# KOTON WEB SCRAPPER

Documentation

**Languages, Technologies, Frameworks and Libraries:**

- Programming Language: Python 3.*
- Web Scrapping Framework: BeautifulSoup
- API Development Framework: Flask
- Database Technology: Document Based
- Database Framework: CouchDB

**Python Files:**

- scrapper.py
- db_initialization.py
- search_api.py
- unit_tests.py
- run_tests.py

**Support Files:**

- data_output.json
- requirements.txt

**Folders:**

- /Screenshots

**scrapper.py**

This script handles the scrapping of information from the Koton Retailer Website in an iterative manner by using HTTP requests.

Operation Steps:

- Retrieve the main product page as an http response.
- Initialize the encoding method as utf-8 standard.

- By using BeautifulSoup library, parse the response.
- Extract the product page links from the response.
- Print the total number of products retrieved.
- By using the links, retrieve the information for each of the products in an iterative manner.
    - Retrieve product id.
    - Retrieve product title.
    - Retrieve product description
    - Retrieve product image URIs.
    - **Could not retrieve date of publishing**, as it was not mentioned on the page.
    - Retrieve product category
    - Retrieve product colour
    - Retrieve price and currency data
    - Retrieve discount condition
    - Retrieve size and inventory condition data
    - Retrieve general inventory condition
- Save the retrieved data in json format, to a json file.

**db_initialization.py**

This script creates a database for the retrieved products and saves them in the database. The database technology used in the project is CouchDB and preferred for the ease of use and document based storage purposes.

Operation Steps:

- Create a connection with the Database Server
- Provide database administration credentials.
    - Note: Not a safe way to do, normally it should be handled with safe authorization methodologies.
- Create the database for the products
- Save the data to the database by using the json file created by the previous script a.k.a scrapper.py

- Inform the user about the successful creation and insertion to the database.

**search_api.py**

This script contains the API methods for the use of retrieving product information from the database. The preferred API development framework is Flask, as it is an effective and relatively fast framework for Python language. Two types of API methods have been used in the project. First is the static type of method that retrieves specific queries and handles them accordingly by the search parameters such as id, title, description, category, colour or price range. Second type is the dynamic method that can handle multiple parameters whenever the keywords are provided.

Methods:

- search_by_id(query_id) -> str(results)
  - URL: /search/search_by_id/<query_id>
  - Takes one parameter; query_id and uses it to retrieve the product related with it, if such product exists.
- search_by_title(query_title) -> str(results)
  - URL: /search/search_by_title/<query_title>
  - Takes one parameter; query_title and uses it to retrieve the products related with it, if such products exist.
- search_by_description(query_description) -> str(results)
  - URL: /search/search_by_category/<query_category >
  - Takes one parameter; query_description and uses it to retrieve the products related with it, if such products exist.
- search_by_colour(query_colour) -> str(results)
  - URL: /search/search_by_colour/<query_colour>

    - Takes one parameter; query_colour and uses it to retrieve the products related with it, if such products exist.
- search_by_price(query_price_low, query_price_high) -> str(results)
    - URL: /search/search_by_price/<query_price_low>& <query_price_high>
    - Takes two parameters, query_price_low, which is the lowest bound of the price range demanded; and the query_price_high, which is the highest bound of the price range demanded. Then, it retrives the products within the suggested ranges, if such products exist.
- search_by_multiple_parameters() -> str(results)
    - URL: /search/multiple_search
    - Can handle multiple parameters such as:
        - Description
        - Lowest bound for price range
        - Highest bound for price range
        - Colour
    - The given parameters are used for retrieving the product that satisfy the multiple filtering aspects, if such products exist. Can be used instead of the other seperate methods, however, other methods can come in handy in some of the scenarios.

**unit_tests.py**

This script containts the tests for the various API methods in the project.

Methods:

- Test_id_search(_id) -> results
    - Test for retrieving the products from the database by using product ID.

- Test_title_search(title) -> results
    - Test for retrieving the products from the database by using product title
- Test_description_search(description) -> results
    - Test for retrieving the products from the database by using product description
- Test_category_search(category) -> results
    - Test for retrieving the products from the database by using product category
- Test_colour_search(colour) -> results
    - Test for retrieving the products from the database by using product colour
- Test_price_search(price_low, price_high) -> results
    - Test for retrieving the products from the database by using the lower and upper bounds provided by the requester.
- Test_multiple_search(description, price_low, price_high, colour) -> results
    - Test for retrieving the products from the database by providing more than one parameters and apply more complex filtering as a result.

**run_tests.py**

This script is used for running the tests in the unit_tests module. Several different scenarios have been tested for different API calls and API methods.

Tests:

- Testing the search with product ID feature
- Testing the search with product title feature
- Testing the search with product description feature
- Testing the search with product category feature
- Testing the search with product colour feature
- Testing the search with price range feature

- Testing the multiple parameter search feature

**data_output.json**

This json file contains the retrieved product information from the web scrapping script (scrapper.py)

**requirements.txt**

This txt file contains the libraries **necessary for the program to work** and the necessary packages can be installed via terminal/console depending on the need.