-Umgebung:
  `go get github.com/spf13/hugo`

- Binärpaket-Installation für MacOS/Homebrew:
  `brew install hugo`

- Binärpakete für Linux (DEB/tar.gz), FreeBSD, Windows, Mac unter
  https://github.com/spf13/hugo/releases

Hugo – Kommandos

- `hugo` (ohne weitere Parameter)
  Generiert die statische Website.
- `hugo server`
  Startet Hugo als Webserver (für Tests).
- `hugo config`
  Gibt die Konfiguration aus.
- `hugo new …`
  Erstellt neue Site oder neues Dokument
- `hugo help`
  Zeigt Hilfetext an.

Hugo – Verzeichnisstruktur

Verzeichnisse der obersten Ebene:

`content/`	Inhalte in Markdown
`layouts/`	Layout-Vorlagen
`static/`	Statische Dateien (CSS, JS, Bilder)
`public/`	Generierte Webseiten
`config.toml`	Hugo Konfiguration

Hugo – Konfigurationsdatei

Datei: config.toml

```
baseurl = "http://sfd.koelnerlinuxtreffen.de/
contentdir = "content"
layoutdir = "layouts"
publishdir = "public"
builddrafts = false
canonifyurls = true

[taxonomies]
  category = "categories"
  tag = "tags"

[params]
  description = "Software Freedom Day"
  author = "Harald Weidner"
```

Alle Konfigurationsparameter: http://gohugo.io/overview/configuration/

Hugo – Verzeichnisstruktur (Bsp.)

```
content/news/hello.md
content/news/sfd.md
content/news/froscon.md
content/wiki/start.md
content/wiki/entry.md
content/wiki/subpage/page.md
content/about.md
content/toc.md
content/impressum.md
```

Hugo – Verzeichnisstruktur (Bsp.)

```
layouts/index.html
layouts/news/single.html
layouts/news/list.html
layouts/news/summary.html
layouts/wiki/single.html
layouts/_default/single.html
layouts/partials/header.html
layouts/partials/footer.html

static/css/
static/js/
static/img/
```

Hugo – Inhalte

- Inhaltsdateien bestehen aus
  - Front Matter in TOML (oder YAML oder JSON)
  - Inhalt in Markdown

```
+++
title  = "Was ist freie Software?"
date   = "2014-04-08"
+++

Was ist Freie Software?
=======================

Der Begriff _Freie Software_ (_free software_) wurde Anfang der 1970'er
Jahre von Richard Stallman geprägt. Stallman arbeitete gemeinsam mit
Kollegen am Massachusetts Institute of Technology an Programmen
...
```

Hugo – Layout

- Layouts sind HTML-Vorlagen mit Platzhaltern
- Verwenden Go Template Language
- Layouts für einzelne Seiten (single.html) oder Übersichtsseiten (list.html)
- Verschiedene Layouts für Bereiche möglich (z.B. Blog / Wiki / Gallery / …)
- Einbinden von gemeinsamen Bestandteilen (Subheader)

Hugo – Layout (Beispiel)

```
<html>

  <head>
    <title>{{.Title}}</title>
  </head>

  <body>
    <p>Posted on {{.Date}}.</p>
    {{.Content}}
  </body>

</html>
```

Liste aller verfügbaren Variablen: http://gohugo.io/templates/variables/

Hugo – Logik

- (Limitierte) Unterstützung für Logikelemente
- Beispiele
  - Liste einer Zusammenfassung der 10 neuesten Artikel
  - Liste aller Artikel, die zu einer Kategorie gehören
  - Schlagwortverzeichnis / TagCloud
- Benötigte Templates
  - Liste / View

Hugo – Logik (Beispiel)

Template list.html

```
{{ template "header.html" . }}

<h1>{{ .Title }}</h1>

{{ range first 10 .Data.Pages.ByDate }}
    {{ .Render "summary" }}
{{ end }}

{{ template "footer.html" . }}
```

Template summary.html

```
<h2>{{ .Title }}</h2>

<p>Posted on {{ .Date }} -
{{ .WordCount }} Words</p>

{{ .Summary }}

<a href='{{ .Permalink }}'>
Read more</a>
```

Hugo – Themes

- Verwendung vorgefertigter Layouts (Themes) anstelle eigener Layouts
- Zentrale Sammlung: https://github.com/spf13/hugoThemes/
- Download
  - Themes sind meist eigene Projekte auf GitHub, z.B. git clone https://github.com/key-amb/hugo-theme-bootie-docs
- Einbindung:
  - In config.toml: `theme = hugo-theme-bootie-docs`
  - Verzeichnis `layouts` löschen!

Hugo – Weitere Funktionen

- Ordnungskriterien (taxonomies)
  - Klassifikation von Artikeln nach verschiedenen Kriterien
  - Häufig: Kategorien (z.B. Politik, Technik, Recht) oder Tags (Schlagwörter)
- Musterartikel (archetypes)
  - Vorgefertiges Fragment für neuen Artikel
  - Vorausgefüllter Front Matter
- Inhalt in reStructuredText statt Markdown