Scripting no Processamento de Linguagem Natural

MEI

Trabalho Prático 2

Template multi-file

Bruno Campos
(pg50275)

11 de junho de 2023

Resumo

Relatório para ferramenta Python que permite a criação de diretorias estruturadas a partir de um template, automatizando a criação de diretorias e ficheiros (bem como o conteúdo de ficheiros se desejado). Adicionalmente, é fornecida uma ferramenta complementar que cria um template a partir da diretoria desejada.

Conteúdo

1 Ferramenta principal: mkfstree **2**
1.1 Utilização 2
1.2 Formato do Template 2

2 Ferramenta complementar: mktemplateskel **3**
2.1 Utilização 3

3 Testes e Resultados **4**
3.1 Teste mkfstree 4
3.1.1 Template 4
3.1.2 Output 5
3.2 Teste mktemplateskel 6
3.2.1 Diretoria 6
3.2.2 Output 6

Capítulo 1

Ferramenta principal: mkfstree

1.1 Utilização

```
mkfstree [-h] [-n NAME] [-a AUTHOR] [--version] template_file

    positional arguments:
        template_file
    optional arguments:
        -h, --help          show this help message and exit
        -n NAME, --name NAME Project name
        -a AUTHOR, --author AUTHOR Project author
        --version, -V       show programs version number and exit
```

1.2 Formato do Template

```
=== meta                  // Declaracao do nome e autor sao obrigatorios
name: nome do projeto     // (mas podem ser deixados em branco).
author: nome do autor
// ... Declaracao das variaveis desejadas aqui.

=== tree                  // Estrutura desejada da arvore de diretorios.

ficheiro;
{{name}}/                 // substituido pelo valor da variavel
  - {{name}}.md;          // "-", declarar ficheiros ou diretorias dentro duma diretoria.
  - tests/                // diretorias terminam com '/'
     - teste.py;          // ficheiros terminam em ';'

=== ficheiro              // Conteudo do ficheiro (nao obrigatorio).
...
```

Capítulo 2

Ferramenta complementar: mktemplateskel

2.1 Utilização

```
mktemplateskel [-h] [-n NAME] [-a AUTHOR] [-c] [-V] [-o OUTPUT] directory

    positional arguments:
        directory           Root directory
    optional arguments:
        -h, --help          show this help message and exit
        -n NAME, --name NAME Project name
        -a AUTHOR, --author AUTHOR Project author
        -c, --content       Specify if file contents should be included
        -V, --version       show programs version number and exit
        -o OUTPUT, --output OUTPUT
```

Capítulo 3

Testes e Resultados

Testes realizados em WSL2 (Windows Subsystem for Linux).
A ferramenta criado está disponibilizada em https://test.pypi.org/project/mkfstree/ e pode ser obtida correndo o comando:

```
pip install -i https://test.pypi.org/simple/ mkfstree
```

3.1 Teste mkfstree

3.1.1 Template

```
=== meta
name: teste
author: JJoao

=== tree

pyproject.toml;
{{name}}/
  - __init__.py;
  - {{name}}.md;
  - {{name}};
exemplo/
  - aninhado/
    - adeus.txt;
README.md;
tests/
  - test-1.py;

=== pyproject.toml
[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"
```

```
[project]
name = "{{name}}"
authors = [ {name = "{{author}}", email = "FIXME"}]
license = {file = "LICENSE"}
dynamic = ["version", "description"]
dependencies = [ ]
readme = "{{name}}.md"

[project.scripts]
## script1 = "{{name}}:main"

=== {{name}}.md

# NAME

{{name}} - FIXME the fantastic module for...

=== __init__.py
""" FIXME: docstring """
__version__ = "0.1.0"

=== test-1.py
import pytest
import {{name}}

def test_1():
    assert "FIXME" == "FIXME"
```

3.1.2 Output

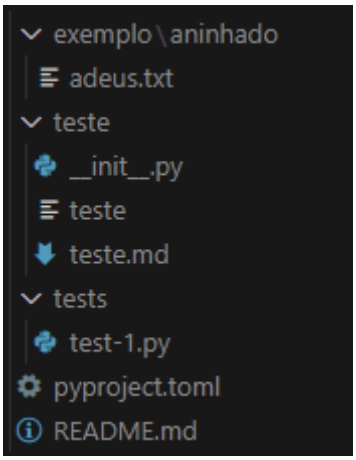

Figura 3.1: Estrutura Criada

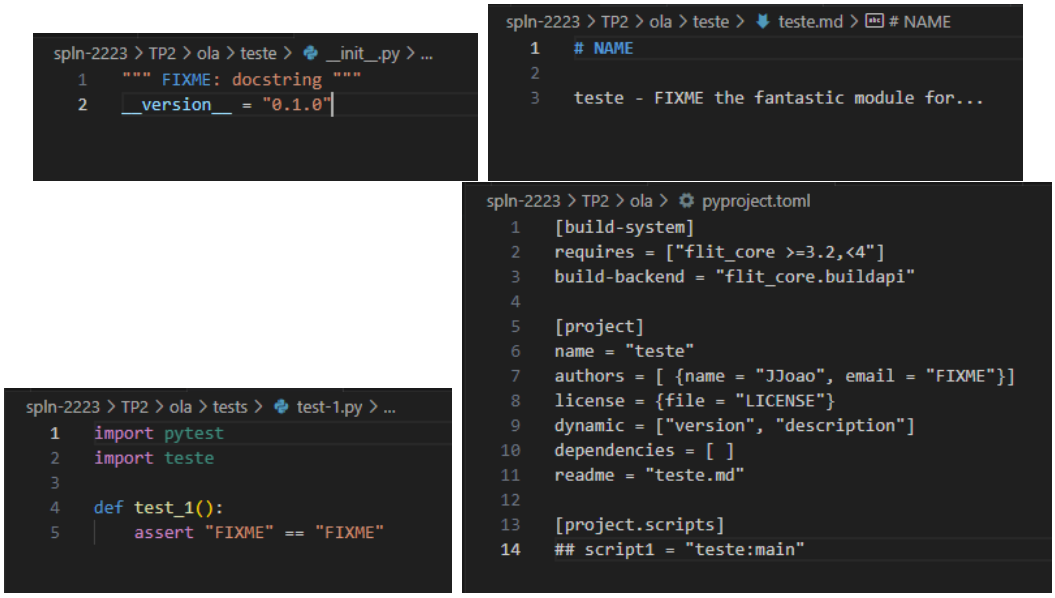

Figura 3.2: Conteúdos Criados

3.2 Teste mktemplateskel

3.2.1 Diretoria

A diretoria usada foi a gerada no teste anterior.

3.2.2 Output

```
=== meta
name: teste
author: JJoao

=== tree

pyproject.toml
README.md
exemplo/
   - aninhado/
      - adeus.txt
teste/
   - teste
   - teste.md
   - __init__.py
tests/
   - test-1.py

=== pyproject.toml
```

```
[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[project]
name = "teste"
authors = [ {name = "JJoao", email = "FIXME"}]
license = {file = "LICENSE"}
dynamic = ["version", "description"]
dependencies = [ ]
readme = "teste.md"

[project.scripts]
## script1 = "teste:main"

=== teste.md
# NAME

teste - FIXME the fantastic module for...

=== __init__.py
""" FIXME: docstring """
__version__ = "0.1.0"

=== test-1.py
import pytest
import teste

def test_1():
    assert "FIXME" == "FIXME"
```