

Automaty, Języki i Obliczenia

automatycznie wyodrębnianie danych o gatunkach roślin

Robert Kozik

Grudzień 2021

Spis treści

1	Założenia projektu	2
1.1	Cel projektu	2
1.2	Technologie	2
1.3	Działanie Aplikacji	2
2	Projekt	3
2.1	Struktura drzewa HTML artykułu	3
2.2	Parsowanie wybranej struktury drzewa HTML	4
2.3	Tworzenie wyjściowej struktury pliku	6
2.4	Zapis pliku do bazy danych	6
3	Realizacja	7
3.1	Moduł klienta bazodanowego	7
3.2	Moduł parsowania drzewa HTML	8
3.3	Postawienie środowiska deweloperskiego	9
3.4	Schemat działania aplikacji	9

1 Założenia projektu

1.1 Cel projektu

Projekt ma na celu automatyzację wyodrębniania danych ze stron internetowych, zapisanie ich w czytelnej formie oraz w zewnętrznej bazie danych. Dane będą zawierać podstawowe informacje o wybranych roślinach doniczkowych, które to będą mogły być automatycznie aktualizowane w bazie danych poprzez uruchomienie powstałego w toku tego projektu programu.

1.2 Technologie

W wyniku skrupulatnych poszukiwań, do wykonania projektu należy wybrać język programowania *Python*. Wybór ten jest podyktowany szerokim spektrum gotowych rozwiązań, bibliotek służących do parsowania struktury HTML wybranej strony internetowej. Jest to niezbędna czynność umożliwiająca wyciąganie danych umieszczonych wewnątrz tagów HTML. Do samego wyciągania danych posłuży biblioteka *BeautifulSoup*. Umożliwia ona przeszukiwanie sparsowanego wcześniej pliku HTML ze względu na umieszczone w nim tagi HTML oraz ich atrybuty, takie jak klasa czy id tagu.

W celu pobrania struktury drzewa HTML z wybranej strony internetowej należy użyć biblioteki *requests*. Umożliwia ona wykonanie żądań HTTP, oraz zapisanie do zmiennej odpowiedzi serwera.

Zapisanie wyodrębnionych danych będzie odbywało się do nierelacyjnej bazy danych *MongoDB*. Wybór tego systemu bazodanowego podyktowany był jego obecnością w innym przedsięwzięciu ¹, którego, opisywany przez ten dokument projekt, będzie integralną częścią. W celu komunikacji z zewnętrznym dostawcą bazy danych należy użyć biblioteki *PyMongo*. Dystrybucja ta jest rekomendowana przez twórców *MongoDB*[2] do komunikacji z taką bazą danych.

1.3 Działanie Aplikacji

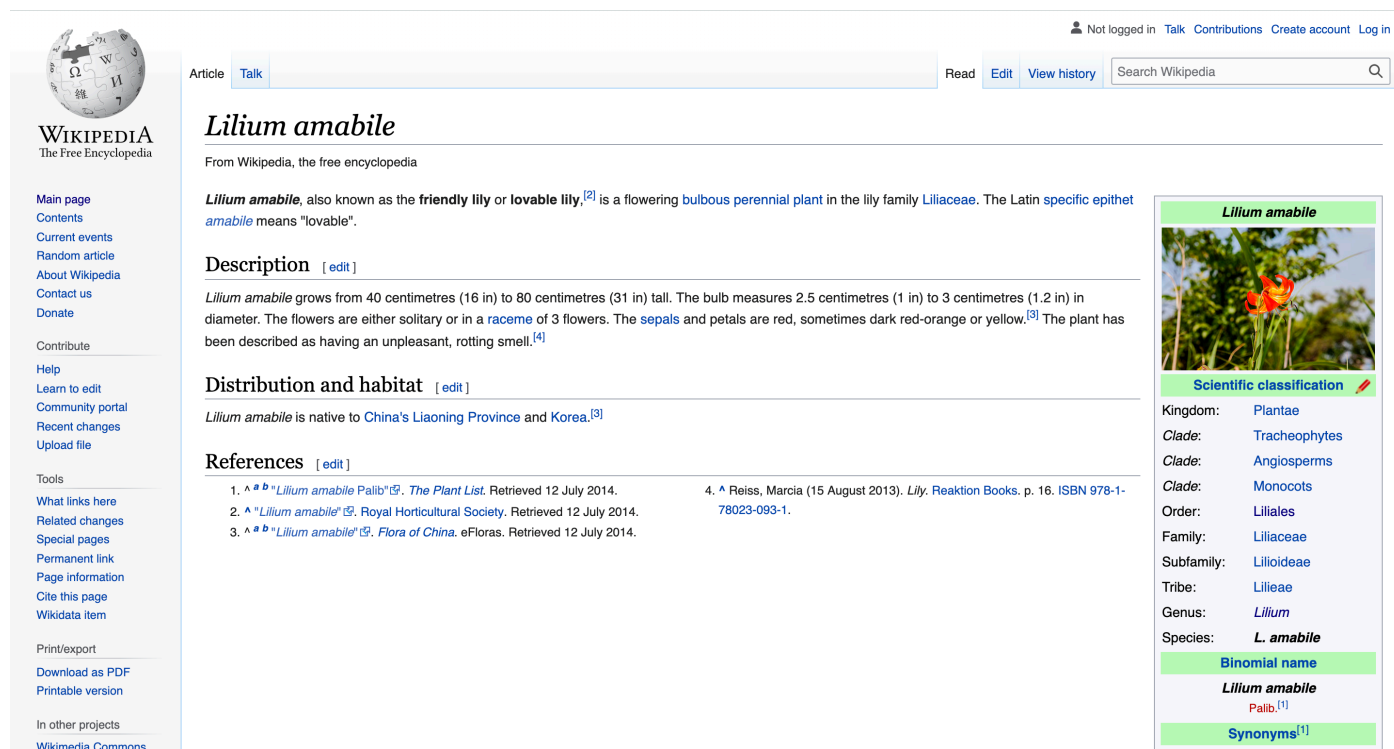
Powstała aplikacja na podstawie podanej listy, dowolnej długości, adresów artykułów *Wikipedii* ma wyodrębnić informacje o nazwach gatunku rośliny, jej kategoriach systematycznych, zdjęcia oraz opis. Dane te mają być możliwe zarówno do zapisania w postaci pliku, jak i przekazania ich do zewnętrznej bazy danych *MongoDB*. Przy założeniu że struktura drzew HTML, podanych w liście adresów, jest jednakowa i nie zmieniała się od daty implementacji rozwiązania.

¹Jest to praca inżynierska o temacie "System zdalnego monitorowania warunków wegetacji roślin"

2 Projekt

2.1 Struktura drzewa HTML artykułu

Artykuły strony Wikipedia, będące przedmiotem analizy w tym projekcie ¹, posiadają standaryzowaną konstrukcję umożliwiającą wyodrębnianie danych w nich zawartych poprzez działanie skryptu. Będąc w kręgu zainteresowania dane ² W większości można znaleźć w zbiorczej tabeli znajdującej się po prawej stronie wybranego artykułu. Dane zawarte w tabelarycznym wykazie są prostsze do parsowania i wyodrębniania ze względu na powtarzalny charakter tagów HTML. W celu stworzenia tabeli używa się tagów: `<table/>`, oznaczającego całą tabelę, `<tr/>` jako jej wiersz, oraz `<td/>` jako pod-kolumnę. Główna nazwa gatunkowa zawsze zawiera się w tytule artykułu, natomiast opis jest pierwszą sekcją po niej następującą. Dzięki takim spostrzeżeniom można zakładać że dane zawarte w tych artykułach są możliwe do wyodrębnienia.



The screenshot shows the Wikipedia article for *Lilium amabile*. The page layout includes a sidebar on the left with navigation links, a main content area with the article text, and a right-hand table for scientific classification and synonyms.

Article Title: *Lilium amabile*

From Wikipedia, the free encyclopedia

Lilium amabile, also known as the **friendly lily** or **lovable lily**,^[2] is a flowering **bulbous perennial plant** in the lily family **Liliaceae**. The Latin specific epithet *amabile* means "lovable".

Description [edit]

Lilium amabile grows from 40 centimetres (16 in) to 80 centimetres (31 in) tall. The bulb measures 2.5 centimetres (1 in) to 3 centimetres (1.2 in) in diameter. The flowers are either solitary or in a **raceme** of 3 flowers. The **sepals** and petals are red, sometimes dark red-orange or yellow.^[3] The plant has been described as having an unpleasant, rotting smell.^[4]

Distribution and habitat [edit]

Lilium amabile is native to **China's Liaoning Province** and **Korea**.^[3]

References [edit]

- ¹ ^a ^b ^c *"Lilium amabile Palib"*. *The Plant List*. Retrieved 12 July 2014.
- ² ^a *"Lilium amabile"*. *Royal Horticultural Society*. Retrieved 12 July 2014.
- ³ ^a ^b *"Lilium amabile"*. *Flora of China*. eFloras. Retrieved 12 July 2014.
- ⁴ ^a Reiss, Marcia (15 August 2013). *Lily*. Reaktion Books. p. 16. ISBN 978-1-78023-093-1.

Scientific classification [edit]

Kingdom:	Plantae
Clade:	Tracheophytes
Clade:	Angiosperms
Clade:	Monocots
Order:	Liliales
Family:	Liliaceae
Subfamily:	Lilioideae
Tribe:	Lilieae
Genus:	<i>Lilium</i>
Species:	<i>L. amabile</i>

Binomial name

Lilium amabile
Palib.^[1]

Synonyms^[1]

Rysunek 2.1: Wygląd przykładowego artykułu [6] na którym można zobaczyć poszukiwane dane: nazwę gatunkową, opis, oraz zebrane w tabeli: zdjęcie, klasyfikację systematyczną i synonimy nazwy

¹ Artykuły dotyczące roślin doniczkowych, ozdobnych np. Lili

² nazwy gatunkowe, klasyfikacja, zdjęcia, opis

2.2 Parsowanie wybranej struktury drzewa HTML

Znalezione wcześniej części artykułu należy zlokalizować na podstawie użytych właściwości w obiekтовym modelu dokumentu³. W związku z wybraną biblioteką do przetwarzania obiekowego modelu dokumentu - *BeautifulSoup*, poszukiwania wybranego tagu można prowadzić na podstawie zarówno nazwy tagu HTML, przypisanych do niego klas i identyfikatorów, jak i wyrażeń regularnych. Filtry te można łączyć w celu dokładniejszego wyszukiwania.

Wyodrębnianie danych dotyczących zdjęć stanowi najprostsze zadanie, ponieważ wszystkie zdjęcia która można znaleźć w takim artykule stanowią przedmiot zainteresowania w tym projekcie. W związku z czym w celu ich znalezienia należy wyszukać wszystkie możliwe tagi `` znajdujące się w obiekтовym modelu dokumentu artykułu. Ponadto w celu odfiltrowania grafik niebędących zdjęciami, lecz graficznymi przedstawieniami ikon (np. ołówek jako ikona edycji) należy sprawdzić czy w atrybucie `alt` obiektu znajduje się rozszerzenie `".jpg"`. W plikach znajdujących się w artykułach Wikipedii, atrybut ten przechowuje ścieżkę dostępu do pliku, dlatego takie podejście można uznać za słuszne w celu filtracji dużej ilości obiektów ``.

Opis danego gatunku rośliny znajduje się w obiekcie `<p />` artykułu który jest bezpośrednim rodzeństwem⁴ elementu z unikalną klasą `"mw-empty-elt"`.

```
<p class="mw-empty-elt"> </p> == $0
▶<table class="infobox biota" style="text-align: left; width: 200px; font-size: 100%">...</table>
▼<p>
  ▶<i>...</i>
    ", also known as the "
    <b>friendly lily</b>
    " or "
    <b>lovable lily</b>
    ", "
  ▼<sup id="cite_ref-RHC_2-0" class="reference">
    <a href="#cite_note-RHC-2">[2]</a>
  </sup>
  " is a flowering "
  <a href="/wiki/Bulbous" class="mw-redirect" title="Bulbous">bulbous</a>
  <a href="/wiki/Perennial plant" title="Perennial plant">perennial plant</a>
  " in the lily family "
  <a href="/wiki/Liliaceae" title="Liliaceae">Liliaceae</a>
  ". The Latin "
  <a href="/wiki/Binomial nomenclature" title="Binomial nomenclature">specific epithet</a>
  ▶<i>...</i>
    " means "lovable". "
  </p>
```

Rysunek 2.2: Wygląd fragmentu obiekowego modelu dokumentu artykułu [6] na którym widoczna jest klasa `"mw-empty-elt"` a następnie fragment który chcemy wyodrębnić

Elementem charakterystycznym tabeli zbiorczej widocznej zawsze po prawej stronie artykułu jest

³ang. Document Object Model - **DOM**

⁴węzeł znajdujący się w drzewie bezpośrednio za węzłem bieżącym

klasa "infobox biota", po tej klasie można z całą pewnością zlokalizować ten konkretny element. W celu zebrania danych z tej tabeli, należy zmapować jej strukturę na postać słownikową (ang. *dict*) w Pythonie, w której klucz stanowi nagłówek danej sekcji tabeli a wartość, dane znajdujące się pod/obok nagłówka. Problem stanowi fakt, że zależenie od wybranego artykułu, pierwszy interesujący nas, będący bezpośrednio pod zdjęciem, nagłówek ma różne położenie w strukturze DOM. Należy więc na początku zlokalizować ten nagłówek i od niego rozpoczynać mapowanie tabeli.

```
▼<table class="infobox biota" style="text-align: left; width: 200px; font-size: 100%">
  ▼<tbody>
    ▼<tr>
      ▶<th colspan="2" style="text-align: center; background-color: rgb(180,250,180)">...</th>
    </tr>
    ▼<tr>
      ▶<td colspan="2" style="text-align: center">...</td>
    </tr>
    ▼<tr>
      ▶<th colspan="2" style="min-width:15em; text-align: center; background-color: rgb(180,250,180)">...</th>
    </tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▼<tr>
      <td>Order: </td>
      ▼<td>
        <a href="/wiki/Liliales" title="Liliales">Liliales</a>
      </td>
    </tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
    ▶<tr>...</tr>
  </tbody>
```

Rysunek 2.3: Wygląd fragmentu obiektowego modelu dokumentu artykułu [6] na którym widoczna jest budowa tabeli

Nazwa gatunkowa rośliny znajduje się zarówno w nagłówku artykułu, jak i w tabeli zbiorczej danych pod kluczem: "Binomial name". W związku z uproszczeniem architektury jak i ze zmniejszeniem ilości przeszukiwań całego modelu obiektowego należy użyć wartości znajdującej się w tabeli zbiorczej, omawianej powyżej.

2.3 Tworzenie wyjściowej struktury pliku

Otrzymana w wyniku przeszukiwania obiektowego modelu dokumentu struktura nie jest zarówno przyjemna dla oka, jak i jest niemożliwa do przekazania do bazy danych. Aby uzyskać pożądaną strukturę pliku potrzebna jest ponowne zmapowanie struktury, przy znajomości wyglądu pliku zwracanego z modułu wydobywającego dane. Docelowo, struktura pliku, który może być załadowany do bazy danych, powinna być w formacie **JSON** i wyglądać następująco:

```
[...,
  {
    "classification": {
      Kingdom,
      Clade,
      Order,
      Family,
      Subfamily,
      Tribe,
      Genus,
      Species,
    },
    images: {
      <<nazwa zdjęcia>> : <<url>>,
      ...
    },
    name,
    description,
  },
...]
```

2.4 Zapis pliku do bazy danych

Tak przygotowany wyjściowy plik jest gotowy do przekazania do bazy danych MongoDB. W związku z tym, że ta baza danych przyjmuje dane w formacie **JSON** w celu ich zapisania należy, poprzez przygotowany moduł kliencki bazy danych, przesłać przygotowany wcześniej plik. W celu zapewnienia braku duplikacji encji, przed zapisaniem należy wyczyścić całkowicie bazę danych. Każdorazowo przekazywany plik wyjściowy będzie miał taką samą kolejność wpisów w związku z czym takie działanie nie powinno generować błędów związanych z ciągłością indeksów kolejnych encji.

3 Realizacja

Całość aplikacji została podzielona na trzy osobne moduły i jeden plik wejściowy:

- `data_fetcher.py` - parsujący DOM artykułu i wydobywający z niego dane
- `mongo_client.py` - moduł zapewniający komunikację z bazą danych
- `helpers.py` - dostarcza funkcje wspierające wydobywanie danych
- `urls.py` - tablica adresów artykułów do parsowania

3.1 Moduł klienta bazodanowego

Na Moduł łączący się z bazą danych składa się klient udostępniony w bibliotece *PyMongo* uzupełniony danymi na temat bazy danych z którą ma się połączyć. W celu utworzenia obiektu tej klasy w konstruktorze należy podać nazwę kolekcji z którą ten moduł będzie wchodzić w interakcję. Jako że klasa ta rozszerza klasę **Db_client**, wymagane jest aby nadpisywała ona metodę **clean_push**, która następnie jest wykorzystywana w głównym module odpowiedzialnym za parsowanie i wyodrębnianie danych z artykułów.

```
def clean_push(self, collection, data):  
    selected_collection = self.db[collection]  
    selected_collection.drop()  
    self.push_multiple_to_db(collection=collection, data=data)
```

W celu uproszczenia implementacji wymaganej metody **clean_push** utworzono metodę pomocniczą **push_multiple_to_db** odpowiedzialną za zapisywanie w bazie danych tablicy encji.

3.2 Moduł parsowania drzewa HTML

Na moduł umożliwiający parsowanie drzewa HTML i wyodrębnianie danych składa się niezależna klasa posiadające pola **urls**, **output_dict**, oraz **db_client**. Pole **db_client** przyjmuje argument będący klasą **Db_client** lub obiektem klasy dziedziczącej po niej. Wykorzystanie wzorca Dependency Injection umożliwia połączenie tego modułu do dowolnej bazy danych poprzez rozszerzenie klasy **Db_client**. Klasa ta posiada główną metodę **fetch_data** która odpowiada zarówno za wykonanie żądania HTTP jak i parsowanie drzewa wraz z wyodrębnianiem danych. Końcowo zwraca odpowiednią wersję danych do zmiennej **output_dict** po wykonaniu funkcji **get_proper_json_from_raw_data**.

```
def get_proper_json_from_raw_data(dict):
    proper_json = {}
    classification = {}

    for sub_dict in dict["table_data"]["Scientific classification"]:
        for key in sub_dict:
            sliced_key = key[:-1]
            if sliced_key in classification:
                if isinstance(classification[sliced_key], list):
                    classification[sliced_key] =
                        classification[sliced_key] + [sub_dict[key]]
                else:
                    classification[sliced_key] =
                        [classification[sliced_key]] + [sub_dict[key]]
            else:
                classification[sliced_key] = sub_dict[key]

    proper_json["classification"] = classification
    proper_json["images"] = dict["images"]
    proper_json["name"] = dict["table_data"]["Binomial name"][0]
    proper_json["description"] = dict["description"]

    return proper_json
```

Do zapisania słownika wynikowego służą dwie funkcje: **save_output_as_json** oraz **save_output_to_db**. Pierwsza z tych funkcji umożliwia zapis do pliku dostępnego lokalnie, natomiast druga wykorzystując klienta bazodanowego zapisuje słownik do bazy danych.

3.3 Postawienie środowiska deweloperskiego

W celu postawienia środowiska oraz uruchomienia programu należy pobrać projekt z repozytorium *github*¹ a następnie zainstalować wymagane dependencje komendami:

```
$ pip install beautifulsoup4
$ pip install lxml
$ pip install requests
$ pip install pymongo
$ pip install pymongo[srv]
$ python -c "import ssl; print(getattr(ssl, 'HAS_SNI', False))"
```

ostatnie dwie linijki umożliwiają połączenie z bazą danych *MongoDB* poprzez podanie gotowego URL z atlas MongoDB postaci[3]:

```
mongodb+srv://name:pass@cluster0.example.mongodb.net/myFirstDatabase
```

W celu uruchomienia zainstalowanego w powyższy sposób środowiska należy użyć polecenia:

```
$ python main.py
```

Takie środowisko można również postawić w środowisku izolowanym, wirtualnym za pomocą *Python venv*, co jest polecanym działaniem.

3.4 Schemat działania aplikacji

- Ładowanie listy adresów artykułów
- Dla każdego adresu wykonanie żądania HTTP w celu pobrania DOM artykułu
- Dla każdego DOM wyodrębnienie informacji o danej roślinie
- Restrukturyzacja pliku do schematu wymaganego przez bazę danych
- Zapisanie wyniku do pliku
- Zapisanie wyniku do bazy danych

¹<https://github.com/RobertKozik/AJi0>

Bibliografia

- [1] Python Software Foundation. Python documentation. <https://docs.python.org/3/>, 2021 (dostęp uzyskano 05.12.2021).
- [2] MongoDB Inc. How to Use Python with MongoDB. <https://www.mongodb.com/languages/python>, 2021 (dostęp uzyskano 04.12.2021).
- [3] MongoDB Inc. Connection String URI Format. <https://docs.mongodb.com/manual/reference/connection-string/>, 2021 (dostęp uzyskano 05.12.2021).
- [4] MongoDB Inc. PyMongo documentation. <https://pymongo.readthedocs.io/en/stable/>, 2021 (dostęp uzyskano 05.12.2021).
- [5] Leonard Richardson. Beautiful Soup documentation. <https://beautiful-soup-4.readthedocs.io/en/latest/>, 2021 (dostęp uzyskano 05.12.2021).
- [6] Wikipedia the free encyclopedia. Lilium amabile. https://en.wikipedia.org/wiki/Lilium_amabile, 2021 (dostęp uzyskano 05.12.2021).