Code Description Document

For Source code

Web -> Scraper -> Fetch_description

```
def description_from_url_amazon(link):
  """
This is a function to take the user product URL as a link and return the
description of the user product
   :param link:
   :return:Search term/description
   """


def description_from_url_bjs(link):
   """
This is a function to take the user product URL as a link and return the
description of the user product
   :param link:
   :return:Search term/description
   """

def description_from_url_costco(link):
   """
  This is a function to take the user product URL as a link and return the
description of the user product
   :param link:
   :return:Search term/description
   """

def description_from_url_ebay(link):
   """
This is a function to take the user product URL as a link and return the
description of the user product
   :param link:
   :return:Search term/description
   """

def description_from_url_walmart(link):
   """
 This is a function to take the user product URL as a link and return the
description of the user product
   :param link:
   :return:Search term/description
   """
```

Web -> Scraper -> scrap

For amazon:

```
def get_url_amazon(search_term):
    '''
    Take the user's product description
        return the constructed Amazon product url.

    :param search_term: product description
    :return template: Amazon product URL
    '''

def scrap_amazon(driver, search_term):
    '''
    Take the web driver and search_term(user product description)
        scrap the Amazon product page and return the list of items found.

    :param driver: instance of  webdriver
    :param search_term: product description
    :return results: list of items found
    '''

def extract_item_amazon(driver, search_term):
    '''
    Take the web driver and search_term(user product description),
        return child dictionary of 1st item found the scraped product list.

     :param driver: instance of webdriver
     :param search_term: product description
     :return result: product dictionary
'''
```

For BJ's

```
def get_url_bjs(search_term):
    '''
    Take the user's product description
        return the constructed Bjs product url.

    :param search_term: product description
    :return url: bjs product URL
    '''

def scrap_bjs(driver, search_term):
    '''
```

```
    Take the web driver and search_term(user product description)
        scrap the Amazon product page and return the list of item found.

    :param driver: instance of webdriver
    :param search_term: product description
    :return results: list of items found
    '''

def extract_item_bjs(driver, search_term):

    '''
    Take the web driver and search_term(user product description),
        return child dictionary of 1st item found the scraped product list.

    :param driver:instance of chrome webdriver
    :param search_term:product description
    :return result:product details dictionary
    '''
```

For Costco

```
def get_url_costco(search_term):
    '''
    Take the user's product description
        return the constructed Amazon product url.

    :param search_term:product description
    :return url:specific product URL
    '''

def scrap_costco(driver, search_term):
    """
    Take the web driver and search_term(user product description)
        scrap the Amazon product page and return the list of items found.

    :param driver: instance of  webdriver
    :param search_term: product description
    :return results: list of items found
    """

def extract_item_costco(driver, search_term):
    """
    Take the web driver and search_term(user product description)
        scrap the Amazon product page and return the list of items found.

    :param driver:instance of chrome webdriver
    :param search_term:product description
    :return result:product details dictionary
```

```
    """
```

For ebay:

```
def get_url_ebay(search_term):
    '''
    Take the user's product description
        return the constructed Amazon product url.

    :param search_term:product description
    :return url:specific product URL
    '''

def scrap_ebay(driver, search_term):
    """
    Take the web driver and search_term(user product description)
        scrap the Amazon product page and return the list of items found.

    :param driver: instance of chrome webdriver
    :param search_term: product description
    :return results: html code
    """

def extract_item_ebay(driver, search_term):
    """
    Take the web driver and search_term(user product description)
        scrap the Amazon product page and return the list of items found.

    :param driver:instance of chrome webdriver
    :param search_term:product description
    :return result:product details dictionary
    """
```

For Walmart

```
def get_url_walmart(search_term):
    '''
    Take the user's product description
        return the constructed Amazon product url.

    :param search_term:product description
    :return url:specific product URL

    '''

def scrap_walmart(driver, search_term):
    """
    Take the web driver and search_term(user product description)
        scrap the Amazon product page and return the list of items found.
```

```
    :param driver: instance of webdriver
    :param search_term: product description
    :return results: list of items found

    """

def extract_item_walmart(driver, search_term):
    """
    Take the web driver and search_term(user product description)
        scrap the Amazon product page and return the list of items found.

    :param driver:instance of webdriver
    :param search_term:product description
    :return result:product details dictionary

    """
```

Web -> Scraper -> Web Scraper

```
def get_driver():
    """
    This function provides the instance of a chrome and firefox driver to get
the request

    :return:  instance of Chrome and firefoxWebDriver.
    """

def set_results(to, from_):
    """
    sets the main result dictionary from the child result dictionary(results
from each scraped site.)

    :param to: main dictionary
    :param from_: child dictionary
    :return: None
    """

def search_amazon(driver, description, results):
    """
    Get the details of 1st picked item(in child dictionary)

    :param driver: instance of Chrome WebDriver
    :param description: search term/ product description.
    :param results: reference to set_results(to)
    :return: None
    """
```

```
def search_bjs(driver, description, results):
    """
    Get the details of 1st picked item(in child dictionary)

    :param driver: instance of Chrome WebDriver
    :param description: search term/ product description
    :param results: reference to set_results(to)
    :return: None
    """

def search_walmart(driver, description, results):
    """
    Get the details of 1st picked item(in child dictionary)

    :param driver:instance of Chrome WebDriver
    :param description:search term/ product description
    :param results: reference to set_results(to)
    :return: None
    """

def search_costco(driver, description, results):
    """
    Get the details of 1st picked item(in child dictionary)

    :param driver:instance of Chrome WebDriver
    :param description:search term/ product description
    :param results: reference to set_results(to)
    :return: None
    """

def search_ebay(driver, description, results):
    """
    Get the details of 1st picked item(in child dictionary)

    :param driver:instance of Chrome WebDriver
    :param description:search term/ product description
    :param results: reference to set_results(to)
    :return: None
    """

def scraper(link):
    """
    takes the user product URL,
        looks for the the website in the URL, extract the description from it,
and
            scrap the other sites, finally return the main result dictionary
with product details of each scrapped site.

    :param link: user product url
```

```
    :return results: main result dictionary.
    """
```

Web -> Scraper -> Server_api

```
def post(self):
    """
    Accepts user's requests from popup.js extension and returns the main result
dictionary with entries for the items found on different scraped sites.
    The main result dictionary will be read and rendered on the extension.

    :return results: main result dictionary with 200| "" with 404 | custom
message with 404.
    """
```

For Test cases:

Tests->test_end_to_end

For user on amazon:

```
def test_scrapper_amazon_result():
    '''
    For a given user product url
        tests whether the returned dict has at least 1 entry from the 4 scraped
sites :ebay,costco,walmart,bjs

    :return: None
    '''

def test_scrapper_amazon_result_len():
    '''
    For a given user product url
        tests whether the returned dict has 4 entries respective to each
scraped sites :ebay,costco,walmart,bjs

    :return: None
    '''

def test_scrapper_amazon_result_description_count():
    '''
    For a given user product url
        tests whether the returned dict-key{description} at least 1 entry from
the 4 scraped sites :ebay,costco,walmart,bjs

    :return: None
```

```
    '''
```

For user on Bj's

```
def test_scrapper_bjs_result1():
    '''
    For a given user product url
        tests whether the returned dict has at least 1 entry from the 4 scraped
sites :ebay,costco,walmart,amazon

    :return:
    '''

def test_scrapper_bjs_result_len1():
    '''
    For a given user product url
        tests whether the returned dict has 4 entries respective to each
scraped sites :ebay,costco,walmart,amazon

    :return: None
    '''


def test_scrapper_bjs_result_description_count():
    '''
    For a given user product url
        tests whether the returned dict-key{description} at least 1 entry from
the 4 scraped sites :ebay,costco,walmart,amazon

    :return: None
    '''
```

For user on Costco

```
def test_scrapper_costco_result1():
    '''
    For a given user product url
        tests whether the returned dict has 4 entries respective to each
scraped sites :ebay,amazon,walmart,bjs

    :return: None
    '''

def test_scrapper_costco_result_len1():
    '''
    For a given user product url
        tests whether the returned dict has 4 entries respective to each
scraped sites :ebay,amazon,walmart,bjs
```

```
    :return: None
    '''

def test_scrapper_costco_result_len2():
    '''
    For a given user product url
        tests whether the returned dict has 4 entries respective to each
scraped sites :ebay,amazon,walmart,bjs

    :return: None
    '''
```

For Walmart

```
def test_scrapper_walmart_result():
    '''
    For a given user product url
        tests whether the returned dict has at least 1 entry from the 4 scraped
sites :ebay,costco,walmart,bjs

    :return: None

    '''

def test_scrapper_walmart_result_len():
    '''
    For a given user product url
        tests whether the returned dict has 4 entries respective to each
scraped sites :ebay,costco,walmart,amazon

    :return: None
    '''
```

Tests->test_units

Fetch_description_amazon

```
def test_fetch_description_amazon1():
    '''
    tests whether the description string extraction from a user product URL is
working okay.

    :return: None
```

Fetch_description_BJs

```
def test_fetch_description_bjs1():
    '''
```

```
    tests whether the description string extraction from a user product URL is
working okay.

    :return: None
    '''
```

Fetch_description_Costco

```
def test_fetch_description_costco1():
    '''
    tests whether the description string extraction from a user product URL is
working okay.

    :return: None
    '''
```

Fetch_description_walmart

```
def test_fetch_description_walmart1():
    '''
    tests whether the description string extraction from a user product URL is
working okay.

    :return: None
    '''
```

Get_drivers

```
def test_get_driver():
    '''
    Tests whether it returns instances of selenium.webdrivers such as
        selenium.webdriver.chrome.webdriver.WebDriver
        selenium.webdriver.firefox.webdriver.WebDriver

    :return: None
    '''
```

Scraper_amazon

Tests individual scraping functions of Amazon!

```
def test_get_url_amazon_1():
    '''
    For a given product/item description
        tests whether the url building happens correctly.

    :return: None
    '''
```

```
def test_scrap_amazon():
    '''
    For a given product/item description
        tests whether the returned list has at least 1 item found from the
scraped site :Amazon

    :return: None
    '''
```

Scraper_BJ's

Tests individual scraping functions of Bjs!

```
def test_get_url_bjs_1():
    '''
    For a given product/item description
        tests whether the url building happens correctly.

    :return: None
    '''

def test_scrap_bjs():
    '''
    For a given product/item description
        tests whether the returned list has at least 1 item found from the
scraped site :Bjs

    :return: None
    '''
```

Scraper_costco

Tests individual scraping functions of Costco!

```
def test_get_url_costco_1():
    '''
    For a given product/item description
        tests whether the url building happens correctly.

    :return: None
    '''

def test_scrap_costco():
    '''
    For a given product/item description
        tests whether the returned list has at least 1 item found from the
scraped site :Costco
    :return: None
```

```
    '''
```

scraper_Walmart

Tests individual scraping functions of Walmart!

```
def test_get_url_walmart_1():
    '''
    For a given product/item description
        tests whether the url building happens correctly.

    :return: None
    '''
```

tests/__init__.py

```
def setup_get_driver_details():
    """
    This function provides the instance of a chrome and firefox driver to get
the request

    :return:  instance of Chrome and firefoxWebDriver.
    """
```