Git for Web Publishing

2026-08-26 Frank Siebert

It is time to write the promised documentation for the idee-web-publishing, named such, because it serves the publishing for my German idee-web-site "Idee der eigenen Erkenntnis".

Concept

This publishing grew out of the development I made to get rid of word-press. In its core it stays the same, but it got now much simpler.

The web pages to be published are created via the "Vim Plugin for Web Publishing" [1] .

The web-site folder structure is already shown in that earlier article, but that is not the complete picture of the idee-project folder structure, which is set up as git repository.

But before I dive into these details, let me first give an overview, what this is all about and what the basic idea of the setup is.

Workflow

My previous article described the content writing workflow up to the point, where the new vim command `IdeePublish` performs the handover to the publishing. This handover is nothing more and nothing less but the copying of the created content into the client git repository workspace.

With the authoring process finished, the publishing process is the next step.

Web-sites usually provide

- an index-page listing the latest articles
- archive pages to give access to older articles
- sitemap.xml with update information to guide indexing and re-indexing by search engines
- RSS-feeds to enable users to subscribe via RSS-reader or podcast client

This content contains information about the article, but is not written by the article author. It needs to be generated during the publishing process.

The publishing itself consists of the following steps in the client git repository:

```
# git add .
# git commit .
```

If you are uncertain that everything has been correctly generated, you might now check the result, apply corrections and repeat the two commands. If everything is right, just do:

```
# git push
```

Three Git-Repositories

Authoring happens on a workstation, in my case a notebook. But the web-server runs, who would guess it, on a server.

The commit happens in the git-repository on my notebook, and the push transfers the changes in the repository to a remote git-repository set up as server-repository, which has no working directory. This means, it does not replicate the folders, but only in the git data structure.

This, sadly, cannot simply be served as web-content via a web-server. Therefore a third git-repository, a second client repository, is required, which fetches the pushed changes and serves as web-root for the web-server.

I described the setup of this in the article "Replacing WordPress" in chapter "Idee Website Server Setup" [2], and since this stays unchanged, I see no point in repeating the description here.

Just some words on the conceptional side: The last workflow step, the manual push to the server-git, triggers there a post-receive-hook-function, which invokes a pull in the third git repository serving as www-root.

Client Git Setup

A function is hooked into the pre-commit-hook of the client repository to produce some portal specific content related to the published web-page.

Here we take a look at the relevant folder structure, the setup of the hook and the python implementation which creates that web-site content.

Idee Project Folders

```
frank@Asimov:~/projects/idee$ tree -d
.
├── author
├── config
│   └── hooks
│       └── samples
├── generator
│   └── __pycache__
├── nginx
├── snippets
├── test
└── website
    ├── archive
    ├── article
    ├── audio
    ├── css
    ├── env
    │   └── bootstrap
    │       └── css
    ├── files
    ├── image
    ├── js
    ├── legal
    ├── MathJax -> SimpleMathJax/resources/MathJax/es5
    ├── pdf
    ├── portal
    ├── qrcode
    ├── SimpleMathJax
    │   └── resources
    │       └── MathJax
    │           └── es5
    │               ├── a11y
    │               ├── adaptors
    │               ├── input
    │               │   ├── mml
    │               │   └── tex
    │               │       └── extensions
    │               ├── output
    │               │   ├── chtml
    │               │   │   └── fonts
    │               │   │       └── woff-v2
    │               │   └── svg
    │               │       └── fonts
    │               ├── sre
    │               │   └── mathmaps
    │               └── ui
    └── sitemap
```

In the folder `author` the authored markdown-files are stored.

The folder `config` contains a bash-script to change the git configuration, the pre-commit-hook and the hook-samples provided by the git-developers.

The folder `generator` contains the python programs, which generates the additional portal content required when a new or modified article is published.

The folder `nginx` contains the web-site configuration usually stored in the folder `sites-available` , which is referenced in that folder via logical link.

The folder `snippets` contains markdown snippets to be included in articles during the authoring.

The folder `test` contains some code to test the python code without doing commits.

The folder `website` contains the web-content and is served by nginx as www-root.

The folder `SimpleMathJax` is a git-repository cloned from the SimpleMathJax project. For convenience a symbolic link shortens the path to the javascript used from that repository to render formulas into the HTML-pages.

Git Configuration

In folder `config` a bash file to change the git configuration exists. It configures the location of the git-hooks, which I want to have inside of the git repositories version control. It switches the git-option `quotepath` of, which was required when I used German mediawiki titles as filename for exported articles I had written in the wiki, It switches the git-option `relativPaths` of, to simplify me the checks which processing needs to be triggered in the pre-commit-hook.

config/gitconfig

```
#!/bin/bash

# We develop hooks and want version control for that
git config --local core.hooksPath ./config/hooks

# We want easy reading of German äüö in the file names
git config --local core.quotepath off

# We provide some variable override options in a modified template
# git config --local commit.template ./config/commit-message

# We process the files committed and need absolute file paths from $GIT_DIR
# written into the commit-message
git config --local status.relativePaths false
```

The configuration finally looks as follows:

.git/config

```
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
    hooksPath = ./config/hooks
    quotepath = off
[remote "origin"]
    url = ssh://git@sol/home/git/idee.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
    remote = origin
    merge = refs/heads/master
[status]
    relativePaths = false
```

Pre-Commit-Hook

The pre-commit-hook is the first hook called, when the user invokes `git commit` .

config/hooks/pre-commit

```
#!/bin/bash

git diff-index --name-status HEAD | /usr/bin/python3 generator/pre-commit.py

git status

read -p "Press Enter to continue" </dev/tty
```

The result of the command `git diff-index --name-status HEAD` is passed on into stdin of the pre-commit python program. This result contains the staged changes of the git repository. The commit-message is not yet written and any files changed during the processing, like the index page, the rss-feed, the sitemap pages or the archive pages, can still be added to the ongoing commit, which is done in the respective python modules.

The reason to run `git status` and to wait for `Enter` to be pressed is that git writes its commit message while the pre-commit-hook still adds new created or modified files to the commit. The to be committed changes in the later shown commit message thus do not show the correct status.

I want to see the correct status before finalizing the commit, and showing it here was the simplest way to get it done. Modifying the generated commit message would be another option and the way to go, if the implementation would be for other users than me alone.

Module pre-commit

The pre-commit module is called from the pre-commit-hook.

generator/pre-commit.py

```
"""Website Generator - "fs-commit-msg-hook 2.0".

@author: Frank Siebert
@license: https://creativecommons.org/publicdomain/zero/1.0/deed.en
@date: 2022-03-15

https://wiki.frank-siebert.de/inst/Replacing_Wordpress
https://idee.frank-siebert.de/article/replacing-wordpress.html
... replacing mediawiki

Website Generator uses Beautiful Soup and GIT to manage
publishing.

Trigger Input
-------------
./website/article/*.html

Other Input
-----------
Article related assets in other website subfolders, like
* audio
* file
* image
* js
* pdf
* qrcode
does not trigger the generator to do anything.

Output
------
For de-DE content:
* idee-index.html
* idee-map.xml
* idee-ress.xml
* sitemap.xml

For en-US content:
* concept-index.html
* concept-map.xml
* concept-ress.xml
* sitemap.xml

Website Generator works with Python 3 and up. It works better if lxml
and/or html5lib is installed, as Beautiful Soup states it runs better then.
"""
# Systen Imports
import sys
import re
from pathlib import Path
import getopt
```

```
from bs4 import BeautifulSoup
from bs4.builder._htmlparser import HTMLParserTreeBuilder

from article import Article
from sitemap import SiteMap
from archive import Archive
from rssbuilder import RSSBuilder
from idxbuilder import IDXBuilder


def get_article_list():
    """
    Get the list of added and modified articles from stdin.

    The list of article are be sorted by modification date,
    smallest dates first, to make sure the articles show up with newest/latest
    changed articles first.

    For this we need to look into the html. Because we do not want to read the
    files multiple times, we add the respective soup to the list.

    Returns
    -------
        articles: list of Article or None
    """
    articles = []

    # see man git diff --diff-filter for list of filters
    # M - modified
    # A - added
    searchpattern = re.compile(r"^(M|A).*/article/")

    for line in sys.stdin:
        filename = line[2:].strip()
        if searchpattern.match(line):
            filepath = Path(filename)
            with open(filepath, 'r', encoding="utf-8") as htmlfile:
                html_doc = htmlfile.read()
                htmlfile.close()
                builder = HTMLParserTreeBuilder()
                soup = BeautifulSoup(html_doc, builder=builder)
                article = Article(soup)
                articles.append(article)

    articles.sort(key=lambda x: x.get_modified())
    return articles if len(articles) > 0 else None


if __name__ == "__main__":

    HELPTEXT = 'Usage: git diff-index --name-status HEAD | '\
            '/usr/bin/python3 generator/pre-commit.py\n'

    try:
        opts, args = getopt.getopt(sys.argv[1:], "h:", ["help"])

    except getopt.GetoptError:
        print(HELPTEXT)
        sys.exit(2)

    for opt, arg in opts:
        if opt in {"-h", "--help"}:
            print(HELPTEXT)
            sys.exit()

    ARTICLES = get_article_list()

    if ARTICLES:
        SiteMap(ARTICLES).update()
        # Generate Archive
        Archive(ARTICLES).update()
        # Generate RSS feed (bilingual)
        RSSBuilder(ARTICLES).update()
        # Generate Index Pages (English and German Version)
        IDXBuilder(ARTICLES).update()
```

```
    sys.exit(0)
```

This module is called by the pre-commit-hook, which provides the short list of staged changes via stdin. Modified or added articles are read, their beautiful soup created and via Article-instance appended to the articles work list.

This work list then is used to run the update function of the SiteMap, the Archive, the RSSBuilder and the IDXBuilder.

Module article

The module `article` contains the class `Article` , which provides access to meta-data and to the HTML-soup created via the BeautifulSoup module.

generator/article.py

```
"""
The class Article provides access methods to meta data stored in the soup and to
the soup itself.

"""

import re
from gitmsgconstants import GitMsgConstants as gmc

class Article():
    """
    The class Article
    """

    def __init__(self, soup):
        """
        Initialize Article Data
        """
        self.soup = soup

    def get_modified(self):
        """
        Returns:
        --------
        datetime string of last modification in ISO format
        """
        return self.soup.find("meta",
                              attrs={"property":
                                     "article:modified_time"})["content"]

    def get_language(self):
        """
        Returns:
        --------
        language string like 'de-DE'
        """
        return self.soup.find("html")["lang"].strip()

    def get_soup(self):
        """
        Returns:
        --------
        html soup of the article
        """
        return self.soup

    def get_published(self):
        """
        Returns:
        --------
        datetime string of first publishing in ISO format
        """
        return self.soup.find("meta",
                              attrs={"property":
                                     "article:published_time"})["content"][:19]
```

```
    def get_author(self):
        """
        Returns:
        --------
        author namen of the article
        """
        return self.soup.find("meta",
                              attrs={"property":
                                     "article:author"})["content"].strip()

    def get_title(self):
        """
        Returns:
        --------
        title of the article
        """
        return self.soup.find("meta", attrs={"property":
                                             "og:title"})["content"].strip()

    def get_urn(self):
        """
        Returns:
        --------
        urn of the article
        """
        return self.soup.find("meta", attrs={"property":
                                             "article:urn"})["content"].strip()

    def get_site(self):
        """
        Returns:
        --------
        site the article belongs to, "Idee" or "Concept"
        """
        return self.soup.find("meta", attrs={"property":
                                             "og:site_name"})["content"].strip()

    def get_abstract(self):
        """
        Returns:
        --------
        the text of the first paragraph, or the first 406 characters of it,
        whichever is shorter.
        """
        text = self.soup.find("p").text.split()  # I forgot why I split
        return " ".join(text)[0:406]

    def prepare_article_tag_for_rss(self):
        """
        Returns:
        --------
        article_tag

        For RSS, the urls are changed from relative to absolute.
        """

        article_tag = self.soup.find("article")
        host = gmc.website + "/"

        # RSS is downloaded, there is no use case for relatvie links
        # even if RSS consumer theoritically could compute them
        # to absolute links

        # "../" becomes "https://idee.frank-siebert.de/"
        tags = article_tag.find_all(re.compile(r".*"), attrs={
            "href": re.compile(r"^\.\./")})
        for tag in tags:
            href = tag.attrs["href"]
            href = href.replace("../", host)
            tag.attrs.update({"href": href})

        tags = article_tag.find_all(re.compile(r".*"), attrs={
            "src": re.compile(r"^\.\./")})
        for tag in tags:
            href = tag.attrs["src"]
```

```
            href = href.replace("../", host)
            tag.attrs.update({"src": href})

        # "./" becomes "https://idee.frank-siebert.de/article/"
        tags = article_tag.find_all("a", attrs={
            "href": re.compile(r"^\./")})
        for tag in tags:
            href = tag.attrs["href"]
            href = href.replace("./", host + "article/")
            tag.attrs.update({"href": href})

        article_tag.prettify()
        return article_tag
```

Module gitmsgconstants

The module gitmsgconstants defines some constants used in the different generator modules.

generator/gitmsgconstants

```
"""
GitMsgConstants provides project wide constants.

@author: Frank Siebert
@license: https://creativecommons.org/publicdomain/zero/1.0/deed.en
@date: 2022-03-15

No Instance is required. Could leverage in future a config file.
"""
from pathlib import Path


class GitMsgConstants():
    """
    Dispatch the lines of the git message to registered workers.

    Parameters
    ----------
    gitmessagepath : Path
        Path as type str or type Path pointing to the git message.

    msgworkers : List of MsgWorker
        The list of message workers is used as worker queue. Workers first
        in the queue get their workitems first.

        Workers can return their work result to be picked up by
        later workers.

    Returns
    -------
    GitMsgConstants.

    """

    generator = "pandoc, fs-commit-msg-hook 1.0"
    website = "https://idee.frank-siebert.de"
    pdfimage = "3cd97bab8bb20288768b35fd72979ec3bbf4b2a8.png"

    plainpath = Path("plain")
    confpath = Path("config")
    sitepath = Path("website")
    articlepath = sitepath / "article"
    audiopath = sitepath / "audio"
    csspath = sitepath / "css" / "fs.css"
    headerpath = sitepath / "portal" / "header.html"
    imagepath = sitepath / "image"
    pdfpath = sitepath / "pdf"
    qrpath = sitepath / "qrcode"

    migrationlistpath = confpath / "migrationlist.csv"
    publishingdatapath = sitepath / "pubmetadata.csv"
```

```
    pdfdraft = "pdf:draft"
    locale = "og:locale"

    archivepath = sitepath / "archive"
    idee_archive = archivepath / Path("idee-archive.html")
    concept_archive = archivepath / Path("concept-archive.html")
    sitemap = sitepath / Path("sitemap.xml")
    idee_map = sitepath / Path("idee-map.xml")
    concept_map = sitepath / Path("concept-map.xml")
    sitemappath = sitepath / "sitemap"
    map_template = sitepath / "portal" / "monthly-map.xml"
    archive_template = sitepath / "portal" / "monthly-archive.html"

    idee_rss = sitepath / Path("idee-rss.xml")
    concept_rss = sitepath / Path("concept-rss.xml")

    idee_index = sitepath / Path("idee-index.html")
    concept_index = sitepath / Path("concept-index.html")


if __name__ == "__main__":
    pass
```

Module sitemap

The sitemap module creates and updates sitemap xml-files.

generator/sitemap.py

```
"""
Update the sitemap of the webseite.

@author: Frank Siebert
@license: https://creativecommons.org/publicdomain/zero/1.0/deed.en
@date: 2022-03-15

@author: Frank Siebert
"""
import re
import datetime
import subprocess

from bs4 import BeautifulSoup
from bs4.builder._lxml import LXMLTreeBuilderForXML
from gitmsgconstants import GitMsgConstants as gmc

URLSET_TAG = "urlset"
URL_TAG = "url"
LOC_TAG = "loc"
LASTMOD_TAG = "lastmod"
# Not used:
# CHANGEFREQ_TAG = "changefreq"
# PRIORITY_TAG = "priority"

INDEX_TAG = "sitemapindex"
SIDEMAP_TAG = "sitemap"


class SiteMap():
    """Manage all changes in the sitemaps."""

    def __init__(self, articles):
        """
        Initialize changelists.

        Returns
        -------
        None.

        """
```

```
        # map information for German page changes on site "Idee".
        self.de_list = []
        # map information for English page changes on site "Concept".
        self.en_list = []
        # The time of the update
        self._nowdate = datetime.datetime.now().isoformat()

        self._updates = articles

    def update(self):
        """
        Iterate over changes and update respective sitemaps.

        Add the respective sitemaps to their respective change list.
        The information about the changed html pages comes from
        PubMetaData.instance._updates  and
        PubMetaData.instance._deletions  .

        Returns
        -------
        None.

        """
        for article in self._updates:
            creation_month = article.get_published()[0:7]

            site = article.get_site().lower()
            sitemap_path = gmc.sitemappath / f"{site}-{creation_month}.xml"

            if site in "idee":
                if sitemap_path not in self.de_list:
                    self.de_list.append(sitemap_path)
            else:
                if sitemap_path not in self.en_list:
                    self.en_list.append(sitemap_path)

            self._update(sitemap_path, article)

        self._update_de()
        self._update_en()
        self._update_main()

    def _update_de(self):
        """Update idee-map.xml."""
        if len(self.de_list) == 0:
            return

        with open(gmc.idee_map, 'r', encoding="utf-8") as sitemap_file:
            xml_doc = sitemap_file.read()
            sitemap_file.flush()
            sitemap_file.close()

        builder = LXMLTreeBuilderForXML
        soup = BeautifulSoup(xml_doc, builder=builder, features='xml')

        for sitemap_path in self.de_list:
            url = gmc.website + "/" + sitemap_path.parts[-2] + \
                "/" + sitemap_path.parts[-1]
            tag = soup.find(LOC_TAG, text=re.compile(r"" + url))

            if not tag:
                tag = soup.find(INDEX_TAG)
                new_tag = soup.new_tag(SIDEMAP_TAG)
                tag.append(new_tag)
                tag = new_tag
                new_tag = soup.new_tag(LOC_TAG)
                new_tag.string = url
                tag.append(new_tag)
                new_tag = soup.new_tag(LASTMOD_TAG)
                tag.append(new_tag)
            else:
                tag = tag.parent

            # tag holds now the correct SIDEMAP_TAG.
            # Either it had been found or created.
            # All used child tags exist also.
```

```
            tag = tag.find(LASTMOD_TAG)
            tag.string = self._nowdate

        xml_doc = soup.prettify()

        with open(gmc.idee_map, 'w', encoding="utf-8") as sitemap_file:
            print(xml_doc, file=sitemap_file)
            sitemap_file.flush()
            sitemap_file.close()
            subprocess.run(['git', 'add', gmc.idee_map], check=True)

    def _update_en(self):
        """Update concept-map.xml."""
        if len(self.en_list) == 0:
            return

        with open(gmc.concept_map, 'r', encoding="utf-8") as sitemap_file:
            xml_doc = sitemap_file.read()
            sitemap_file.flush()
            sitemap_file.close()

        builder = LXMLTreeBuilderForXML
        soup = BeautifulSoup(xml_doc, builder=builder, features='xml')

        for sitemap_path in self.en_list:
            url = gmc.website + "/" + sitemap_path.parts[-2] + \
                "/" + sitemap_path.parts[-1]
            tag = soup.find(LOC_TAG, text=re.compile(r"" + url))

            if not tag:
                tag = soup.find(INDEX_TAG)
                new_tag = soup.new_tag(SIDEMAP_TAG)
                tag.append(new_tag)
                tag = new_tag
                new_tag = soup.new_tag(LOC_TAG)
                new_tag.string = url
                tag.append(new_tag)
                new_tag = soup.new_tag(LASTMOD_TAG)
                tag.append(new_tag)
            else:
                tag = tag.parent

            # tag holds now the correct SIDEMAP_TAG.
            # Either it had been found or created.
            # All used child tags exist also.

            tag = tag.find(LASTMOD_TAG)
            tag.string = self._nowdate

        xml_doc = soup.prettify()

        with open(gmc.concept_map, 'w', encoding="utf-8") as sitemap_file:
            print(xml_doc, file=sitemap_file)
            sitemap_file.flush()
            sitemap_file.close()
            subprocess.run(['git', 'add', gmc.concept_map], check=True)

    def _update_main(self):
        """Update sitemap.xml."""
        if len(self.de_list) == 0 and len(self.en_list) == 0:
            return

        with open(gmc.sitemap, 'r', encoding="utf-8") as sitemap_file:
            xml_doc = sitemap_file.read()
            sitemap_file.flush()
            sitemap_file.close()

        builder = LXMLTreeBuilderForXML
        soup = BeautifulSoup(xml_doc, builder=builder, features='xml')

        if len(self.de_list) > 0:
            url = gmc.website + "/" + gmc.idee_map.name
            tag = soup.find(LOC_TAG, text=re.compile(r"" + url))
            # We know in this case, that the tag exists
            tag = tag.parent
```

```
            tag = tag.find(LASTMOD_TAG)
            tag.string = self._nowdate

        if len(self.en_list) > 0:
            url = gmc.website + "/" + gmc.concept_map.name
            tag = soup.find(LOC_TAG, text=re.compile(r"" + url))
            # We know in this case, that the tag exists
            tag = tag.parent
            tag = tag.find(LASTMOD_TAG)
            tag.string = self._nowdate

        xml_doc = soup.prettify()

        with open(gmc.sitemap, 'w', encoding="utf-8") as sitemap_file:
            print(xml_doc, file=sitemap_file)
            sitemap_file.flush()
            sitemap_file.close()
            subprocess.run(['git', 'add', gmc.sitemap], check=True)

    @staticmethod
    def _update(sitemap_path, article):
        sitemap_path.resolve()
        if sitemap_path.exists():
            with open(sitemap_path, 'r', encoding="utf-8") as sitemap_file:
                xml_doc = sitemap_file.read()
                sitemap_file.flush()
                sitemap_file.close()
        else:
            gmc.map_template.resolve()
            with open(gmc.map_template, 'r', encoding="utf-8") as sitemap_file:
                xml_doc = sitemap_file.read()
                sitemap_file.flush()
                sitemap_file.close()

        builder = LXMLTreeBuilderForXML
        soup = BeautifulSoup(xml_doc, builder=builder, features='xml')


        urn = article.get_urn()
        article_url = gmc.website + f"/article/{urn}.html"

        tag = soup.find(LOC_TAG, text=re.compile(r"" + article_url))
        if not tag:
            tag = soup.find(URLSET_TAG)
            new_tag = soup.new_tag(URL_TAG)
            tag.append(new_tag)
            tag = new_tag
            new_tag = soup.new_tag(LOC_TAG)
            new_tag.string = article_url
            tag.append(new_tag)
            new_tag = soup.new_tag(LASTMOD_TAG)
            tag.append(new_tag)
        else:
            tag = tag.parent

        # tag holds now the correct URL_TAG.
        # Either it had been found or created.
        # All used child tags exist also.

        tag = tag.find(LASTMOD_TAG)
        tag.string = article.get_modified()

        xml_doc = soup.prettify()

        with open(sitemap_path, 'w', encoding="utf-8") as sitemap_file:
            print(xml_doc, file=sitemap_file)
            sitemap_file.flush()
            sitemap_file.close()
            subprocess.run(['git', 'add', sitemap_path], check=True)
```

Sitemap Template

The module sitemap uses a template to create new sitemap xml files as needed.

website/portal/monthly-map.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

</urlset>
```

Module archive

The module archive creates or updates archive files, one per month.

generator/archive.py

```
"""
Update the archive of the webseite.

@author: Frank Siebert
@license: https://creativecommons.org/publicdomain/zero/1.0/deed.en
@date: 2022-03-15
"""
import re
import datetime
import subprocess

from bs4 import BeautifulSoup
from bs4 import Comment
from bs4.builder._htmlparser import HTMLParserTreeBuilder
from gitmsgconstants import GitMsgConstants as gmc

class Archive():
    """Manage all changees in the archive."""

    def __init__(self, articles):
        """
        Initialize changelists.

        Returns
        -------
        None.

        """
        # map information for German page changes on site "Idee".
        self.de_list = []
        # map information for English page changes on site "Concept".
        self.en_list = []
        # The time of the update
        self._nowdate = datetime.datetime.now().isoformat()

        self._updates = articles

    def update(self):
        """
        Iterate over changes and update respective archive pages.

        Add the archive pages to their respective change list.

        Returns
        -------
        None.

        """
        for article in self._updates:
            creation_month = article.get_published()[0:7]
```

```
            site = article.get_site().lower()
            archive_path = gmc.archivepath / f"{site}-{creation_month}.html"

            if site in "idee":
                if archive_path not in self.de_list:
                    self.de_list.append(archive_path)
            else:
                if archive_path not in self.en_list:
                    self.en_list.append(archive_path)

            soup = self._update(archive_path, article)
            html_doc = soup.prettify()

            with open(archive_path, 'w', encoding="utf-8") as archive_file:
                print(html_doc, file=archive_file)
                archive_file.flush()
                archive_file.close()
                subprocess.run(['git', 'add', archive_path], check=True)

        self._update_de()
        self._update_en()

    def _update_de(self):
        """Update idee-archive.html."""
        if len(self.de_list) == 0:
            return

        with open(gmc.idee_archive, 'r', encoding="utf-8") as archive_file:
            html_doc = archive_file.read()
            archive_file.flush()
            archive_file.close()

        builder = HTMLParserTreeBuilder
        soup = BeautifulSoup(html_doc, builder=builder)

        for archive_path in self.de_list:
            url = './' + archive_path.name
            tag = soup.find("a", href=re.compile(r"" + url))

            if not tag:
                tag = soup.find("main")
                new_tag = soup.new_tag("h3")
                tag.insert(0, new_tag)
                tag = new_tag
                new_tag = soup.new_tag("a")
                new_tag.attrs.update({"href": url})
                new_tag.string = archive_path.name
                tag.append(new_tag)

        html_doc = soup.prettify()

        with open(gmc.idee_archive, 'w', encoding="utf-8") as archive_file:
            print(html_doc, file=archive_file)
            archive_file.flush()
            archive_file.close()
            subprocess.run(['git', 'add', gmc.idee_archive], check=True)

    def _update_en(self):
        """Update concept-archive.html."""
        if len(self.en_list) == 0:
            return

        with open(gmc.concept_archive, 'r', encoding="utf-8") as archive_file:
            html_doc = archive_file.read()
            archive_file.flush()
            archive_file.close()

        builder = HTMLParserTreeBuilder
        soup = BeautifulSoup(html_doc, builder=builder)

        for archive_path in self.en_list:
            url = './' + archive_path.name
            tag = soup.find("a", href=re.compile(r"" + url))

            if not tag:
                tag = soup.find("main")
```

```
                new_tag = soup.new_tag("h3")
                tag.insert(0, new_tag)
                tag = new_tag
                new_tag = soup.new_tag("a")
                new_tag.attrs.update({"href": url})
                new_tag.string = archive_path.name
                tag.append(new_tag)

        html_doc = soup.prettify()

        with open(gmc.concept_archive, 'w', encoding="utf-8") as archive_file:
            print(html_doc, file=archive_file)
            archive_file.flush()
            archive_file.close()
            subprocess.run(['git', 'add', gmc.concept_archive], check=True)

    @staticmethod
    def _update(archive_path, article, article_loc="../article/"):
        is_new = None
        archive_path.resolve()
        if archive_path.exists():
            with open(archive_path, 'r', encoding="utf-8") as archive_file:
                html_doc = archive_file.read()
                archive_file.flush()
                archive_file.close()
                is_new = False
        else:
            gmc.archive_template.resolve()
            with open(gmc.archive_template, 'r', encoding="utf-8") as archive_file:
                html_doc = archive_file.read()
                archive_file.flush()
                archive_file.close()
                is_new = True

        builder = HTMLParserTreeBuilder
        soup = BeautifulSoup(html_doc, builder=builder)

        if is_new:
            tag = soup.find("body")
            # SSI header injection is a function of the language
            if "idee-" in str(archive_path):
                new_tag = Comment('# include file="/portal/idee-header.html" ')
                language = "de"
                site_name = "Idee"
                title_prefix = "Archiv"
            else:
                new_tag = Comment(
                    '# include file="/portal/concept-header.html" ')
                language = "en"
                site_name = "Concept"
                title_prefix = "Archive"
            tag.insert(0, new_tag)

            tag = soup.find("html")
            tag.attrs.update({"lang": language, "xml:lang": language})
            tag = soup.find("meta", property="og:site_name")
            tag.attrs.update({"Content": site_name})

            tag = soup.find("title")
            pubdate = article.get_published()[0:7]
            tag.string = f"{title_prefix} {pubdate}"

            tag = soup.find("h1")
            tag.string = f"{title_prefix} {pubdate}"

        urn = article.get_urn()
        article_url = article_loc + f"{urn}.html"

        tag = soup.find("a", href=article_url)
        if not tag:
            tag = soup.find("h1")

            new_tag = soup.new_tag("article")
            if tag:  # true in archive, false in index page
                tag.insert_after(new_tag)
            else:
```

```
                tag = soup.find("main")
                tag.insert(0, new_tag)
            tag = new_tag

            new_tag = soup.new_tag("header")
            tag.append(new_tag)
            tag = new_tag

            new_tag = soup.new_tag("h2")
            tag.append(new_tag)
            tag = new_tag

            new_tag = soup.new_tag("a")
            new_tag.attrs.update({"href": article_url})
            new_tag.string = article.get_title()
            tag.append(new_tag)
            tag = tag.parent  # header

            new_tag = soup.new_tag("div")
            tag.append(new_tag)
            tag = new_tag

            pubtime = article.get_published()
            new_tag = soup.new_tag("time")
            new_tag.attrs.update({"datetime": pubtime,
                                  "pubdate": "true"})
            new_tag.string = pubtime[:10]
            tag.append(new_tag)

            new_tag = soup.new_tag("address")
            new_tag.string = article.get_author()
            tag.append(new_tag)
            tag = tag.parent.parent  # article

            new_tag = soup.new_tag("p")
            tag.append(new_tag)
            tag = new_tag

            new_tag = soup.new_tag("a")
            new_tag.attrs.update({"href": article_url})
            new_tag.string = "..."
            tag.append("placeholder")  # for the article abstract
            tag.append(new_tag)
            tag = tag.parent  # article

            new_tag = soup.new_tag("hr")
            tag.append(new_tag)
        else:
            tag = tag.parent.parent.parent  # article

        # tag holds now the article tag.
        # Either it had been found or created.
        # All used child tags exist also.

        # Write or update the article abstract
        tag = tag.find("p")
        tag = tag.find("a")
        tag.previousSibling.replace_with(article.get_abstract())

        # We give every anchor a tabindex
        # 5 Tabindexes are in the portal header
        index = 6
        tags = soup.find_all(re.compile(r"^a$|^audio$|^input$"))
        for tag in tags:
            tag.attrs.update({"tabindex": index})
            index += 1

        return soup
```

Archive Template

The module archive uses a template to create new monthly archive pages as needed.

website/portal/monthly-archive.html

```
<!DOCTYPE html>
<html lang="de-DE" xml:lang="de-DE" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta charset="utf-8"/>
  <meta content="pandoc, fs-commit-msg-hook 1.0" name="generator"/>
  <meta content="noindex" name="robots">
  <meta content="width=device-width, initial-scale=1.0, user-scalable=yes" name="viewport"/>

  <meta content="Idee" property="og:site_name"/>
  <link href="../css/fs.css" rel="stylesheet"/>
  <link href="../idee-rss.xml" rel="alternate" type="application/rss+xml" title="RSS"/>
  <link href="../image/favicon.ico" rel="icon" type="image/x-icon"/>
  <title>
  Archive 2021-01
  </title>
 </head>
 <body>
  <main>
   <h1>
    Archive xyz
   </h1>
  </main>
 </body>
</html>
```

Module rssbuilder

The module rssbuilder creates the rss files for english and german content.

generator/rssbuilder.py

```
"""
Update the rss feed of the webseite.

@author: Frank Siebert
@license: https://creativecommons.org/publicdomain/zero/1.0/deed.en
@date: 2022-03-15

All links provided relative to the /article/ folder

@author: Frank Siebert
"""
import re
import datetime
import subprocess
import xml.dom.minidom
from bs4 import BeautifulSoup
from bs4.builder._htmlparser import HTMLParserTreeBuilder
from gitmsgconstants import GitMsgConstants as gmc

CHANNEL_TAG = "channel"
LASTBUILD_TAG = "lastBuildDate"
ITEM_TAG = "item"
TITLE_TAG = "title"
LINK_TAG = "link"
PUBDATE_TAG = "pubDate"
GUID_TAG = "guid"
DESCRIPTION_TAG = "description"
CONTENT_TAG = "content:encoded"
ENCLOSURE_TAG = "enclosure"
AUDIO_TAG = "audio"

# for the testing on server sol
# HOST = "http://sol:88/"
# for the website
HOST = gmc.website + "/"
```

```
# Number of items to included into the RSS feed
ITEM_COUNT = 15


class RSSBuilder():
    """Manage all changees in the sitemaps."""

    def __init__(self, articles):
        """
        Initialize changelists.

        Returns
        -------
        None.

        """
        # information for German page changes on site "Idee".
        self.de_list = []
        # information for English page changes on site "Concept".
        self.en_list = []
        # The time of the update
        self._nowdate = datetime.datetime.now().isoformat()
        # soup of currently processed RSS xml
        self._rss_xml = None
        # soup tag of currently processed article
        self._article_tag = None

        for article in articles:
            site = article.get_site().lower()
            if site in "idee":
                self.de_list.append(article)
            else:
                self.en_list.append(article)

    def update(self):
        """
        Iterate over changes and update respective rss files.

        Returns
        -------
        None.

        """
        # Update idee-rss.xml.
        if len(self.de_list) > 0:
            self._update(self.de_list, gmc.idee_rss)

        # Update concept-rss.xml.
        if len(self.en_list) > 0:
            self._update(self.en_list, gmc.concept_rss)


    def _article_cleanup(self):
        """
        Remove some things from the articles BeautifulSoup model.

        Remove those things, which are not rendered nicely in the
        RSS feed consumer, or which are simply dysfunctional there.

        Changes are applied to the currently processed article
        referenced by self._article_tag

        Consumers tested: GPodder, Liferea, Tidings

        Returns
        -------
        None.

        """
        # peel out sections
        sections = self._article_tag.find_all("section")
        for section in sections:
            section.unwrap()

        # fallback to more common tags
        tag = self._article_tag.find("header")
```

```
        tag.name = "div"
        self._article_tag.name = "div"

        # Remove toc
        nav = self._article_tag.find("nav")
        if nav:
            nav.decompose()

        # Remove footnote-back anchors.
        tags = self._article_tag.find_all("a", class_="footnote-back")
        for tag in tags:
            tag.decompose()

        # Remove footnote-ref anchors, preserve the footnote.
        tags = self._article_tag.find_all("a", class_="footnote-ref")
        for tag in tags:
            suptag = tag.find("sup")
            # make footnotes more visible
            suptag.string.replace_with("(" + suptag.text + ")")
            tag.unwrap()

        # Remove category anchors
        tags = self._article_tag.find_all("a", class_="category")
        for tag in tags:
            tag.decompose()

        # Remove attributes from image preventiong it
        # to be shown in gpodder
        images = self._article_tag.find_all("img")
        for img in images:
            img.attrs = {"src": img.attrs["src"]}

        # Remove id attributes or some tags might not
        # render nicely
        idtags = self._article_tag.find_all(re.compile(r".*"), attrs={
            "id": True})
        for tag in idtags:
            tag.attrs.pop("id")

        # Remove role attributes or some tags might not
        # render nicely
        idtags = self._article_tag.find_all(re.compile(r".*"), attrs={
            "role": True})
        for tag in idtags:
            tag.attrs.pop("role")

        # Remove tabindex attributes not working anyhow in gpodder
        idtags = self._article_tag.find_all(re.compile(r".*"), attrs={
            "tabindex": True})
        for tag in idtags:
            tag.attrs.pop("tabindex")

    def _get_item_tag(self, channel_tag, url, article):
        """
        Find the item tag based on the url information.

        Parameters
        ----------
        channel_tag : xml.dom.minidom.Tag
            The <channel> tag from the minidom document model.
        url : Str
            The url of the article, whose item tag is to be returned.
        article_data : Dict
            Data dictionary of the currently processed article.

        Returns
        -------
        item_tag : xml.dom.minidom.Tag
            The pre-existing or created <item> tag for the currently
            processed article.
        """
        item_tag = None
        tag = None
        links = channel_tag.getElementsByTagName(LINK_TAG)

        for link in links:
```

```
            savedurl = None
            if len(link.childNodes) > 0:
                savedurl = link.childNodes[0].data.strip()
            if url == savedurl:
                tag = link
                break

        if tag:
            item_tag = tag.parentNode
        else:
            item_tag = self._rss_xml.createElement(ITEM_TAG)

            new_tag = self._rss_xml.createElement(TITLE_TAG)
            nodetext = article.get_title()
            textnode = self._rss_xml.createTextNode(nodetext)
            new_tag.appendChild(textnode)
            item_tag.appendChild(new_tag)

            new_tag = self._rss_xml.createElement(LINK_TAG)
            nodetext = url
            textnode = self._rss_xml.createTextNode(nodetext)
            new_tag.appendChild(textnode)
            item_tag.appendChild(new_tag)

            new_tag = self._rss_xml.createElement(PUBDATE_TAG)
            pubdatetime = datetime.datetime.fromisoformat(
                    article.get_published())
            # running your computer on an english locale
            # is helpful for the next line.
            nodetext = pubdatetime.strftime(
                    "%a, %d %b %Y %H:%M:%S +0000")
            textnode = self._rss_xml.createTextNode(nodetext)
            new_tag.appendChild(textnode)
            item_tag.appendChild(new_tag)

            new_tag = self._rss_xml.createElement(GUID_TAG)
            new_tag.setAttribute("isPermaLink", "false")
            item_tag.appendChild(new_tag)

            new_tag = self._rss_xml.createElement(DESCRIPTION_TAG)
            item_tag.appendChild(new_tag)

            new_tag = self._rss_xml.createElement(CONTENT_TAG)
            item_tag.appendChild(new_tag)

        # Processing oldes first, and inserting the items always
        # before the frst childNode, wie get newest first in the XML.
        # To become the sepcification compliant, we finalize by moving
        # all item tags to the end of the channel tag later.
        channel_tag.insertBefore(item_tag,
                                 channel_tag.childNodes[0])

        return item_tag

    def _finalize_channel(self, channel_tag):
        """
        Move the items behind the other channel tags.

        Take care that the number of items does not exceed ITEM_COUNT.
        Update the lastBuildDate.

        Parameters
        ----------
        channel_tag : xml.dom.minidom.Tag
            The <channel> tag from the minidom document model.

        Returns
        -------
        None.

        """
        tags = channel_tag.getElementsByTagName(ITEM_TAG)
        item_count = 0
        for tag in tags:
            if item_count < ITEM_COUNT:
                channel_tag.appendChild(tag)
```

```
                item_count += 1
            else:
                channel_tag.removeChild(tag)

        # change last build date
        # running your computer on an english locale
        # is helpful for this.
        tag = channel_tag.getElementsByTagName(LASTBUILD_TAG)[0]
        pubdatetime = datetime.datetime.fromisoformat(
                self._nowdate)
        nodetext = pubdatetime.strftime(
                "%a, %d %b %Y %H:%M:%S +0000")
        tag.childNodes[0].nodeValue = nodetext

    @staticmethod
    def _remove_empty_lines(xml_doc):
        """Remove empty lines with and without whitespaces."""
        pattern = re.compile(r"^\s*$", re.MULTILINE)
        xml_doc = pattern.sub("", xml_doc)
        pattern = re.compile(r"\n\n", re.MULTILINE)
        xml_doc = pattern.sub("\n", xml_doc)
        return xml_doc

    def _update(self, article_list, rss_path):
        """
        Update the RSS file based on the list of changed or added articles.

        Parameters
        ----------
        article_list : List
            The list of article_data entries of changed or added articles.
            Oldest posts are first in the list.
        rss_path : Path
            The Path to the RSS file.

        Returns
        -------
        None.

        """
        with open(rss_path, 'r', encoding="utf-8") as rss_file:
            self._rss_xml = xml.dom.minidom.parse(rss_file)
        channel_tag = self._rss_xml.getElementsByTagName(CHANNEL_TAG)[0]

        for article in article_list:
            self._article_tag = article.prepare_article_tag_for_rss()

            url = f"{HOST}article/{article.get_urn()}.html"
            item_tag = self._get_item_tag(channel_tag, url, article)

            tag = item_tag.getElementsByTagName(GUID_TAG)[0]
            # Changing the guid on update creates problems with some
            # consumers
            nodetext = url  # + "-" + self._nowdate
            if not tag.hasChildNodes():
                textnode = self._rss_xml.createTextNode(nodetext)
                tag.appendChild(textnode)
            else:
                tag.childNodes[0].nodeValue = nodetext

            tag = item_tag.getElementsByTagName(DESCRIPTION_TAG)[0]
            nodetext = article.get_abstract() + " ..."
            if not tag.hasChildNodes():
                textnode = self._rss_xml.createCDATASection(nodetext)
                tag.appendChild(textnode)
            else:
                tag.childNodes[0].nodeValue = nodetext

            # save the audio uri before the removal
            # of the header tag
            url = None
            tag = self._article_tag.find(AUDIO_TAG)
            if tag:
                url = tag.attrs["src"]

            self._article_cleanup()
```

```
            tag = item_tag.getElementsByTagName(CONTENT_TAG)[0]
            if tag.hasChildNodes():
                tag.removeChild(tag.childNodes[0])
            nodetext = self._article_tag.prettify()
            nodetext = " ".join(nodetext.split())
            # <div style="page-break-before: always;"> </div>
            # inserted in some content to have nice page-breaks in the PDF
            # might be seen as security risk by some consumers
            nodetext = nodetext.replace(
                    '<div style="page-break-before: always;"> </div>',
                    ''
                    )
            textnode = self._rss_xml.createCDATASection(nodetext)
            tag.appendChild(textnode)

            # An update might add or update the audio
            tags = item_tag.getElementsByTagName(ENCLOSURE_TAG)
            tag = None
            if url and len(tags) == 0:
                tag = self._rss_xml.createElement(ENCLOSURE_TAG)
                item_tag.appendChild(tag)
            elif len(tags) > 0 and not url:
                item_tag.removeChild(tags[0])

            # Update enclosure tag
            if tag:
                audio = gmc.audiopath / f"{article.get_urn()}.mp3"
                filelength = 0
                audio.resolve()
                if audio.exists():
                    filelength = audio.stat().st_size
                tag.setAttribute("url", url)
                tag.setAttribute("length", f"{filelength}")
                tag.setAttribute("type", "audio/mpeg")

        self._finalize_channel(channel_tag)

        xml_doc = self._rss_xml.toprettyxml(indent=" ", encoding="utf-8")
        xml_doc = self._remove_empty_lines(xml_doc.decode("utf-8"))

        with open(rss_path, 'w', encoding="utf-8") as rss_file:
            print(xml_doc, file=rss_file)
            rss_file.flush()
            rss_file.close()
            subprocess.run(['git', 'add', rss_path], check=True)
```

Module idxbuilder

The module idxbuilder updates the index files for the English and German web-sites. It's a fixed maximum number of articles featured on these index pages. Old articles containing updates are re-posted in the index to make readers aware of the update or correction.

generator/idxbuilder.py

```
"""
Update the index pages of the webseite.

@author: Frank Siebert
@license: https://creativecommons.org/publicdomain/zero/1.0/deed.en
@date: 2022-03-15

All links provided relative to the /article/ folder

@author: Frank Siebert
"""
import datetime
import subprocess
from gitmsgconstants import GitMsgConstants as gmc
from archive import Archive
```

```
# Number of items to included into the RSS feed
ITEM_COUNT = 15


class IDXBuilder():
    """Manage all changees in the index page."""

    def __init__(self, articles):
        """
        Initialize changelists.

        The information about the changed html pages comes from
        PubMetaData.instance._updates  and
        PubMetaData.instance._deletions  .

        Returns
        -------
        None.

        """
        # information for German page changes on site "Idee".
        self.de_list = []
        # information for English page changes on site "Concept".
        self.en_list = []
        # The time of the update
        self._nowdate = datetime.datetime.now().isoformat()
        # soup of currently processed Index html

        for article in articles:
            site = article.get_site().lower()
            if site in "idee":
                self.de_list.append(article)
            else:
                self.en_list.append(article)


    def update(self):
        """
        Iterate over changes and update respective index pages.

        Returns
        -------
        None.

        """
        for article in self.de_list:
            soup = Archive._update(gmc.idee_index, article,
                                   article_loc="./article/")
            soup = IDXBuilder._limit_entries(soup)
            html_doc = soup.prettify()

            with open(gmc.idee_index, 'w', encoding="utf-8") as index_file:
                print(html_doc, file=index_file)
                index_file.flush()
                index_file.close()
                subprocess.run(['git', 'add', gmc.idee_index], check=True)

        for article_data in self.en_list:
            soup = Archive._update(gmc.concept_index, article_data,
                                   article_loc="./article/")
            soup = IDXBuilder._limit_entries(soup)

            html_doc = soup.prettify()

            with open(gmc.concept_index, 'w', encoding="utf-8") as index_file:
                print(html_doc, file=index_file)
                index_file.flush()
                index_file.close()
                subprocess.run(['git', 'add', gmc.concept_index], check=True)

    @staticmethod
    def _limit_entries(soup):
        tags = soup.find_all("article")
        count = 0
        for tag in tags:
```

```
            if count > ITEM_COUNT:
                tag.decompose()
            else:
                count += 1
        return soup
```

Static Website Content

The static website content has undergone only minor changes since the publishing of "Replacing WordPress" [3] .

Legal Pages

The legal pages moved into the folder legal, which enables me to update them without getting them posted as article.

Cascading Stylesheets

The CSS files for HTML and for PDF have slightly changed.

website/css/fs.css

```
/* ********************************************************************************
 * Frank Siebert's CSS
 +
 * Licence: CC0
 * httpx://frank-siebert.de/article/creative-commons-cc0-1-0-universal.html
 * ********************************************************************************/

:root {
    /* kind of blue */
    --theme-color: #006080;
    /* black on white */
    --theme-text-color: #000000;
    /* white background */
    --theme-background-color: #ffffff;
    /* important right border */
    --important-right-color: #ffffff;
    /* note right border */
    --note-right-color: #999999;
    /* quote left border */
    --quote-left-color: #cccccc;
    /* quote background */
    --quote-background-color: #f9f9f9;
    /* for minor meta information */
    --theme-meta-color: #999999;
    /* Arial and Helvetica exist on my Computer */
    /* --theme-font-family: Arial, Helvetica, Verdana, Tahoma, sans-serif; */
    --theme-font-family: Liberation Sans, sans-serif;
    /* One theme font only, based on the theme font-family */
    --theme-font: 16px/1.4 Liberation Sans, sans-serif;
    /* Improve readability */
    --theme-letter-spacing: normal; /* 0.05em; */
}

html {
    padding: 0px 5px 0px 0px;
    margin: 0;
    border: 0;
    font: var(--theme-font);
    letter-spacing: var(--theme-letter-spacing);
    background-color: lightgray;
}

body {
    width: 100%;
    height: 100%;
```

```
    min-width: 280px;
    max-width:1200px;
    padding: 0 0 0 0;
    margin-top: 0;
    margin-bottom: 0;
    margin-left:auto;
    margin-right:auto;
    border-right: 1px solid var(--theme-color);
    border-left: 1px solid var(--theme-color);

    color: var(--theme-text-color);
    background-color: var(--theme-background-color);
    font-size: 1em;
    word-wrap: break-word;
}

/* ********************************************************************
 * keep the two body elements in sync
 * ********************************************************************/

div.row,
body header,
body main {
    min-height: 100px;
    padding: 5px;

    background-color: var(--theme-background-color);
    background-repeat: no-repeat;
    background-position: top center;
    background-size: auto;
}

body header nav {
    padding: 0 0 0 0;
    /* background: #ddcc99; */
}

/* ********************************************************************
 * The tag <figure> comes with build in padding,
 * but we have to have the same for the article.
 *
 * These styles keep the respectve block elements horizontally alligned.
 *
 * ====  MEDIA SCREEN Variants ====
 * ********************************************************************/

@media screen and (min-width: 641px) {
    body div div, /* yacy search */
    header figure,
    header nav,
    header hr,
    /* main>h3 is used in the archive.html*/
    main article,
    main>h3 {
        display: block;
        margin: 1em 3em 1em 3em;
        /* border-style: dotted;
         * border-width: 2px;     */
    }
    main>h1 {
        display: block;
        margin: 0.6em 1.8em 0.6em 1.8em;
    }
    .searchinput {
        max-width: 600px;
    }
}

@media screen and (max-width: 640px) {
    body div div, /* yacy search */
    header figure,
    header nav,
    header hr,
    /* main>h3 is used in the archive.html*/
    main article,
```

```
    main>h3  {
        display: block;
        margin: 1em 1em 1em 0;
    }
    main>h1 {
        display: block;
        margin: 0.6em 0.6em 0.6em 0.2em;
    }
    .searchinput {
        max-width: 260px;
    }
}

/* ****************************************************************************
 * ==== END OF MEDIA SCREEN Variants ====
 * ****************************************************************************/

/* the main content is the article */
article {
    display: block;
}

/* ****************************************************************************
/* ==== all about headlines ====
 * ****************************************************************************/
/* Ich glaube nicht, dass ich a tags unter die Überschriften legen werde.
 * h1 a, h2 a, h3 a, h4 a, h5 a, h6 a { text-decoration: none; } */
h1, h2, h3, h4, h5, h6
{
    line-height: 1.1;
    margin: 0;
    padding: 1em 0 0.5em 0;

    color: var(--theme-color);
    font-family: var(--theme-font-family);
    font-weight: bold;
}

h1      { font-size: 1.8em; }
h2      { font-size: 1.6em; }
h3      { font-size: 1.4em; }
h4      { font-size: 1.2em; }
h5, h6  { font-size: 1em; }

/* Newspaper Style First Letter of First Paragraph Upper-Case */
article>p:first-of-type::first-letter,
hr+p::first-letter,
h2+p::first-letter,
h3+p::first-letter,
h4+p::first-letter {
    font-family: serif;
    font-size: 1.8em;
    font-weight: bold;
}

/* ****************************************************************************
 * ==== Article Header ====
 * - h1 headline
 * - address information
 * - page qr-code
 * - licence information
 * - audio player
 * ****************************************************************************/

article header {
    min-height: 0
}

article header h1 {
    padding: 0 0 0.2em 0;
}

article header div {
    color: var(--theme-meta-color);
    font-size: 0.8em;
    padding: 0 0 1em 0;
```

```
}


/* The browser decided, that address gets rendered italic,
 * but we do not want this */
article header time,
article header address {
    padding-right: 20px;
    display: inline;
    font: var(--theme-font);
    font-size:inherit
}

/* ***************************************************************************
 * ==== Article Block Elements
 * ***************************************************************************/
p {
    margin: 0;
    font-size: 1em;
    padding: 0 0 1em 0;
}

p:last-child
{
    padding-bottom: 0;
}

table {
    padding: 10px 20px 10px 20px;
  overflow-wrap: anywhere;
    display: block;
}

table th {
    background: #ddd;
    border-right: 1px solid #fff;
    padding: 10px 20px;
}

table tr th:last-child {
    border-right: 1px solid #ddd;
}

table td {
    padding: 5px 20px;
    border: 1px solid #ddd;
}

table caption { font: var(--theme-font); font-size: 0.8em;
color: var(--theme-color); font-style: italic; padding: 2px;
caption-side:bottom; padding: 0 20px 20px 20px }

/* ***************************************************************************
 * ==== Figures in the header and in the article ====
 * ***************************************************************************/

figure img    { width: 100%; height: auto; }
figure audio  { width: 50%;  height: auto; min-height:2em;}
header figure figcaption  { font: var(--theme-font); font-size: 1em;
color: var(--theme-color); font-weight: bold}
article figure            { margin: 10px }
figure figcaption { font: var(--theme-font); font-size: 0.8em;
color: var(--theme-color); font-style: italic; padding: 2px;}

article header div figure           { display: Inline; }
article header div figure img       { width: 50px; }
article header div figure figcaption { display: Inline; width: 150px }
article header div figure audio { margin: .5em .5em .5em .5em; }

/* ***************************************************************************
 * ==== Navigation in the header ====
 * ***************************************************************************/

header>nav>a {
    font-size: 1.2em;
    padding: 0 0.5em 0 0;
```

```
    display: inline-grid;
    grid-template-columns: 30px auto auto auto;
}

header>nav>a>img {
    width: 24px;
    vertical-align: sub;
}

header>nav>form {
    display: inline;
    padding: 0 0.5em 0 0;
    margin: 0 0 0 0;
}

header>nav>form>input{
    font: var(--theme-font);
    letter-spacing: var(--theme-letter-spacing);
    font-size: 1em;
    vertical-align: super;
    padding: 0 0 0 0;
    margin: 0 0 0 0;
    border-color: var(--theme-color);
}

/* context break is meta information */
hr {
    height:1px;
    border-width:0;
    background-color: var(--theme-meta-color);
}

/* ********************************************************************
 *    inline HTML TAGS
 * ********************************************************************/

pre {
    background: #f5f5f5;
    border: 1px solid #ddd;
    padding: 10px;
    text-shadow: 1px 1px rgba(255, 255, 255, 0.4);
    font-size: 0.8em;
    line-height: 1.25;
    margin: 0 0 1em 0;
    overflow: auto;
}

sup, sub {
    font-size: 0.75em;
    height: 0;
    line-height: 0;
    position: relative;
    vertical-align: baseline;
}

sup {
    bottom: 1ex;
}

sub {
    top: 1ex;
}

small {
    font-size: 0.75em
}


/* ********************************************************************
 * ==== Navigation and their targets ====
 * ********************************************************************/

*:target {
    border-bottom: 0.3em solid var(--theme-color);
}
```

```
a {
    text-decoration: none;
    font: var(--theme-font);
    font-size: 1em;
    font-weight: bold;
    color: var(--theme-color);
    border-width: 0 0 0 0;
    border-style: none;
}
a:link        { color: var(--theme-color);       }
a:visited     { color: var(--theme-text-color);  }

/* figure:has(a:focus), */ /* Wait for CSS 4 */
a:focus,
a:hover /* ,
a:active */ {
color: var(--theme-background-color);
background-color: var(--theme-color);
outline: none;
}

figure a:focus,
figure a:hover {
    color: var(--theme-background-color);
    background-color: var(--theme-color);
    outline: none;
    border: none;
}

header>div>a:focus,
header>div>a:hover {
    background-color: var(--theme-background-color);
    color: var(--theme-color);
    outline: none;
    border: none;
}

a.category    { visibility:visible } /*  hidden; */


/* ****************************************************************************
 * ==== YaCy Search ====
 * ****************************************************************************/

p.urlinfo :nth-child(2),
p.urlinfo :nth-child(3),
p.urlinfo :nth-child(4),
p.urlinfo :nth-child(5),
p.urlinfo :nth-child(6),
p.urlinfo :nth-child(7),
.favicon,
.navbar,
.starter-template,
.hidden,
.urlactions,
.input-group-btn,
.sidebar,
#datehistogram,
#api {
    display: none;
}

div {
    min-height: 10px;
    margin: 0 0 0 0;
    padding: 0 0 0 0;
}

span#resNav ul li {
    display: inline;
    font-size: 1.4em;
}

.searchinput {
    font: var(--theme-font);
    letter-spacing: var(--theme-letter-spacing);
```

```
    font-size: 1em;
    border-color: var(--theme-color);
    outline: 5px solid var(--theme-meta-color);
}

.linktitle,
.pagination {
    font-size: 1.4em;
    border-top: 2px solid var(--theme-meta-color);
}

/* ****************************************************************************
 * notes (update and correction notes)
 * ***************************************************************************/
.important {
  border-right: 2px solid var(--important-right-color);
  margin: 0 2em 0.5em;
  padding: 0.5em 10px;
}

/* ****************************************************************************
 * notes (update and correction notes)
 * ***************************************************************************/
.note {
  border-right: 2px solid var(--note-right-color);
  margin: 0 2em 0.5em;
  padding: 0.5em 10px;
}

/* ****************************************************************************
 * blockquote
 * ***************************************************************************/
blockquote {
  font-style:italic;
  background: var(--quote-background-color);
  border-left: 10px solid var(--quote-left-color);
  margin: 0 2em 0.5em;
  padding: 0.5em 10px;
}

/* ****************************************************************************
 * ==== syntaxhighlight ====
 * CSS as created in the html style-element by WeasyOrint for syntaxhighlight
 * Changes for the print version need to be applied in fspdf.css
 * Changes for the browser version need to be applied at the end of this file.
 * ***************************************************************************/

code{white-space: nowrap;}
span.smallcaps{font-variant: small-caps;}
span.underline{text-decoration: underline;}
div.column{display: inline-block; vertical-align: top; width: 50%;}
div.hanging-indent{margin-left: 1.5em; text-indent: -1.5em;}
ul.task-list{list-style: none;}
pre > code.sourceCode { white-space: pre; position: relative; }
pre > code.sourceCode > span { display: inline-block; line-height: 1.25; }
pre > code.sourceCode > span:empty { height: 1.2em; }
code.sourceCode > span { color: inherit; text-decoration: inherit; }
div.sourceCode { margin: 1em 0; }
pre.sourceCode { margin: 0; }

@media screen {
    div.sourceCode { overflow: auto; }
}


@media print {
    pre > code.sourceCode { white-space: pre-wrap; }
    pre > code.sourceCode > span { text-indent: -5em; padding-left: 5em; }
}

pre.numberSource code
  { counter-reset: source-line 0; }
pre.numberSource code > span
  { position: relative; left: -4em; counter-increment: source-line; }
pre.numberSource code > span > a:first-child::before
  { content: counter(source-line);
```

```
    position: relative; left: -1em; text-align: right; vertical-align: baseline;
    border: none; display: inline-block;
    -webkit-touch-callout: none; -webkit-user-select: none;
    -khtml-user-select: none; -moz-user-select: none;
    -ms-user-select: none; user-select: none;
    padding: 0 4px; width: 4em;
    color: #aaaaaa;
  }
pre.numberSource { margin-left: 3em; border-left: 1px solid #aaaaaa;
    padding-left: 4px; }
div.sourceCode
  {   }

@media screen {
    pre > code.sourceCode > span > a:first-child::before {
        text-decoration: underline; }
}

code span.al { color: #ff0000; font-weight: bold; } /* Alert */
code span.an { color: #60a0b0; font-weight: bold; font-style: italic;
    } /* Annotation */
code span.at { color: #7d9029; } /* Attribute */
code span.bn { color: #40a070; } /* BaseN */
code span.bu { } /* BuiltIn */
code span.cf { color: #007020; font-weight: bold; } /* ControlFlow */
code span.ch { color: #4070a0; } /* Char */
code span.cn { color: #880000; } /* Constant */
code span.co { color: #60a0b0; font-style: italic; } /* Comment */
code span.cv { color: #60a0b0; font-weight: bold; font-style: italic;
    } /* CommentVar */
code span.do { color: #ba2121; font-style: italic; } /* Documentation */
code span.dt { color: #902000; } /* DataType */
code span.dv { color: #40a070; } /* DecVal */
code span.er { color: #ff0000; font-weight: bold; } /* Error */
code span.ex { } /* Extension */
code span.fl { color: #40a070; } /* Float */
code span.fu { color: #06287e; } /* Function */
code span.im { } /* Import */
code span.in { color: #60a0b0; font-weight: bold; font-style: italic;
    } /* Information */
code span.kw { color: #007020; font-weight: bold; } /* Keyword */
code span.op { color: #666666; } /* Operator */
code span.ot { color: #007020; } /* Other */
code span.pp { color: #bc7a00; } /* Preprocessor */
code span.sc { color: #4070a0; } /* SpecialChar */
code span.ss { color: #bb6688; } /* SpecialString */
code span.st { color: #4070a0; } /* String */
code span.va { color: #19177c; } /* Variable */
code span.vs { color: #4070a0; } /* VerbatimString */
code span.wa { color: #60a0b0; font-weight: bold; font-style: italic;
    } /* Warning */

/* ****************************************************************************
 * ==== syntaxhighlight ====
 * Own Part
 * ****************************************************************************/
pre.sourceCode, pre.mysql, pre.nginx {
    width: 90ch; /* classic terminal width for code sections is 80 */
}
```

website/css/fspdf.css

```
/* ****************************************************************************
 * Frank Siebert's PDF CSS
 +
 * Licence: CC0
 * httpx://frank-siebert.de/article/creative-commons-cc0-1-0-universal.html
 * ****************************************************************************/

html {
    font-family: Liberation Sans, sans-serif !important;
    font: 12px/1.4 Liberation Sans, sans-serif !important;
```

```
    background-color: #ffffff !important;
}

@page {
  size: A4; /* Change from the default size of A4 */
  margin: 1.5cm; /* Set margin on each page */

  @top-right {
    content: counter(page);
    color: #006080;
    font-size: 1.2em;
  }

  @top-left {
    content:  string(pageheader);
    color: #006080;
    font-size: 1.2em;
  }
}

header h1 {
  string-set: pageheader content();
}

article header div figure img       { width: 150px !important; }
article figure a img { width: 70% !important; }

/* ****************************************************************************
 * ==== syntaxhighlight ====
 * ****************************************************************************/

/* Allow only intentional line breaks in source code */
pre > code.sourceCode > span {
    white-space: pre !important;
}
pre > code.sourceCode > span > span {
    white-space: pre !important;
}

pre.sourceCode, pre.mysql, pre.nginx {
    width: 90ch !important; /* classic terminal width for code sections is 80 */
}

body {
  border: none !important;
}
```

Server Setup

The server setup has not changed, look into the article "Replacing Wordpress" for the details of the setup.

Footnotes

1. **Vim Plugin for Web Publishing** ; Frank Siebert; concept; 2026-03-09 ↑
   https://idee.frank-siebert.de/article/vim-plugin-for-web-publishing.html
2. **Replacing WordPress - Idee Website Server Setup** ; Frank Siebert; Concept; 2022-03-17 ↑
   https://idee.frank-siebert.de/article/replacing-wordpress.html#idee_website_server_setup
3. **Replacing WordPress** ; Frank Siebert; Concept; 2022-03-17 ↑
   https://idee.frank-siebert.de/article/replacing-wordpress.html