

Qikai's Test Plan

Qikai Yang

March 2021

Contents

1	Introduction	2
2	Structure of the code	2
3	OS support	3
4	Tools	3
5	Environment	3
6	Test View.py	3
6.1	Test initialization	4
6.2	Test Choice 1	4
6.3	Test Choice 2	5
6.4	Test Choice 3	6
6.5	Test Choice 4	7
6.6	Test Choice 5	7
7	Test Controller.py	8
8	Test Processor.py	9
9	Test API	9
9.1	Test PUT	9
9.2	Test POST	10
9.3	Test DELETE	10
9.4	Test Get	10

1 Introduction

This document is to record how I test my UI under MVC model. The tests mainly contain 3 parts. The first part is testing the View. The second part is testing the Controller. The third part is to test the Processor.

2 Structure of the code

The structure of my code is strictly following the guide of MVC model. The Processor.py serves the function of a server. The Controller.py serves the function of a controller. The View.py serves the function of displays for users. A detailed graph can be seen as below:

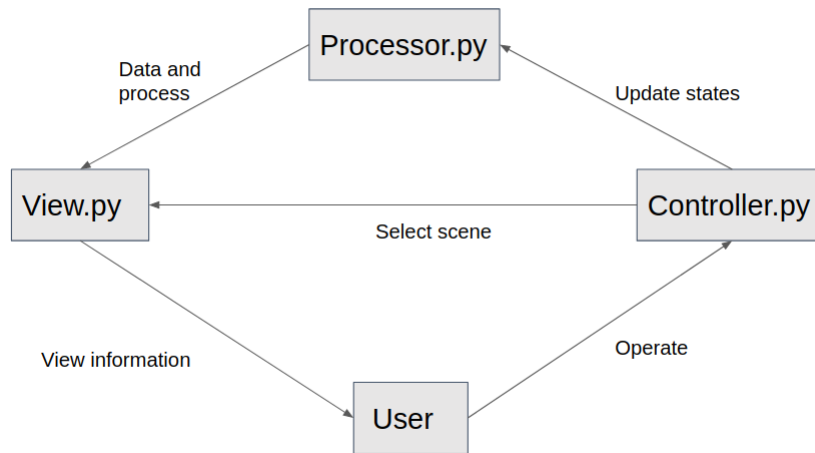


Figure 1: The structure of my code under MVC model.

Therefore, I will divide my test into three parts. The first part is testing the View. The second part is testing the Controller. The third part is to test the Processor.

3 OS support

The following OS configurations should be used when testing:

- **Ubuntu 20.04**

4 Tools

The following tool should be used when testing:

- **python 3.7**
- **PyCharm**

5 Environment

The following libraries is expected for testing:

- **firebase and its dependency**
- **beautiful soup**

6 Test View.py

The structure of my View mainly have 3 aspects along with many printing helper functions. A detailed graph can be seen as below:

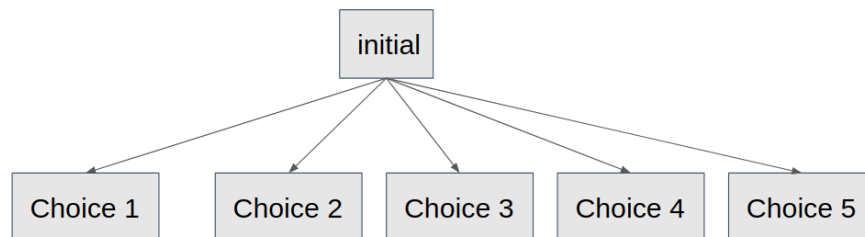
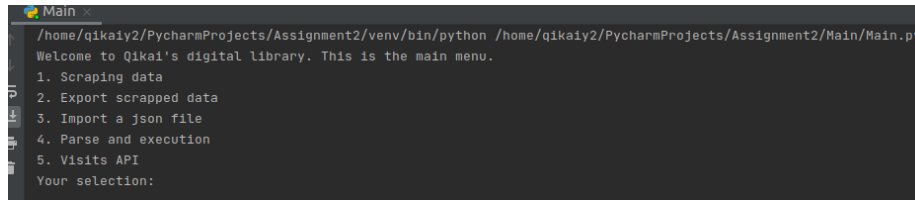


Figure 2: The structure of my View.

Therefore, the tests for this part are divided into several aspects.

6.1 Test initialization

The view of initialization is shown in the figure below.



```

Main
/home/qikaiy2/PycharmProjects/Assignment2/venv/bin/python /home/qikaiy2/PycharmProjects/Assignment2/Main/Main.py
Welcome to Qikai's digital library. This is the main menu.
1. Scraping data
2. Export scrapped data
3. Import a json file
4. Parse and execution
5. Visits API
Your selection:
```

Figure 3: View of the initialization stage.

Tests include:

1. Test all choices are listed
2. Test if warning will be given when input choice is invalid
3. Test all information are correctly shown on the terminal
4. Test if all variables are correctly set to assigned values after the user input the choice.
5. Test given choice 1, it can be switched to Choice 1's view correctly.
6. Test given choice 2, it can be switched to Choice 2's view correctly.
7. Test given choice 3, it can be switched to Choice 3's view correctly.

6.2 Test Choice 1

The view of Choice 1 is shown in the figure below:

```
Welcome to Qikai's digital library. This is the main menu.
1. Scraping data
2. Export scrapped data
3. Import a json file
Your selection:1
Please input your start url
Your start url:xxx
Please input number of books
Number of books to be scrapped:
```

Figure 4: View of the choice 1 stage.

Tests include:

1. Test all choices are listed
2. Test if a warning will be given when input url is invalid
3. Test if a warning will be given when the number of scrapped books exceeds 200
4. Test if a warning will be given when the number of scrapped authors exceeds 50
5. Test if all variables are correctly set to assigned values after the user input the choice.
6. Test if the scrapper starts to work smoothly given a correct URL and appropriate number of books and authors.

6.3 Test Choice 2

The view of Choice 2 is shown in the figure below:

```
Welcome to Qikai's digital library. This is the main menu.
1. Scraping data
2. Export scrapped data
3. Import a json file
Your selection:2
What kind of data are you trying to import? 0 for author. 1 for book.
Your selection. Export Book or Author?:
```

Figure 5: View of the choice 2 stage.

Tests include:

1. Test all choices are listed
2. Test if a warning will be given when input choice is invalid
3. Test if all variables are correctly set to assigned values after the user input the choice.
4. Test if the data is correctly exported after the user input the choice.

6.4 Test Choice 3

The view of Choice 3 is shown in the figure below:

```
Welcome to Qikai's digital library. This is the main menu.
1. Scraping data
2. Export scrapped data
3. Import a json file
Your selection:3
What kind of data are you trying to import? 0 for author. 1 for book.
Your selection:0
What is the address of your input json file?
Address to json file:
```

Figure 6: View of the choice 3 stage.

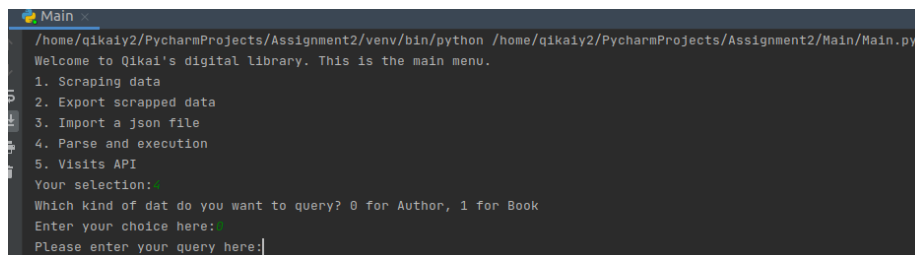
Tests include:

1. Test all choices are listed
2. Test if a warning will be given when input choice is invalid

3. Test if a warning will be given when input address is invalid
4. Test the data is correctly imported after the user input the choice and address.

6.5 Test Choice 4

The view of Choice 4 is shown in the figure below:



```
Main ~
/home/qikaiky2/PycharmProjects/Assignment2/venv/bin/python /home/qikaiky2/PycharmProjects/Assignment2/Main/Main.py
Welcome to Qikai's digital library. This is the main menu.
1. Scraping data
2. Export scrapped data
3. Import a json file
4. Parse and execution
5. Visits API
Your selection:
Which kind of dat do you want to query? 0 for Author, 1 for Book
Enter your choice here:
Please enter your query here:
```

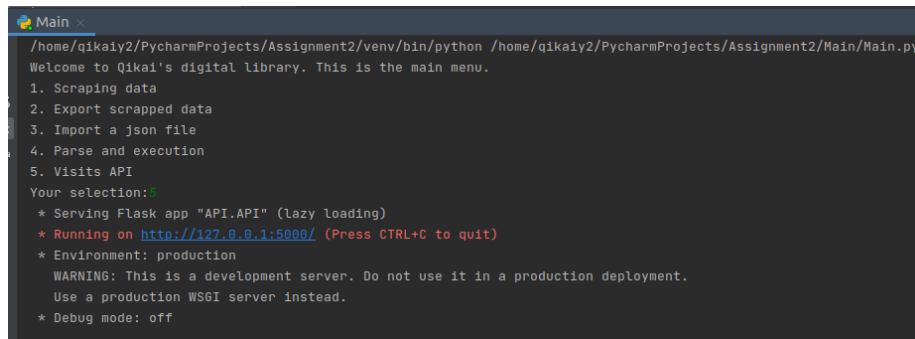
Figure 7: View of the choice 4 stage.

Tests include:

1. Test all choices are listed
2. Test if a warning will be given when input choice is invalid
3. Test if a warning/error will be given when input query is invalid
4. Test the data is correctly updated after the user input the choice and query.

6.6 Test Choice 5

The view of Choice 5 is shown in the figure below:



```

Main x
/home/qikaiky2/PycharmProjects/Assignment2/venv/bin/python /home/qikaiky2/PycharmProjects/Assignment2/Main/Main.py
Welcome to Qikai's digital library. This is the main menu.
1. Scraping data
2. Export scrapped data
3. Import a json file
4. Parse and execution
5. Visits API
Your selection: 5
* Serving Flask app "API.API" (lazy loading)
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off

```

Figure 8: View of the choice 5 stage.

Tests include:

1. Test all choices are listed
2. Test if a warning will be given when input choice is invalid
3. Test the data is correctly updated (i.e. server starts to run) after the user input the choice.

7 Test Controller.py

1. Test if function begin() can pop out correct view
2. Test if controller pops out correct view, given choice 1
3. Test if controller pops out correct view, given choice 2
4. Test if controller pops out correct view, given choice 3
5. Test if controller updates Process's status correctly when user chooses choice 1.
6. Test if controller can catches the user's actions correctly given choice 1
7. Test if controller can catches the user's actions correctly given choice 2

8. Test if controller can catches the user's actions correctly given choice 3
9. Complex test on the whole pipeline of the function of main_process.

8 Test Processor.py

1. Test if scrapper branch works smoothly
2. Test if export branch works smoothly
3. Test if import branch works properly
4. Test if the scrapper will stop scrapping authors when number of scrapped authors reaches the specified number.
5. Test if the scrapper will stop scrapping books when number of scrapped books reaches the specified number.
6. Test if the scrapper communicates with the database smoothly.

9 Test API

To test API, we need to test 4 modules separately.

9.1 Test PUT

By using **postman** to send requests, I will mainly test the conditions below.

1. Test if the book is updated after inputting a valid ID query
2. Test if the author is updated after inputting a valid ID query
3. Test if an error is given after inputting an invalid ID query
4. Test if an error is given after inputting an invalid ID query

9.2 Test POST

By using **postman** to send requests, I will mainly test the conditions below.

1. Test if the book is put after inputting a valid ID query
2. Test if the author is put after inputting a valid ID query
3. Test if an error is given after inputting an invalid ID query
4. Test if an error is given after inputting an invalid ID query

9.3 Test DELETE

By using **postman** to send requests, I will mainly test the conditions below.

1. Test if the book is deleted after inputting a valid ID query
2. Test if the author is deleted after inputting a valid ID query
3. Test if an error is given after inputting an invalid ID query
4. Test if an error is given after inputting an invalid ID query

9.4 Test Get

1. Test if a valid list of books is returned after inputting a valid ID query
2. Test if a valid list of authors is returned after inputting a valid ID query
3. Test if an error is given after inputting an invalid ID query
4. Test if an error is given after inputting an invalid ID query