themispy

Khipo

11 out. 2022

Pacotes

1 project **3**
- 1.1 utils.py . . . 3

2 scrapy **5**
- 2.1 items.py . . . 5
- 2.2 pipelines.py . . . 5
- 2.3 readers.py . . . 6
- 2.4 spiders.py . . . 7

Índice **9**

Seja bem-vindo(a) à documentação do componente **themispy**, utilizado durante o desenvolvimento do projeto Themis para extração de dados. Sua estrutura é muito simples, sendo divida em dois pacotes:

- **Project**: referente à funcionalidades relacionadas aos projetos em geral;
- **Scrapy**: referente à funcionalidades relacionadas ao framework e sua integração com a Azure.

Você encontrará mais detalhes sobre os pacotes em suas respectivas páginas, as quais podem ser acessadas na barra de navegação lateral. Você pode instalar o pacote através do comando `pip install themispy`. Mais detalhes sobre sua versão podem ser encontados na página do PyPI.

CAPÍTULO 1

project

Pacote destinado a conter funcionalidades genéricas aos projetos.

1.1 utils.py

themispy.project.utils.**split_filepath**(*url: str*) → list[str, str]

Divide a URL de um arquivo e retorna uma tupla contendo dois elementos, respectivamente: o nome do arquivo e sua extensão.

Parâmetros
url (*str*) – URL do arquivo.

Retorna
Nome do arquivo e sua extensão como uma tupla de dois elementos.

themispy.project.utils.**get_logpath**(*tz: str = 'America/Sao_Paulo'*) → str

Função utilizada para construir um padrão de data com o formato adequado para utilização nas pastas do sistema.

Parâmetros
tz (*str*) – Fuso-horário. (Padrão: America/Sao_Paulo)

Retorna
Data atual com o formato adequado.

CAPÍTULO 2

scrapy

Neste pacote, os módulos foram criados seguindo do modelo do Scrapy, a fim de manter um padrão de documentação e uso.

A única novidade é o módulo denominado `readers.py`, onde existe uma função criada para ler fontes de dados de blobs na Azure.

2.1 items.py

class themispy.project.items.**FileDownloader**

Classe de Item do Scrapy que serve como base para baixar arquivos. Extende a classe de 'Item' do Scrapy.

file_urls

Campo que receberá a URL do arquivo a ser baixado.

files

Campo que registrará o status do arquivo a ser baixado.

2.2 pipelines.py

class themispy.project.items.**AzureBlobUploadPipeline**

Classe criada a fim de subir arquivos para o Azure Storage. Extende a classe 'Spider' padrão do Scrapy.

blob_client

Cliente de conexão com um Blob no Azure Storage. O Blob não precisa existir previamente. Para que o cliente possa ser criado, são necessários: `conn_str`, `container_name`, `blob_name`. Também, por padrão, `logging_enable` possui o valor `True`.

content

String vazia que será preenchida, linha a linha, com o dicionário será retornado da sentença `yield` na construção da Spider.

open_spider(*self*, *spider*)

Neste método será criada a conexão com o Blob e também inicializado atributo `content`.

process_item(*self*, *item*, *spider*)

Aqui será processado o retorno dos dados do crawler e convertido para `.jsonl`.

close_spider(*self*, *spider*)

É aqui, durante o encerramento da spider, que o todo o conteúdo gerado durante o processamento será enviado para o Azure Storage, através do método `upload_blob`. Por padrão, será passado como argumento `overwrite=True`.

class themispy.project.items.**AzureFileDownloaderPipeline**

Classe extendida a partir da classe `FilesPipeline` do Scrapy. É utilizada para se baixar arquivos (os quais são fornecidas suas URLs de download). Os arquivos são enviados diretamente para o Azure Storage.

Nota: Não esquecer de passar nas configurações do Scrapy a chave `FILES_STORE`.

---

spiderinfo

Informações da spider que serão necessárias para o registro do status de download dos arquivos.

container_client

Cliente de conexão com um container no Azure Storage. O Container precisa existir previamente. Para que o cliente possa ser criado, são necessários: `conn_str` e `container_name`. Também, por padrão, `logging_enable` possui o valor `True`.

blob_client

Cliente de conexão com um Blob no Azure Storage. O Blob não precisa existir previamente. Para que o cliente possa ser criado, é necessário passar o nome do `blob` que será criado.

open_spider(*self*, *spider*)

É durante a abertura da spider que a conexão com o container é criada. Dessa maneira, independentemente de quantos arquivos serão baixados, apenas uma conexão com o container é criada.

file_downloaded(*self*, *response*, *request*, *info*, *, *item=None*)

É precisamente neste método, exatamente no momento em que os dados do arquivo baixado estão em memória, que é criado um cliente com o Blob e o arquivo é subido no Azure Storage. Por padrão, é passado ao `upload_blob` o argumento `overwrite=True`.

2.3 readers.py

themispy.project.readers.**list_blob_content**(*url: str*, *encoding: str = 'UTF-8'*, *logging_enable: bool = True*) → list[str]

Lê o conteúdo do blob em questão, convertendo para lista utilizando as quebras de linhas.

Parâmetros

- **url** (*str*) – Caminho completo do blob dentro da Azure. Exemplo: `https://<nome_do_storage>.blob.core.windows.net/<container>/meu_arquivo.jsonl`.
- **encoding** (*str*) – Formato de codificação dos caracteres. (`UTF-8` é o padrão.)
- **logging_enable** (*bool*) – Indica se a função deverá ativar o logger ou não. (Padrão é `True`.)

Retorna
Conteúdo do blob como lista de strings.

2.4 spiders.py

themispy.project.spiders.**run_spider**(*spider: scrapy.Spider*, *pipeline: str = None*, *settings: dict = None*, *override: bool = False*) → None

Processo para executar spiders.

Parâmetros

- **spider** (*scrapy.Spider*) – Spider a ser executada.
- **pipeline** (*str*) – Pipeline a ser utilizada durante a execução da spider. Deve ser `blob` ou `download`. Referindo, respectivamente, as pipelines de AzureBlobUpload ou AzureFileDownloader.
- **settings** (*dict*) – Configurações do Scrapy para a execução das spiders. Passe aqui suas configurações personalizáveis para serem adicionadas às padrões.
- **override** (*bool*) – Caso seja `True`, as configurações recebidas sobrescreverão todas as anteriores.

Índice

B

blob_client (*atributo themispy.project.items.AzureBlobUploadPipeline*), 5
blob_client (*atributo themispy.project.items.AzureFileDownloaderPipeline*), 6

C

close_spider() (*método themispy.project.items.AzureBlobUploadPipeline*), 6
container_client (*atributo themispy.project.items.AzureFileDownloaderPipeline*), 6
content (*atributo themispy.project.items.AzureBlobUploadPipeline*), 5

F

file_downloaded() (*método themispy.project.items.AzureFileDownloaderPipeline*), 6
file_urls (*atributo themispy.project.items.FileDownloader*), 5
files (*atributo themispy.project.items.FileDownloader*), 5
função interna
    themispy.project.readers.list_blob_content(), 6
    themispy.project.spiders.run_spider(), 7
    themispy.project.utils.get_logpath(), 3
    themispy.project.utils.split_filepath(), 3

O

open_spider() (*método themispy.project.items.AzureBlobUploadPipeline*), 6
open_spider() (*método themispy.project.items.AzureFileDownloaderPipeline*), 6

P

process_item() (*método themispy.project.items.AzureBlobUploadPipeline*), 6

S

spiderinfo (*atributo themispy.project.items.AzureFileDownloaderPipeline*), 6

T

themispy.project.items.AzureBlobUploadPipeline (*classe interna*), 5
themispy.project.items.AzureFileDownloaderPipeline (*classe interna*), 6
themispy.project.items.FileDownloader (*classe interna*), 5
themispy.project.readers.list_blob_content()
    função interna, 6
themispy.project.spiders.run_spider()
    função interna, 7
themispy.project.utils.get_logpath()
    função interna, 3
themispy.project.utils.split_filepath()
    função interna, 3