1 Overview

unisos.wsfClassicCars: Classic Cars App Based On WSF (Web Scraping Framework)

2 Support

For support, criticism, comments and questions; please contact the author/maintainer
Mohsen Banan at: http://mohsen.1.banan.byname.net/contact

3 Documentation

This unisos.wsfClassicCars module is usage example of unisos.wsf.

For details of Web Scraping Framework (wsf), see that module's documentation.

WSF is a general purpose scraping engine module. unisos.wsfClassicCars applies to scraping:
https://www.oldclassiccar.co.uk/forum/phpbb/phpBB2/viewtopic.php?t=12591 in specific.

4 Installation

This module is provided as a tar file.

Go to the wsfClassicCars/py3 directory.

```
Run: ./setup.py sdist

Run: pip install --no-cache-dir ./dist/unisos.wsfClassicCars-0.1.tar.gz
```

4 file will be created in your venv/bin directory. These are copies of the ones in the ./bin directory.

5 Usage

In wsfClassicCars/py3, the following files control and run the wsfClassicCars scraper.

./bin/classicCarsScraperParams.py: This is the configuration file for this App. WSF uses python function invocation as the configuration syntax. Even for those unfamiliar with Python, the syntax is intuitive. You can modify these parameters to your liking.

./bin/scraperClassicCars.py: This is the class that implements the concrete class that scrapes the inputs.

The last invocation in that file:

```
 wsf_config.scrapingProcessor(
    scraperClass=ClassicCars,
)
```

passes:

```
class ClassicCars(wsf_scraperMultipage.ScraperMultipage):
```

to the config machinery.

./bin/scrapeExample.py: This is example of how to run the scraper in full, minimally.

The entire relevant code is:

```
import classicCarsScraperParams
import scraperClassicCars

from unisos.wsf import wsf_parallelProc

if __name__ == '__main__':
    wsf_parallelProc.dispatchWorkersUsingParams()
```

The first two imports bring over the concrete class and set configuration parameters.

The main entry to wsf is wsf_parallelProc.dispatchWorkersUsingParams()

./bin/icmClassicCarsWebScraper.py: This is the preferred way of running this App on the command line.

Running the ICM (Interactive Command Module) by itself as:

```
icmClassicCarsWebScraper.py
```

Gives you a list of commands that you can pick and run.

Choose:

```
 icmClassicCarsWebScraper.py  --load classicCarsScraperParams.py
--load scraperClassicCars.py  -i scrape
```

(run all of it in one line.)

Parameters and the concrete class are first loaded, then the "scrape" command is executed.

For debugging purposes, if needed, you can enable verbosity and callTracking with:

```
icmClassicCarsWebScraper.py -v 1 --callTrackings monitor+ --callTrackings invoke+
 --load classicCarsScraperParams.py --load scraperClassicCars.py  -i scrape
```

(run all of it in one line.)

6 Context And History

I, Mohsen BANAN, have put togther this as a sample of my python code.

It is about writing a scraper for
https://www.oldclassiccar.co.uk/forum/phpbb/phpBB2/viewtopic.php?t=12591

I could use a web scraper development framework for a project that I was doing and decided to make this part of it public.

Here is the process that I went through to put this together in 2020.

7 Initial Web Searches

I first searched the web to see if this, or something similar, has been done before. I found the following relevant pointers:

- https://github.com/nneibaue/yukon_cornelius

  This is a scraper for oldclassiccar.co.uk.

  The design and modeling quality is not great. But the code and some the design is re-usable and I have used it. Later, I'll revisit these.

- https://stackoverflow.com/questions/56211202/attributeerror-str-object-has-no-attribute-keys-when-

  Nothing useful there.

- PyPi Web Scraping Engines/Tools/Packages.

  There are several there. But I did not find any that I liked.

8 The "Web Scraping Development Framework" Model

I decided to build a web scraping development framework and then immediately use it for my own projects and also have it scrape oldclassiccar.co.uk.

Very much by choice, I avoided calling it a "web scraping engine". The domain of web scraping is too broad and too diverse to be reasonably codified as an "engine".

Using web scraping development framework (wsdf), a developer can quickly customize the specifics of a particular site's scraping. The common aspects of web scraping go into wsdf.

9 About unisos.wsf

unisos.wsf is a pip package included in this repo.

It is a generalized scraping framework that can be considered a public resource. There is nothing in wsf which is specific to oldclassiccar.co.uk or any other site in there.

In this case, unisos is just a namespace to avoid name conflicts.

10 About unisos.wsfClassicCars

unisos.wsfClassicCars is also a pip package. unisos.wsfClassicCars uses unisos.wsf.

The code in unisos.wsfClassicCars is very minimal.

Configuration file, the concrete ClassicCars class and the executable are all in the bin directory.

11 About Contents Of This Repo

After untar-ing, you will have two directories.

- wsf
- wsfClassicCars

There are two files that you need to read.

1. wsfClassicCars/py3/README.pdf
2. wsf/py3/README.pdf

12 Installation

I have tested these with Python 3.9. Both packages will likely work fine with earlier Python 3.x release.
Create a fresh virtual environments. Install the two packages by following these instructions:

1. Go to wsf/py3. Follow the instructions in READEME.pdf Section 4.2.
2. Go to wsfClassicCars/py3. Follow the instructions in READEME.pdf Section 4.

The "requires" section of wsf/py3/setup.py enumerates all other package dependencies.
A pip list after the installation should produce something like:

```
Package               Version
--------------------- ---------
beautifulsoup4        4.10.0
certifi               2021.10.8
charset-normalizer    2.0.7
enum34                1.1.10
idna                  3.3
lxml                  4.6.4
pip                   21.3.1
requests              2.26.0
setuptools            58.3.0
soupsieve             2.3.1
unisos.icm            0.25
unisos.ucf            0.15
unisos.wsf            0.1
unisos.wsfClassicCars 0.1
urllib3               1.26.7
wheel                 0.37.0
```

13 About unisos.icm

I want Web Scraping Application (WS-Apps) to function as plug-able modules on the command line interface.
unisos.wsfClassicCars is a WS-App.

ICM (Interactive Command Modules) is a pip package that I have developed. It is similar to "click" but it also supports "–load fileName". fileName can be any python code. This is how wsfClassicCars becomes a plug-able command line module.

Also, the flexibility that ICM provides allows for regression testing of whole or parts of the code. This renders the usual traditions of unit testing obsolete.

14 About COMEEGA And Dynamic Blocks

Parts of my code are written as COMEEGA. COMEEGA stands for "Collaborative Org-Mode Enhanced Emacs Generalized Authorship". Think of it as inverse of Literate Programming. Where the code is also a document. You can switch between code mode and document mode by switching between org-mode and python-mode.

Without emacs and org-mode, such code is not pleasant. I wont use COMEEGA on other people's code.

Dynamic Blocks are a feature of org-mode. What is between +BEGIN: and +END: is controlled with lisp code and will be overwritten if edited.

This allows me to add visible macro capabilities to python.

Both COMEEGA and Dynamic Blocks are mostly used in icmClassicCarsWebScraper.py. If you view that as unpleasant, I suggest that you just consider it as awareness of other powerful ways of doing things ...

15 Design And Implementation Considerations

I did all of this on a rush basis. So, the code is weak in terms of error handling and robustness. But, there is a proper starting point in place and over time it can improve and expand.