Report. Contemporary Software Development project. Developing a housing parser

Mikhail Boronin

May 6, 2019

Contents

1 Design **2**

2 Development **2**
2.1 Time issue 2
2.2 Data Issue 2
2.3 API Issues 3

3 Testing **3**
3.1 Testing in a multiple environment 3
3.2 Travis CI 3

4 Logging **4**

5 What I have learned **4**

6 Future Development **4**
6.1 Docker 4
6.2 Data optimization 4
6.3 Other sources 4
6.4 Machine learning 5
6.5 Location 5
6.6 Users 5

Abstract

This report is devoted to housing parsing tool. It describes first iteration of the tool, which is able to parse housing portals in Uppsala and present them via Telegram bot. Tool is developed in Python 3. MySQL 8 database is used for data storage. Code is covered with unit and integration tests. Code coverage is measured with coverage tool for Python. Git is used as version control system. Link to the github repository is `https://github.com/mboronin/housing_bot`.

I have selected this project, since I do not have a lot of experience in parsing web parsing and bot creating, therefore I considered it as a good idea to get a hands-on experience in development of such tools.

1 Design

To develop a design for this software project, I have decided to inspect the code of the most famous portal: `bostad.uppsala.se`. I have noticed, that code contains a lot of tags on the page with available housing. Those tags contain all object information, so there is no need to open and parse a page related to one specific apartment. Bot is supposed to me located and executed on a server, to connect to telegram API `https://core.telegram.org/` it uses token, which is provided by Telegram messenger via developers tools. python-telegram-bot framework was used as a main tool during the development process `https://github.com/python-telegram-bot/python-telegram-bot`.

2 Development

I have tried working with lxml, which is a quite popular library for Python web-parsing. However, after reading some articles, I have realised, that BeatifulSoup is a better and modern approach. It is a framework, which includes lxml inside of it, however it also contains additional features, which made it easier to implement the bot. It also makes the solution more scalable. So, I began with the link `https://gist.github.com/yosemitebandit/1805918`, where basic Beautiful Soup template is introduced.

2.1 Time issue

Time in this particular example is stored in a quite unambiguous way, it is a time from 1 Jan of 0 year, but it still has some small delta, which I had to count in order to get correct time in a human readable format.

2.2 Data Issue

There also was an issue with ids, since the ids used by website developers were unreasonably big, I have decided to create new set of ids to keep track of records. However it was easy to develop using MySQL. I have realized that adress cannot be used as a unique key in a database since not all the published object specify the exact address of the property. Sometimes only building number and street name are provided, which of course are not unique. I had to use two separate classes for importing of data and data representation, even though they represent the same object: appartment. However, sicne the data is represented in different way during these two

processes and it is not possible to implement more than one constructor for a Python class, I have decided to work with two different types of objects, which actually makes code more readable.

2.3 API Issues

It also turned out, that Telegram API does not have support for both Markdown and HTML tables for messages, which gave me problem in displaying table of all available housing. This is now solved with <pre> tag, which allows to format text, so it looks structured. However, I want to find a better way of implementing this view.

3 Testing

For testing I have decided to use a combination of unit tests and integration tests, which seems a reasonable approach to me. Code coverage also will be added in next iteration since it is imprortant for understanding how well code is tested. Pytest is used as the main framework for unit tests. Each project module has its separate tests file, where each function is tested. Some sort of integration testing is only developed with Travis CI so far, since most of the integration test solutions for Python are built for website testing, which is not the case for this project. However, pyramid framework will be used for developing some extra integration tests. Currently I am looking for a reason why those tests are failing in my project, therefore I do not include info about it in my report.

3.1 Testing in a multiple environment

Testing with tox is not yet implemented, however it is very important for Python, since code is often ran in different environments and it is quite insecure to keep tests only for one environment. However, Travis CI configuration uses several Python versions, which makes it create a couple of different environments, which seems to be sufficient for the beggining.

3.2 Travis CI

For integration testing I have decided to use Tracis CI since it is the most common tool nowadays and it is easy to set it up with a student repository. I have created the travis file and put it into project root directory. The tests are run automatically after every commit, which becomes a trigger in this case. Code is tested for several Python versions (3.4, 3.5, 3.6). Later, it will be automatically deployed into a docker container and relaunched on the server.

4 Logging

Simple Python logging is used in this application. Python logs should be separated into two different flows: debug and info. Log file should also be created in order to keep track of execution history. Logs are used mostly for tool debugging, which is very important during bot developing, since it has quite a complex function call structure.

5 What I have learned

I have became familiar with Handlers and callback usage. Also, I gained some experience in exploring and using API and 3rd party framework in combination, which required me to combine two sources of information in order to make solution work. I had a huge experience reading Python test suite, pytest, which is the most commonly used framework to develop tests for python applications. I have decided not to use suites developed for telegram bots in Python since they are not maintained recently and are most likely outdated. I have used Python + MySQL 8.0, which is also a new combination of tools for me. I have been managing my project using Trello and it simplified the process of task prioritezation and allowed to develop a time plan for future project development.

6 Future Development

6.1 Docker

I have tested my project on the server, but I still want to make it easier to deploy, therefore I plan to dockerize my solution and make the process fully automated, so new version is automatically pulled from master branch on every push.

6.2 Data optimization

Data for now is stored in a single table, which of course is not the best approach according to normalization rules. This will be improved in the next development iteration. Converting code to usage of SQL Alchemy is also planned as it is a must-have for all modern Python projects. It simplifies the code and allows to keep better control of errors and exceptions.

6.3 Other sources

Other sources like blocket and nationsgardarna will be added. Everything is ready to add nationsgardarna, since it uses same developing approach,

but for blocket and studentboet other approach is required and it will be developed in the next iteration

6.4 Machine learning

Machine learning feature is the next step after adding sources with unverified ads, like blocket and studentboet. Machine learning will be used for scam recognition and labeling.

6.5 Location

Location feature will be added to the bot, so user will have an opportunity to sort and filters offers based on location on the map. Also, displaying map with available housing will be implemented.

6.6 Users

Right now user data is stored in DB in unencrypted way, which is very unsafe. Therefore, algorithm for encrypting and decrypting data back needs to be implemented, so user data is never stored in raw format.