Perfetto 🙌 allora ti preparo un pacchetto semplice e pratico:

**📌 1. Template Statico (template\_static.py)**

Per siti “semplici” dove il codice HTML contiene già tutte le notizie.

import requests

from bs4 import BeautifulSoup

from feedgenerator import Rss201rev2Feed

from datetime import datetime

def generate\_feed(url, output\_file):

# Richiesta HTTP classica

response = requests.get(url)

response.raise\_for\_status()

soup = BeautifulSoup(response.text, "html.parser")

# Feed di base

feed = Rss201rev2Feed(

title="Feed RSS simulato",

link=url,

description="Notizie generate automaticamente",

language="it"

)

# 🔧 Adatta qui ai selettori reali del sito

articles = soup.select("h3.card-title")

for art in articles:

a = art.find("a")

if not a:

continue

title = a.get\_text(strip=True)

link = a["href"]

# Se i link sono relativi:

if link.startswith("/"):

link = url.rstrip("/") + link

# Data (se disponibile)

date\_tag = art.find\_next("span", class\_="data")

pub\_date = None

if date\_tag:

try:

pub\_date = datetime.strptime(

date\_tag.get\_text(strip=True), "%A, %d %B %Y"

)

except Exception:

pass

# Descrizione opzionale

description = ""

feed.add\_item(

title=title,

link=link,

description=description,

pubdate=pub\_date

)

# Salvataggio feed

with open(output\_file, "w", encoding="utf-8") as f:

feed.write(f, "utf-8")

**📌 2. Template Dinamico (template\_playwright.py)**

Per siti che mostrano notizie solo dopo caricamento dinamico (es. Posina, Schio).

import asyncio

from playwright.async\_api import async\_playwright

from bs4 import BeautifulSoup

from feedgenerator import Rss201rev2Feed

from datetime import datetime

async def generate\_feed(url, output\_file):

async with async\_playwright() as p:

browser = await p.firefox.launch(headless=True)

page = await browser.new\_page()

await page.goto(url, wait\_until="networkidle")

# Prendi l'HTML reso dal browser

html = await page.content()

await browser.close()

soup = BeautifulSoup(html, "html.parser")

# Feed di base

feed = Rss201rev2Feed(

title="Feed RSS simulato (dinamico)",

link=url,

description="Notizie generate automaticamente",

language="it"

)

# 🔧 Adatta qui ai selettori reali del sito

articles = soup.select("h3.card-title")

for art in articles:

a = art.find("a")

if not a:

continue

title = a.get\_text(strip=True)

link = a["href"]

if link.startswith("/"):

link = url.rstrip("/") + link

# Data

date\_tag = art.find\_next("span", class\_="data")

pub\_date = None

if date\_tag:

try:

pub\_date = datetime.strptime(

date\_tag.get\_text(strip=True), "%A, %d %B %Y"

)

except Exception:

pass

description = ""

feed.add\_item(

title=title,

link=link,

description=description,

pubdate=pub\_date

)

with open(output\_file, "w", encoding="utf-8") as f:

feed.write(f, "utf-8")

if \_\_name\_\_ == "\_\_main\_\_":

import sys

if len(sys.argv) != 3:

print("Uso: python template\_playwright.py <url> <output.xml>")

else:

asyncio.run(generate\_feed(sys.argv[1], sys.argv[2]))

**📌 3. Nuova Checklist Aggiornata**

1. **Analisi del sito**
   * Apri la pagina delle notizie.
   * Vai su **Ispeziona (F12)** → **Elements**.
   * Vedi già <h3 class="card-title"> con i link dentro l’HTML?
     + **Sì** → modello **statico**.
     + **No** (vedi <script>, contenitori vuoti, o notizie che appaiono solo dopo) → modello **Playwright**.
2. **Adattamento selettori**
   * Titolo → di solito h3.card-title a o simile.
   * Data → span.data o simile.
   * Link → attenzione se relativi (/path → aggiungere dominio).
   * Descrizione → se disponibile (paragrafi, abstract ecc.).
3. **Test locale**
   * Statico:
   * python template\_static.py https://sito.it/Novita feeds/test.xml
   * Playwright:
   * python template\_playwright.py https://sito.it/Novita feeds/test.xml
4. **Integrazione nel progetto**
   * Copia il modello scelto in comuni/nomecomune.py.
   * Aggiorna sites\_list.json con "script": "nomecomune".
   * Lancia python generate\_all.py.

👉 Così avrai un metodo chiaro: prima decidi **statico o dinamico**, poi applichi il modello giusto.