Read the Docs Template Documentation

Release 1.0

Read the Docs

August 11, 2015

Contents

Contents:

CHAPTER 1

Authors

- Ajay Kumar Soma

CHAPTER 2

Idea backgorund - The Existing online discount sites

Edit this doc online

2.1 All discounts and Deals at one place

There are many sites on the net which show all the discounts from eretailers in one place. Make them searchable and accessible to price conscious consumers who like deals and discounts. But the same kind of service is not available for the offline or in-store deals and discounts. I guess the following are the reasons

2.2 sample sites

2.3 1. Item level discount details

- cashkaro => all discounts
- findyogi => only for mobiles

2.4 2. All coupons with out item info

- cupondunia - All cupons
- savemyrupee - All cupons It has wish list, if you give an item, they will find lowest price for you

2.5 online deals advantage

1. online retailers provide apis. so easy to get data.
2. Online retailers pay referral bonus

2.6 offline deals disadvantages

1. too many interfaces and no api support.

2. difficult to earn referral bonus.

CHAPTER 3

Our Idea

To bring all offline discounts and deals to online just like the online discount sites.

3.1 challenges - our approach

1. Product data - use webscraper
2. Store data - build very simple publishing system for each store
3. Referral bonus - right now focus on reaching consumers, later plan for revenue from,
    - Ads
    - Referral
    - Analytic data

3.1.1 Our unique features

0. Discounts Guide

A discounts guide that tells you why the price is marked down.

1. Deal Rating

Also a rating for each discount which tells whether discount is good or not

2. Use Likes and Comments

Like/dislike and comments on each item and over all sale.

3. Reserve item

User can reserve an item for a day or 2 days and collect it from the store.

4. Wish List

User can make a wish list for an item at a certain price or discount percentage and we will send alerts

5. *Location based Discount alerts*

User will get alerts when he is near a store which matches his wish list. This is the most important feature.

6. Offline and Online price comparison

Compare the price offline, if the prod is avail online, show the comparison and give the customer a reason to visit the store.

CHAPTER 4

Installation

Install the package with pip:

```
$ pip install read-the-docs-template
```

CHAPTER 5

Template

$project will solve your problem of where to start with documentation, by providing a basic explanation of how to do it easily.

Look how easy it is to use:

```
import project # Get your stuff done project.do_stuff()
```

5.1 Features

- Be awesome
- Make things faster

5.2 Installation

Install $project by running:

```
install project
```

5.3 Contribute

- Issue Tracker: github.com/$project/$project/issues
- Source Code: github.com/$project/$project

5.4 Support

If you are having issues, please let us know. We have a mailing list located at: project@google-groups.com

5.5 License

The project is licensed under the BSD license.

CHAPTER 6

Scraper

The Scraper is the heart of our app. It fetches the data and pumps to the app. So the scraper design is very critical. I have considered several tools and technologies for scraping.

I have done scraping in node.js and python. **Python** is my choice as it is very simple and follows **sequential programming** approach.

The following explains the scraping browsers and frameworks

6.1 1. Scrapy-Python

Scrapy is one of the best framework for scraping. But it can't scrape dynanmic sites where the content is generated on the fly by java script. because it uses headless browser.

6.2 2. Selenium-xvfb

Selenium uses real browser so it can scrape dynamic sites. But is bit slow. It doesnt matter. it does the job pretty good. XVFB is emulator to run browser in hidden mode.

CHAPTER 7

Puma India scraping

It is a dynamic site. But the initial page is static. so for the first page, we used **scrapy**. for products page used **selenium**.

7.1 Puma-lev1

its job is to fetch all the unique product pages along with the following fields.

1. id
2. process_lev1
3. process_lev2
4. sale
5. name
6. **url**
7. image_small
8. regular_price
9. discounted_price

command to run ::

cd puma-spider/stack/stack/ scrapy crawl puma-lev1

7.1.1 Program Logic

Fetch the data from the site and while saving it, do the following.

1. If the item is not present (check using **URL** as key), Then **Insert** it.
2. **If item exists in the DB, then update the following fields.**
   - sale
   - regular_price
   - discounted_price

7.2 Puma-lev2

lev2-sample-link

command to run :: python puma-lev2.py refresh=N

python puma-lev2.py refresh=Y

The level2 spider reads yields the following fields.

1. images - original, big, small
2. style number
3. availability
4. size

Keep a refresh flag for this program. if refresh flag is Y, then **process all** else process only those with **process_lev2** flag **N**

CHAPTER 8

Indices and tables

- genindex
- modindex
- search