Querido Diário

Open Knowledge Brasil

27 jul., 2022

Querido Diário:

CAPÍTULO 1

O que é o Querido Diário

Um **Diário Oficial** é uma publicação feita por vários esferas da administração pública brasileira, seja na esfera federal, estadual ou municipal e dos poderes executivo, legislativo e judiciário. É através dele que qualquer ação tomadas por essas esferas e poderes torne-se oficial para a população.

Mesmo com os esforços recorrentes para fortalecer a Lei de Acesso à Informação pelo país, a comunicação oficial permanece - na maioria do território nacional - restrita a documentos em formato **PDF**. Esses documentos, apesar de estarem públicos, não estão estruturados de uma maneira que permita a análise por um computador ou que sejam consultados em conjunto (por exemplo, buscando palavras-chaves em diversas publicações ao mesmo tempo).

O objetivo do **Querido Diário** é facilitar o acesso público aos Diários Oficiais dos municípios brasileiro, permitindo que qualquer pessoa tenha acesso e consulte as informações dentro dessas publicações.

Para atingir esse objetivo, temos várias frentes de trabalho que visam mapear e descobrir onde os Diários Oficiais estão publicados, obté-los com regularidade para que possamos processá-los e disponibiliza-los de uma maneira que seja possível com que a população possa pesquisar o seu conteúdo, seja através de uma página de consulta ou uma API para que esse processamento possa ser feito através de um computador.

1.1 Como o projeto é mantido

O **Querido Diário** é um projeto de código aberto, isso quer dizer que todo código utilizado para extrair, processar e disponibilizar os Diários está disponível para que qualquer pessoa possa visualizar, utilizar e alterar para as suas necessidades.

A Open Knowledge Foundation Brasil através do **Programa Ciência de Dados para Inovação Cívica** é a entidade responsável por manter toda a infra estrutura do projeto e organizar o seu desenvolvimento, além da contribuição contínua de centenas de voluntários em todo o mundo.

1.2 Como colaborar

Você pode contribuir:

- Financeiramente ajudando a manter toda nossa infra estrutura de servidores e armazenamento, no desenvolvimento e na organização de eventos relacionados (sprints de desenvolvimento, lives de capacitação e fomento à colaboração) para atingirmos o maior número de cidades possível.
- Com código para quem tem mais conhecimentos técnicos, temos vários projetos que precisam de mãos para desenvolvimento (raspadores, APIs, infra estrutura, documentação, etc).

1.3 Contato

Você pode entrar em contato com o projeto em nossos canais no Discord.

CAPÍTULO 2

Escrevendo um novo spider

Antes de inicial o desenvolvimento de um raspador para uma nova cidade, leia cuidadosamente as seguintes diretrizes necessárias para manter o projeto mais consistente e garantir que sua contribuição seja mais facilmente aceita.

Ler o código de raspadores que já existem também irá ajudar a você compreender melhor como o projeto está organizado e como iniciar o desenvolvimento.

2.1 Regras básicas para seu spider

- Como regra, todos os spiders deve herdar de *BaseGazetteSpider*
- Nome da classe do spider deve seguir o seguinte padrão *<SiglaEstado><NomeCidade>Spider* (por exemplo *SpSaoPauloSpider* ou *MtFelizNatalSpider*)
- Atributo *name* do spider deve seguir o seguinte padrão *<sigla_estado>_<nome_cidade>* (por exemplo *sp_sao_paulo* ou *mt_feliz_natal*)
- Adicione o atributo *TERRITORY_ID* com o código encontrado no arquivo *territories.csv*
- Adicione o atributo *start_date* com a data do primeiro Diário Oficial disponível
- Utilize *.get()* e *.getall()* ao invés de *.extract_first()* e *.extract()*

Exemplo de um spider:

```
from datetime import date
from gazette.spiders.base import BaseGazetteSpider


class SpJundiaiSpider(BaseGazetteSpider):
    TERRITORY_ID = "3525904"
    name = "sp_jundiai"
    allowed_domains = ["jundiai.sp.gov.br"]
    start_urls = ["https://imprensaoficial.jundiai.sp.gov.br/"]
```

(continues on next page)

(continuação da página anterior)

```
    start_date = date(2000, 1, 1)

    def parse(self, response):
        # Aqui colocamos toda nossa lógica de extração
        ...
```

2.2 Definição de campos

Estas são as informações que queremos extrair de cada um dos Diários Oficiais:

- **date (datetime.date)**: data de publicação
- **edition_number (string)**: número de edição do Diário Oficial
- **is_extra_edition (boolean)**: alguns Diarios são edições extras. Você geralmente consegue identificá-los ao encontrar termos como *Extra*, *Extraordinário*, *Suplemento*, etc. Se não for possível identificá-lo, retorne *False* como padrão
- **territory_id (string)**: valor do atributo *TERRITORY_ID* do spider (preenchido automaticamente)
- **power (string)**: se o Diário Oficial é referente ao Poder Executivo, ao Poder Legislativo ou ambos (*executive*, *legislative*, *executive_legislative*). Será necessário abrir manualmente o conteúdo de alguns Diários para descobrir essa informação
- **scraped_at (datetime.datetime)**: fixado em *datetime.datetime.utcnow()* (preenchido automaticamente)
- **file_urls (URL list)**: lista das URLs para baixar os Diários Oficiais (geralmente temos apenas um valor aqui)

2.3 Filtro por data

Nós queremos que todos os spiders sejam capazes de coletar **TODOS** os Diários Oficiais disponíveis. Mas também queremos ser capazes de restringir a quantidade de dados por período de tempo. Isso deve ser alcançado através dos atributos *start_date* e *end_date* que podemos passar ao executar os spiders.

O valor padrão do *start_date* deve ser a data do primeiro Diário Oficial disponível na página do município (*hard coded*). Você deve pesquisar manualmente até encontrar essa data e adicioná-la como um argumento do seu spider.

No exemplo a seguir, temos um spider que irá considerar como data inicial o dia 15 de setembro de 2001 como a primeira data onde é possível encontrar um Diário Oficial para coletar:

```
from datetime import date
from gazette.spiders.base import BaseGazetteSpider

class MyCitySpider(BaseGazetteSpider):
    # (...)
    name = "my_city_spider"
    start_date = date(2001, 9, 15)

    def parse(self, response):
        ...
```

Caso queiramos coletar os dados a partir de outra data, podemos utilizar o seguinte comando ao executar o nosso spider (no exemplo, iremos coletar dados a partir de 28 de abril de 2022)

```
scrapy crawl my_city_spider -a start_date="2021-04-28"
```

O valor padrão de *end_date* é a data de hoje (que estamos executando o nosso spider), então **não é necessário** definí-la em nossa classe do spider. Isso está definido em *BaseGazetteSpider*.

Durante o desenvolvimento, tente fazer **a menor quantidade de requests possível**. Algumas páginas permitem que você filtre os resultados informando um período específico, outras vão exigir que você adicione alguma lógica extra para evitar visitar páginas desnecessárias e retornar Diários Oficiais fora do período desejado.

2.4 Algumas dicas

- Durante o desenvolvimento, para evitar que você faça chamadas repetidas nas páginas das cidades é possível utilizar a configuração HTTPCACHE_ENABLED do Scrapy. Isso também faz com que as execuções sejam mais rápidas, já que todos os dados ficam armazenados localmente
- O shell do Scrapy ajuda muito na hora de testar seletores CSS e XPath

2.5 Submetendo um novo PR

Antes de submeter um novo PR do seu spider, garanta que ele esteja funcionando adequadamente.

EXECUTE O SEU SPIDER EM SEU AMBIENTE DE DESENVOLVIMENTO E AGUARDE ATÉ QUE ELE FINALIZE SUA EXECUÇÃO

Sugestões de verificações:

- Se os arquivos baixados são válidos
- Se você está coletando todos os Diários Oficiais disponíveis;
- Se ao especificar os argumentos *start_date* e/ou *end_date*, você não está coletando mais Diários Oficiais do que o esperado;
- Se ao executar o spider mais de uma vez, os resultados são os mesmos;
- Se não existem erros de execução (*log/ERROR* nas estatísticas do spider);