9GAG-Reader

Usage

- Just visited link: https://a-9gag-reader.herokuapp.com/

Explore

- At first, I explored the 9gag-rss and 9gag websites to see what the feeds looked like and selected 4 feeds in my project: Awesome, Dark Humor, Funny, and Hot. Besides, I also noticed that too many requests for the 9gag would be banned and the author said the website will update every 2 minutes. Then, I started to search the existing python library for RSS parser, after comparing with some libraries, I decided to use the feedparser since it supported multiple RSS formats, the format I used is ATOM 1.0 format. Based on the your suggestion, I chose the cache to store the data instead of the database. To cache the data in the web application, I used the Flask-Caching. The web application I used in this project is the Flask.

Idea

- Since the size of the data received from each request to the RSS feed is kind of large and the update frequency is not high, it is a waste of bandwidth to send request to the RSS feed every time refreshing the page or many people visit the page in the meantime.
- To solve this problem, I think about the HTTP Conditional Get to check whether the RSS feed is updated or not based on the response header instead of get the whole brunch of data. The information needs to be checked includes: Last-Modified and ETag. However, after I parsed the RSS feed got from 9gag-rss, neither of them were provided in the response header.
- Thus, I had to hard code the cache time-out to solve the problem. The time-out I set was 120 seconds since the author of 9gag-rss said the website updates every 2 minutes. Thus, if the time between two requests to my application less than 120 seconds, the cached pages will be returned, otherwise, the new request to 9gag-rss will be sent and my website will show the new content.

Front-end

- To show the feed in a proper way, I tried lots of layouts. Since the size of different content feeds are different, I first tried to present them in the water fall way, like the Pinterest. The javascript library I used is Masonry, however, I tried my best to do experiments, the effect is not good enough. Thus I just simply used break line to separate each feed content and presented them one by one in each row.
- The whole appearance of my website is based on my last project: StanceComp(Research project), and its original idea is based on freelancer, and the logo is credited to the Matt Niblock.
- To present each feed content in a beautiful way, I put the fedd content and feed title in the card from the Bootstrap. However, some feed title is too long and the size of card will automatically fit it. To solve this problem, I used the width attribute from video feed content to set up the size of each card, but the image feed content does not provide this information, thus I send request to download the images based on the src attribute from image, then use the Image class from the Pillow library to get the width of the image. Though this way could fix the card size problem, the time to load page is kind of long.

Test

- I used the Unittest framework to test my application. I mainly tested whether 5 requests: main page, and 4 feed pages.
- `python test.py`

Deployment

- To deploy my application on web server, I used the Heroku, a cloud application platform.