UNIVERSITÀ
DEGLI STUDI
DI MILANO

Trasformazioni di Formato

Paolo Ceravolo

paolo.ceravolo@unimi.it

Editoria Digitale

SESAR LAB

UNIVERSITÀ
DEGLI STUDI
DI MILANO

TRASFORMAZIONI

UNIVERSITÀ DEGLI STUDI DI MILANO

- La gestione di un workflow implica la gestione di passaggi di trasformazione da un formato a un altro
- Idealmente si parte da formati semplici e portabili per poter ottenere molteplici formati di destinazione
- Insieme al contenuto testuale è necessario gestire file multimediali e riferimenti ad altri documenti (include, citazioni, link)
- Le trasformazioni avvengono attraverso la rappresentazione di un formato in un modello intermedio (spesso resta solo nella memoria del programma di trasformazione) che poi può essere trasformato in un formato di destinazione.
  - Diventa quindi essenziale conoscere la struttura e l'espressività del modello intermedio di riferimento
  - Tipicamente i formati usati per i sorgenti dei contenuti sono meno espressivi o al più espressivi come il formato intermedio
  - I formati di destinazione potranno essere più espressivi del formato intermedio e potranno essere quindi necessari degli interventi manuali (es. formattazione tabelle)

Pandoc

PANDOC

UNIVERSITÀ DEGLI STUDI DI MILANO

- Pandoc è una libreria scritta in Haskell per la conversione di documenti da un formato di origine a formato di destinazione
- Pandoc è in grado di convertire numerosi formati di markup e di elaborazione testi, tra cui vari tipi di Markdown, HTML, LaTeX, ePub e Word docx
- La versione estesa di Markdown usata da Pandoc include una sintassi per le tabelle, elenchi di definizioni, blocchi di metadati, note a piè di pagina, citazioni, formule matematiche e altro

PANDOC ORGANIZZAZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Pandoc ha un design modulare
  - un insieme di *reader*, che analizzano un dato formato e producono una rappresentazione astratta del documento (un albero sintattico astratto o AST)
  - un insieme di *writer*, che convertono questa rappresentazione astratta in un formato di destinazione
- AST è meno espressivo di molti formati di marcatura, questo implica che le trasformazioni conserveranno la struttura dei documenti ma non certi dettagli di formattazione come i margini
- Gli utenti possono anche comporre degli script detti pandoc filters per modificare l'AST intermedio di un processo di trasformazione

PANDOC COME USARLO

UNIVERSITÀ DEGLI STUDI DI MILANO

- Pandoc può essere stato in diversi modi
  - Da linea di comando

```
pandoc -o output.html input.txt
pandoc -s -o output.html input.txt
pandoc -f markdown -t latex hello.txt
```

  - Attraverso interfaccia online https://pandoc.org/try/
  - Attraverso Python https://pypi.org/project/pandoc/
  - Interfacciandosi attraverso la sua API https://hackage.haskell.org/package/pandoc

PANDOC INSTALLAZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Per installare Pandoc è possibile usare diversi strumenti

  https://pandoc.org/installing.html

  - Installer forniti dalla distribuzione
  - Attraverso diversi sitemi software management come Chocolatey, Brew, Crew, …
  - Attraverso Docker
- Per un funzionamento completo, oltre alla libreria è necessario installare
  - Python per l'esecuzione dei Pandoc filter
  - un motore LaTex (MiKTeX BasicTeX MacTeX TeX Live) per la compilazione di file LaTex
- Può essere necessario installare ulteriori estensioni per la gestione di particolari formati multimediali

PANDOC COME USARLO

UNIVERSITÀ DEGLI STUDI DI MILANO

- Esempio 1: Conversione di base

  Pandoc consente di convertire documenti da un formato all'altro

  ```
  pandoc input.md -o output.pdf
  ```

- Esempio 2: Intestazione del documento

  Puoi personalizzare il layout, ad esempio, con intestazioni e piè di pagina

  ```
  pandoc input.md -H header.tex -o output.pdf
  ```

- Esempio 3: Applicare un foglio di stile

  È possibile migliorare l'aspetto dei documenti generati da Pandoc applicando un foglio di stile CSS. È necessario usare l’opzione -s per forzare la generazione del head

  ```
  pandoc -s input.md -o output.html --css stile.css
  ```

- Esempio 4: Unire documenti

  Pandoc può unire più documenti in uno

  ```
  pandoc doc1.md doc2.md -o unione.pdf
  ```

PANDOC COME USARLO

UNIVERSITÀ DEGLI STUDI DI MILANO

- Esempio 5: Associazione di Metadati

  È possibile utilizzare un file YAML esterno per specificare metadati da associare al documento

```
pandoc input.md -o output.pdf --metadata-file metadati.yaml
```

```
---
title: "Il mio documento"
author:
  - Nome Autore
  - Autore Aggiuntivo
date: 2023-11-02
keywords:
  - Pandoc
 - Metadati
  - YAML
abstract: "Questo è un esempio di documento Markdown con metadati in formato YAML."
---
```

PANDOC AST

UNIVERSITÀ DEGLI STUDI DI MILANO

- *L'Abstract Syntax Tree* di Pandoc modella ogni documento come un albero di elementi
  - Ogni documento appartiene a un tipo, il meta-modello del documento, che definisce come si strutturano gli elementi all'interno del documento
  - Ogni elemento ha un tipo definito, ad esempio *paragrafo*, *immagine*, *nota*, *link*, ecc.
- Se si desidera analizzare, creare o trasformare documenti, è necessaria una certa conoscenza del meta-modello

  https://hackage.haskell.org/package/pandoc-types-1.22.2.1/docs/Text-Pandoc-Definition.html

PANDOC FILTRI

UNIVERSITÀ DEGLI STUDI DI MILANO

- Un filtro è un programma che modifica lo AST tra un reader e un writer

```
INPUT --reader--> AST --filter--> AST --writer-->
OUTPUT
```

- È possibile scrivere questi programmi utilizzando LUA oppure JSON

```
INPUT --reader--> JSON-formatted AST --filter-->
JSON-formatted AST --writer--> OUTPUT
```

- Ad esempio in Haskell è possibile

```
#!/usr/bin/env runhaskell
-- behead.hs
import Text.Pandoc.JSON

main :: IO ()
main = toJSONFilter behead

behead :: Block -> Block
behead (Header n _ xs) | n >= 2 = Para [Emph xs]
behead x = x
```

PANDOC FILTRI

UNIVERSITÀ DEGLI STUDI DI MILANO

- Da riga di comando un filtro può essere richiamato con l'apposita opzione seguita dal nome dello script che si richiede di eseguire

```
pandoc input.md --filter filtro.py -o output.pdf
```

- I file passati al parametro `--filtro/-F` devono essere eseguibili. Le estensioni del file indicano direttamente l'interprete da utilizzare

```
.py python
.hs runhaskell
.pl perl
.rb ruby
.php php
.js nodo
.r Rscript
```

PANDOC FILTRI - PANFLUTE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Panflute è una libreria per la creazione di filtri Pandoc in Python progettata per funzionare in modo nativo con Pandoc
  - La sua sintassi chiara, la documentazione dettagliata e il supporto per una vasta gamma di elementi Pandoc la rendono una scelta popolare per coloro che desiderano personalizzare il processo di conversione dei documenti con Pandoc
- Assumendo di aver installato Python si può installare Panlflute con

  ```
  pip install panflute
  ```

- Richiamando genericamente Panflute con `pandoc input.md -F -o output.md` è anche possibile eseguire una lista di script andando a indicare la loro collocazione direttamente nei metadati YAML di un file Markdown

  ```
  ---
  title: Some title
  panflute-filters: [remove-tables, include]
  panflute-path: 'panflute/filters'
  ...

  Lorem ipsum
  ```

PANDOC FILTRI – PANFLUTE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Un semplice filtro può essere il seguente:

```
1. #!/usr/bin/env python3
2. from panflute import *
3.
4. def action(elem, doc):
5.     if isinstance(elem, Header):
6.         elem.level = 1
7.
8. def main(doc=None):
9.     return run_filter(action, doc=doc)
10.
11. if __name__ == '__main__':
12.     main()
```

- Alla riga 4 abbiamo la dichiarazione di una funzione di trasformazione
- Alla riga 8 è dichiarata la funzione main (associata un documento parametro per il documento che viene passato da Pandoc oppure di default è pari a None) che esegue il filtro
- Alla riga 11 si verifica che lo script sia eseguito direttamente e non importato come modulo in un altro script

PANDOC METADATI

UNIVERSITÀ DEGLI STUDI DI MILANO

- Pandoc utilizza metadati in formato YAML per consentire agli utenti di fornire informazioni aggiuntive sulla struttura e la formattazione del documento di input
- I metadati YAML vengono di solito inseriti all'inizio del documento Markdown

```
---
title: Il mio documento
author:
   - Primo autore
   - Secondo autore
date: /today
---
# Contenuto del documento in formato Markdown
```

- Si può gestire la conversione di YAML usando una libreria Python come PyYAML

```
pip install pyyaml
```

PANDOC METADATI

UNIVERSITÀ DEGLI STUDI DI MILANO

- PyYAML consente di leggere e scrivere metadati YAML in modo semplice e intuitivo e di trasformali in altri formati
- Caricato un file YAML lo mette in memoria come dictionary e permette poi di salvarlo in un nuovo formato

```
import yaml

# Leggere il file YAML
with open('file.yaml', 'r') as yaml_file:
    yaml_data = yaml.safe_load(yaml_file)

# Generare un file di testo con i dati letti
with open('output.txt', 'w') as txt_file:
    for key, value in yaml_data.items():
        txt_file.write(f"{key}: {value}\n")
```

PANDOC METADATI

UNIVERSITÀ DEGLI STUDI DI MILANO

- Quando oltre a modificare il formato vogliamo modificare la struttura di un insieme di metadati è utile avere a disposizione uno strumento di parsing
  - Gli strumenti più completi sono quelli che lavorano con JSON come ad esempio la libreria standard di Python come `json`

```
import json
# Creare un dizionario Python
data = {
    "name": "John Doe",
    "age": 30,
    "city": "New York"
}
# Convertire il dizionario in una stringa JSON
json_string = json.dumps(data)
print("JSON String:", json_string)
# Convertire una stringa JSON in un dizionario Python
decoded_data = json.loads(json_string)
print("Decoded Data:", decoded_data)
```

PANDOC METADATI

UNIVERSITÀ DEGLI STUDI DI MILANO

- Attraverso la libreria `json` è possibile, aggiungere, rimuovere, spostare e rinominare elementi
  - Aggiungere:

```
import json
# open file in read-mode
with open('original.json', 'r') as file:
    # read JSON data
    data = json.load(file)
    # add field
    data["abc"]["mno"] = 3
```

  - Rimuovere:

```
    # remove
    data.pop("jkl")
```

PANDOC METADATI

UNIVERSITÀ DEGLI STUDI DI MILANO

- Attraverso la libreria `json` è possibile, aggiungere, rimuovere, spostare e rinominare elementi
  - Spostare:

```
# move
field = data["abc"].pop("ghi")
data["ghi"] = field
```

  - Rinominare:

```
# rename
field = data.pop("abc")
data["pqr"] = field
```

```
# save in file
newData = json.dumps(data, indent=4)
# open
with open('modified.json', 'w') as file:
    # write
    file.write(newData)
```

PANDOC METADATI

UNIVERSITÀ DEGLI STUDI DI MILANO

- I metadati possono essere utilizzati anche per parametrizzare una trasformazione utilizzando dei placeholder all'interno di un documento template
- In questo caso solitamente i metadati sono caricati da un file esterno che può essere in formato YAML o JSON

```
{
  "title": "Esempio di Documento HTML",
  "article": "Titolo Articolo",
}
```

- Il template conterrà alcuni palceholder

```
<html lang="en">
<body>
    <header><h1>$title$</h1></header>
    <article>
        <header>$article$</header><
        <!-- Contenuto del documento qui -->
    </article>
</body>
</html>
```

PANDOC METADATI

UNIVERSITÀ DEGLI STUDI DI MILANO

- Possiamo quindi eseguire la trasformazione utilizzando questo comando

```
pandoc --metadata-file=metadata.json --template=template.html -o output.html
```

- Utilizzando JSON in questo modo Pandoc può diventare un sistema di Content Management System vero e proprio
  - Mettendolo al servizio di un framework Web in grado di eseguire chiamate HTTP, come Flask o Express, potrebbe essere un valido strumento di templating

```
from flask import Flask, request, jsonify
import subprocess
# Eseguire Pandoc per convertire il Markdown in HTML
        subprocess.run(['pandoc', '--from=markdown', '--to=html', 'input.md', '-
o', `output.html`])
```

___

```
const express = require('express');
const bodyParser = require('body-parser');
const { exec } = require('child_process');
// Esegui Pandoc per convertire il Markdown in HTML
        exec('pandoc --from=markdown --to=html input.md -o output.html', (error,
stdout, stderr) => {
  // Istruzioni
 });
```

PANDOC – WORKFLOW DI ESECUZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Un workflow di esecuzione di documenti MD dovrebbe idealmente permettere di:

1. Editare in modo collaborativo
2. Gestire il versioning
3. Supportare le funzioni richieste dal progetto
4. Configurare agevolmente le trasformazione
5. Automatizzare la trasformazione per collezioni di file

PANDOC – WORKFLOW DI ESECUZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Un workflow di esecuzione di documenti MD dovrebbe idealmente permettere di:

1. Editare in modo collaborativo
2. Gestire il versioning
3. Supportare le funzioni richieste dal progetto
4. Configurare agevolmente le trasformazione
5. Automatizzare la trasformazione per collezioni di file

} **Git**

} **Filtri e opzioni**

} **Workflow Manager**

PANDOC – WORKFLOW MANAGER

UNIVERSITÀ DEGLI STUDI DI MILANO

- In principio è possibile collegare Pandoc con qualsiasi libreria di scripting che opera come workflow manger
    - Bash scripts
- Esistono tuttavia alcuni progetti dedicati a Pandoc
    - Grunt-Panda
    - Pandocomatic

PANDOC UN WORKFLOW DI ESECUZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Per creare un workflow di esecuzione di documenti MD è possibile predisporre la seguente configurazione
    - Installare Pandoc e un motore LaTeX
    - Installare il filtro Pandoc-crossref (installazione da qui)
    - Installare Visual Studio Code
        - Installare le estensioni vscode-pandoc e pandoc citer
        - Collegare VS Code con GitHub
- In questo modo è possibile
    - Editare documenti MD da un repository condiviso
    - Gestire le trasformazioni usando Pandoc
    - Gestire i riferimenti incrociati nel documento