

LAB1 - EnglishPal Dependency Analysis and Dependency Graph

Author: 占健豪, 王彦超, 陈致远, 汤佳伟

Date: 2021/5/17

Location: 22-206

Introduction

EnglishPal is a website application dedicated to helping people improve their English. This lab study help us understand the current health level of the architecture of EnglishPal.

Materials and Methods

The module-level dependencies are captured by snakefood, and the class/function-level dependency graph for EnglilshPal is hand-drawn and can be plotted by Mermaid.

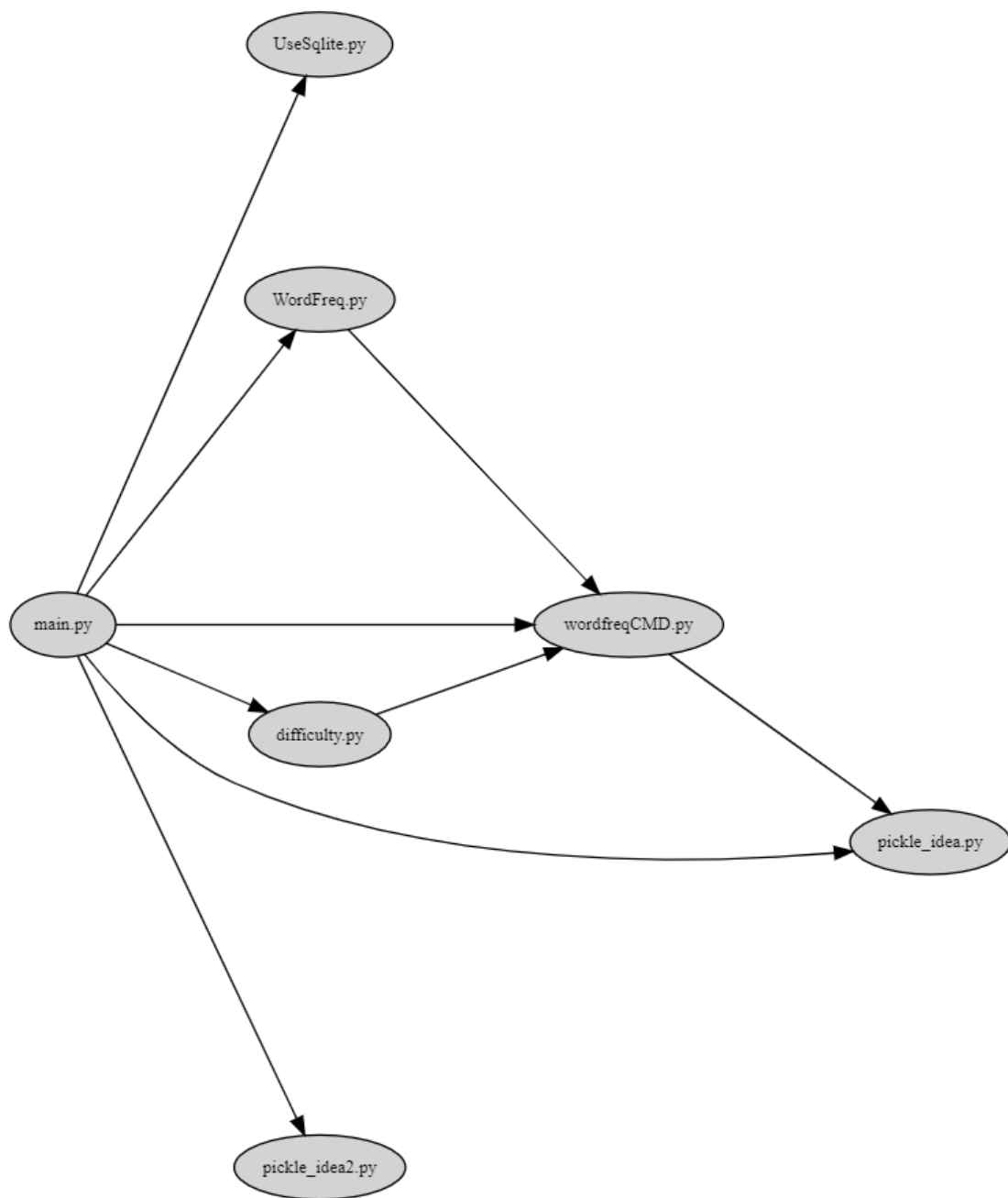
Results

1. EnglishPalModule.dot:

```

1  strict digraph "dependencies" {
2      graph [
3          rankdir="LR",
4          overlap="scale",
5          size="8,10",
6          ratio="fill",
7          fontsize="16",
8          fontname="Helvetica",
9          clusterrank="local"
10     ]
11     node [
12         fontsize=10
13         shape=ellipse
14         // style=filled
15         // shape=box
16     ];
17
18     "UseSqlite.py" [style=filled];
19     "WordFreq.py" [style=filled];
20     "WordFreq.py" ->
21     "wordfreqCMD.py";
22     "difficulty.py" [style=filled];
23     "difficulty.py" ->
24     "wordfreqCMD.py";
25     "main.py" [style=filled];
26     "main.py" -> "UseSqlite.py";
27     "main.py" -> "WordFreq.py";
28     "main.py" -> "difficulty.py";
29     "main.py" -> "pickle_idea.py";
30     "main.py" -> "pickle_idea2.py";
31     "main.py" -> "wordfreqCMD.py";
32     "pickle_idea.py" [style=filled];
33     "pickle_idea2.py" [style=filled];
34     "wordfreqCMD.py" [style=filled];
35     "wordfreqCMD.py" ->
36     "pickle_idea.py";
37 }

```



2. class/function-level.txt

```
1  graph LR
2
3  load_freq_history --> pickle_idea.load_record
4  verify_user --> Sqlite3Template.RecordQuery
5  add_user --> Sqlite3Template.InsertQuery
6  check_username_availability -->
7  Sqlite3Template.RecordQuery
8  get_expiry_date --> Sqlite3Template.RecordQuery
9  get_today_article --> Sqlite3Template.RecordQuery
10 get_today_article --> load_freq_history
11 get_today_article --> difficulty.get_difficulty_level
12 get_today_article --> user_difficulty_level
13 get_today_article --> random.shuffle
14 get_today_article --> random.choice
15 get_today_article --> text_difficulty_level
16 get_today_article --> within_range
17 get_today_article --> get_answer_part
18 mark_word --> load_freq_history
19 mark_word --> pickle_idea.dict2lst
20 mark_word --> pickle_idea.merge_frequency
21 mark_word --> pickle_idea.save_frequency_to_pickle
22 mainpage --> WordFreq
23 mainpage --> load_freq_history
24 mainpage --> pickle_idea.dict2lst
25 mainpage --> pickle_idea.merge_frequency
26 mainpage --> pickle_idea.save_frequency_to_pickle
27 mainpage --> pickle_idea.dict2lst
28 user_mark_word --> load_freq_history
29 user_mark_word --> pickle_idea2.dict2lst
30 user_mark_word --> pickle_idea2.merge_frequency
31 user_mark_word --> pickle_idea2.save_frequency_to_pickle
32 userpage --> WordFreq
33 userpage --> pickle_idea.load_record
34 userpage --> load_freq_history
35 userpage --> sort_in_descending_order
36 signup --> check_username_availability
37 signup --> render_template
38 signup --> add_user
39 signup --> verify_user
40 login --> render_template
41 login --> verify_user
42
43 difficulty.load_record --> pickle.load
44 difficulty.difficulty_level_from_frequency --> math.log
45 difficulty.get_difficulty_level --> revert_dict
46 difficulty.get_difficulty_level -->
47 sort_in_ascending_order
48 difficulty.text_difficulty_level --
49 >sort_in_descending_order
50
51 pickle_idea.merge_frequency --> pickle_idea.lst2dict
52
53 pickle_idea2.merge_frequency --> pickle_idea2.lst2dict
54
55 Sqlite3Template.do --> Sqlite3Template.connect
56 Sqlite3Template.do --> Sqlite3Template.instructions
57 Sqlite3Template.do --> Sqlite3Template.operate
```

```

graph LR
    load_freq_history --> pickle_idea.load_record
    verify_user -->
    SQLite3Template.RecordQuery.add_user --> SQLite3Template.InsertQuery
    check_username_availability --> SQLite3Template.RecordQuery.get_expiry_date --
    >SQLite3Template.RecordQuery.get_today_article --> SQLite3Template.RecordQuery
    get_today_article --> load_freq_history
    get_today_article --> difficulty.get_difficulty_level
    get_today_article --> user_difficulty_level
    get_today_article --> random.shuffle
    get_today_article --> random.choice
    get_today_article --> text_difficulty_level
    get_today_article --> within_range
    get_today_article --> get_answer_part
    mark_word --> load_freq_history
    mark_word --> pickle_idea.dict2lst
    mark_word --> pickle_idea.merge_frequency
    mark_word --> pickle_idea.save_frequency_to_pickle
    mainpage --> WordFreq
    mainpage --> load_freq_history
    mainpage --> pickle_idea.dict2lst
    mainpage --> pickle_idea.merge_frequency
    mainpage --> pickle_idea.save_frequency_to_pickle
    mainpage --> pickle_idea.dict2lst
    user_mark_word --> load_freq_history
    user_mark_word --> pickle_idea2.dict2lst
    user_mark_word --> pickle_idea2.merge_frequency
    user_mark_word --> pickle_idea2.save_frequency_to_pickle
    userpage --> WordFreq
    userpage --> pickle_idea.load_record
    userpage --> load_freq_history
    userpage --> sort_in_descending_order
    signup --> check_username_availability
    signup --> render_template
    signup --> add_user
    signup --> verify_user
    login --> render_template
    login --> verify_user
    difficulty.load_record --> pickle.load
    difficulty.difficulty_level_from_frequency -->
    math.log
    difficulty.get_difficulty_level --> revert_dict
    difficulty.get_difficulty_level -->
    sort_in_ascending_order
    difficulty.text_difficulty_level --> sort_in_descending_order
    pickle_idea.merge_frequency --> pickle_idea.lst2dict
    pickle_idea2.merge_frequency -->
    pickle_idea2.lst2dict
    SQLite3Template.do --> SQLite3Template.connect
    SQLite3Template.do --> SQLite3Template.instructions
    SQLite3Template.do --> SQLite3Template.operate
    WordFreq.get_freq --> wordfreqCMD.sort_in_descending_order

```

3. Pros and cons of the current architecture of EnglishPal:

Shortcoming:

1. The addition of the web pages makes the system inefficient to deliver media elements.
2. All the processing tasks are done by the server before the delivery of the content to the client. The server inefficient to handle multiple user requests.
3. Any development change or maintenance costs a lot.

Advantages:

1. Efficient with full-stack, no communication costs between front-end and back-end.
2. Effective for simple and small projects, with simple CRUD and smaller codebase, it's more enough.
3. Higher security, protecting the API from attack.
4. Similar concept and syntax, it helps focusing on project features.
5. It reduces the mistakes in communications.

Discussions

Through this lab we tried to understand the current health level of the architecture of EnglishPal. During the lab, we learnt to use Snakefood, Graphviz Online, Mermaid as well as Read the Docs. Most importantly, we mastered a basic work flow of analysing the structure and the dependency of an existing project which will sure to contribute to the future work.

References

Graphviz. <https://graphviz.org/>