SESE Price Scraper Documentation

Release 1.0.0

Pavan Rikhi

Aug 26, 2017

Contents

1 Overview **3**

2 Installation **5**

3 API Documentation **7**
- 3.1 `product` Module 7
- 3.2 `settings` Module 7
- 3.3 `util` Module 8
- 3.4 `sites` Module 8

4 Indices and tables **9**

Python Module Index **11**

Contents:

CHAPTER 1

Overview

SESE's Price Scraper is a Python application that was built to scrape competitor's website for price information. SESE uses this information to determine the degree of our yearly price increases for each variety of seed we have.

When run, the application reads a tab-delimited file containing the SKU#, name, category and organic status of each of our products. It uses this information to create a `Product` object for each SESE variety.

Each `Product` object creates a new object for each website to scrape. These objects are from classes that sub-class the `BaseSite` abstract class, such as the `BotanicalInterests` class. The website classes implement the specific functionality for scraping a single website for a single product.

After every `Product` object has created all it's website objects, the application runs through each `Product` object, creating a tab-delimited file as the output.

In order to add additional websites for the application to scrape, a new website class should be created, sub-classing the `BaseSite` class and overriding it's abstract methods. Next, the *`settings`* module should be edited so that the *`COMPANY_HEADER_ORDER`* setting contains the abbreviation of the new website, and the *`COMPANIES_TO_PROCESS`* setting contains the path to the website's implementation class, for example `sites.botanical_interests.BotanicalInterests`.

CHAPTER 2

Installation

These instructions are for *nix systems, Windows has not been tested but may be supported.

Installing the Price Scraper requires that you have the following software on your system:

- git
- python3
- pip or setuptools

Optionally, virtualenv and virtualenvwrapper will make dependency management and isolation much easier.

These can be installed using your systems package manager. Virtualenv and Virtualenvwrapper can be installed using pip. For example, Arch Linux users can simply run the following:

```
$ sudo pacman -S python pip3 git
$ sudo pip install virtualenv virtualenvwrapper
```

Once you have these dependencies, you should clone the source code repository:

```
$ git clone git@aphrodite.acorn:/srv/git/pricescraper.git pricescraper
$ cd pricescraper
```

This will create a local copy of all the source files. If you have virtualenv installed you should now make a new virtual environment, using your python3 binary:

```
$ mkvirtualenv PriceScraper -p python3
```

You can then install the python dependencies into your system or the new virtual environment using pip and our requirements files:

```
$ pip install -r requirements/develop.txt
```

Use `develop.txt` to include development dependencies, such as the test runner and documentation builder. If you just want to run the application, you may use `requirements/base.txt` instead.

After all dependencies are installed by pip, you should be able to run the application using a valid input file(tab-delimited, containing a header row and columns for SESE SKU, Organic Status, Name and Category):

```
$ python3 pricescraper/price_scraper.py
```

CHAPTER 3

API Documentation

The following page describes the Classes and Functions used by the Price Scraper program.

`product` Module

`settings` Module

This module defines the basic settings used throughout the application.

`settings.`**`ATTRIBUTES_TO_NAMES`** **= {'sese_number': 'SESE SKU', 'weight': 'Weight', 'sese_category': 'SESE Category',**
A dictionary mapping attributes to display names

`settings.`**`ATTRIBUTE_HEADER_ORDER`** **= ('price', 'weight', 'number', 'name', 'organic')**
The output order of each Other Company's attributes

`settings.`**`COMPANIES_TO_PROCESS`** **= ['sites.botanical_interests.BotanicalInterests', 'sites.fedco_seeds.FedcoSeeds', 'sites.j**
The Other Company's to process (by path to Class)

`settings.`**`COMPANY_HEADER_ORDER`** **= ('fe', 'js', 'hm', 'ss', 'ts', 'fs', 'hv', 'bi')**
The output order of the Other Companies (by abbrieviation)

`settings.`**`MINIMUM_NAME_MATCHING_PERCENTAGE`** **= 36**
The minimum percentage of words in common between SESE's and Other Company's Product names for them to be considered a match.

`settings.`**`SESE_HEADER_ORDER`** **= ('sese_number', 'sese_organic', 'sese_name', 'sese_category')**
The output order of SESE attributes

`settings.`**`WORKER_PROCESS_COUNT`** **= 8**
The number of worker processes to create when processing Products.

`util` Module

`sites` Module

The *sites* module contains Classes that describe the website the program will scrape. Searching and Parsing for each website is defined by a class that inherits from the `BaseSite` Class. One object will represent a single Product from the Other Company/Website.

Upon initialization, each Object will visit the Other Company's website, find the Product that best matches the provided Name, Category and Organic Status and scrape the Product's details, such as its name, organic status, model number, weight and price.

The `get_company_attributes()` method from each child Class can be used to retrieve the information about the Other Company's Product.

`base` Module

`botanical_interests` Module

`fedco_seeds` Module

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

settings, 7

Index

A

ATTRIBUTE_HEADER_ORDER (in module settings), 7
ATTRIBUTES_TO_NAMES (in module settings), 7

C

COMPANIES_TO_PROCESS (in module settings), 7
COMPANY_HEADER_ORDER (in module settings), 7

M

MINIMUM_NAME_MATCHING_PERCENTAGE (in module settings), 7

S

SESE_HEADER_ORDER (in module settings), 7
settings (module), 7

W

WORKER_PROCESS_COUNT (in module settings), 7