RSS Feed Generator für Podcasts

Alwin Zimmermann

May 18, 2025

1 Ziel

Das Ziel des Generators ist es, dass ein sogenannter Podcatcher den RSS Feed unseres Podcasts einlesen kann. Der RSS Feed muss hierfür von unserem Server als .XML (oder .rss) bereitgestellt werden. Bei einem RSS Feed gibt es einige wichtige Punkte zu beachten, die für den Aufbau eines syntaktisch richtigen Feeds beachtet werden müssen. Eine XML-Datei hat eine Baumstruktur, im Kopf stehen Infos wie: XML-Version, Encoding, RSS-Version.

2 Struktur des RSS Feeds

Nach dem Kopf steht die Channel-Sektion, in der der Channel die wichtigsten Werte erhält. Diese wären:

- Titel
- Link zur Webseite
- Beschreibung
- Sprache
- Veröffentlichungsdatum
- Info, wann der Feed das letzte Mal erstellt/aktualisiert wurde

Ab diesem Bereich beginnen die Items, in denen die Infos über die neuen Podcast-Folgen hinzugefügt werden können. Jedes Item (jede Folge) hat:

- Titel
- Link zur Episode
- Beschreibung
- Enclosure URL
- Veröffentlichungsdatum

Nach allen Items wird der Channel geschlossen und der RSS Feed beendet.

3 Probleme

Das Paket `feedparser` (`https://github.com/kurtmckee/feedparser`) hat leider das Problem, dass es die sogenannte Enclosure URL nicht parsen kann. Diese ist jedoch essenziell für den Podcatcher, um die Datei finden zu können (Issue: #285 enclosure not parsed). Das Problem besteht nicht beim ersten Erstellen des Feeds, sondern erst beim Wiedereinlesen des Feeds. Eine mögliche Lösung hätte sein können, die Infos alle redundant zu speichern, jedoch habe ich diese Lösung schnell verworfen, da sie mit einem unverhältnismäßigen Aufwand verbunden wäre. Eine weitere Lösung könnte das Modul RSS in Ruby sein, jedoch wollte ich die Problematik in Python lösen, da die meisten Teile unseres Projekts bereits in Python geschrieben waren und ich selber bisher noch kein Ruby geschrieben habe. Deshalb habe ich mich für das Paket `lxml` entschieden, welches ein generisches XML-Bearbeitungstool ist (link: `https://github.com/lxml/lxml`). Dieses ist zwar nicht speziell für RSS entworfen, sollte jedoch manuell in der Lage sein, den Feed entsprechend anzupassen.

```python
def load_existing_feed(self):
    """Lädt den bestehenden RSS-Feed, falls vorhanden."""
    if os.path.exists(self.feed_file_path):
        feed = feedparser.parse(self.feed_file_path)
        for entry in feed.entries:
            item = {
                'title': entry.title,
                'description': entry.description,
                'date': entry.published if 'published' in entry else datetime.now().isoformat(),
                'enclosure_url': entry.enclosure.href if 'enclosure' in entry else None,
                'enclosure_type': entry.enclosure.type if 'enclosure' in entry else None,
                'enclosure_length': entry.enclosure.length if 'enclosure' in entry else None
            }
            self.items.append(item)
            print(self)
```

Figure 1: Feedparser Bug

4 Das Script

Das Script ist ein einfaches Python-Modul, das eine Funktion zum Hinzufügen von Podcasts in den Channel ermöglicht. Wichtig ist natürlich das Importieren der benötigten Python-Libraries.

```python
import os
import argparse
import datetime
from lxml import etree
```

Figure 2: Imports

Als Input-Werte benötigt es lediglich:

- einen Titel als String
- eine URL zu der Folge, die hinzugefügt werden möchte
- eine Beschreibung der Folge

Das Veröffentlichungsdatum wird als aktuelles Datum gewählt, da das Script automatisiert ausgeführt werden soll. Zudem wird der Audio-Typ auf MP3 festgelegt, da wir das FLAC-Format von Anfang an verworfen haben. Mithilfe von `lxml` wird nun ein neues `<item>` Objekt hinzugefügt mit den angegebenen Werten.

```
# Erstelle ein neues Item
new_item = etree.Element('item')
etree.SubElement(new_item, 'title').text = title
etree.SubElement(new_item, 'description').text = description
etree.SubElement(new_item, 'pubDate').text = new_episode_pubDate
enclosure = etree.SubElement(new_item, 'enclosure')
enclosure.set('url', url)
enclosure.set('type', 'audio/mpeg')
```

Figure 3: New Item

Auch wird im Kopf das `lastBuildDate` auf den aktuellen Moment abgeändert. Zuletzt wird der neue XML-Feed dann in einen Ordner mit dem Namen `rss` abgespeichert und kann vom Server publiziert werden.

```
def add_episode_to_podcast(title: str, url: str, description: str):
    # Verzeichnis und Dateiname
    directory = './rss'  # Aktuelles Verzeichnis mit dem Unterordner 'rss'
    filename = 'podcast.xml'  # Name der RSS-Datei

    # Vollständiger Pfad zur Datei
    file_path = os.path.join(directory, filename)

    # RSS-Feed laden
    with open(file_path, 'rb') as f:
        feed_content = f.read()

    # XML-Parser initialisieren
    root = etree.fromstring(feed_content)

    # Neue Episode hinzufügen
    new_episode_pubDate = str(datetime.datetime.now())

    # Erstelle ein neues Item
    new_item = etree.Element('item')
    etree.SubElement(new_item, 'title').text = title
    etree.SubElement(new_item, 'description').text = description
    etree.SubElement(new_item, 'pubDate').text = new_episode_pubDate
    enclosure = etree.SubElement(new_item, 'enclosure')
    enclosure.set('url', url)
    enclosure.set('type', 'audio/mpeg')

    # Füge das neue Item zum Channel hinzu
    channel = root.find('channel')
    channel.append(new_item)

    # Aktualisiere lastBuildDate
    last_build_date = channel.find('lastBuildDate')
    last_build_date.text = str(datetime.datetime.now())

    # Speichern des aktualisierten Feeds mit Zeilenumbrüchen und Einrückungen
    with open(file_path, 'wb') as f:
        f.write(etree.tostring(root, pretty_print=True, xml_declaration=True, encoding='UTF-8'))

    print(f'Die neue Episode "{title}" wurde erfolgreich zu {filename} hinzugefügt.')
```

Figure 4: Script

1

[1]Ich erkläre hiermit, dass ich bei der Erstellung dieses Projektes Unterstützung von ChatGPT, einem KI-gestützten Sprachmodell von OpenAI, in Anspruch genommen habe. Alle Inhalte wurden von mir überprüft und bearbeitet.