Implementation and Functionality of stories-web-app

Richard-Johnson Farukh

March 13, 2019

1 Approach

I chose to work with Python 3.7, utilizing the Flask framework for web integration, due the ease of use of the technologies. They provided the necessary functionality for the application, without requiring additional packages on top. To keep things simple, the forms for writing text do not use FlaskWTF, even though for a commercial application that would be the preferred implementation.

For my approach, I chose to use a Tree structure, as the specifications did not mention random access of the individual sentences. Each individual node stores the sentence it contains, a reference to the root (which is used when trying to go back to the start), as well as a dictionary of references to the next possible sentences.

On the client side, to the sides of the current sentence I have forms, which upon being submitted, will change to a link to the next sentence. These forms send their position relative to the center along with their string to the server using "post". I also have a link to the start, which just sets the current node to be the root node.

On the server side, I perform the URL routing part of the application and I add the nodes to the tree. The server adds the nodes from the post requests and changes the current sentence based the clicked text, and it uses its internally stored Tree node called "current_state" to do so.

```
class Story_Node(object):
def __init__(self, sentence, root=None):
    self.sentence = sentence
    self.root = root
    self.nodes = {
        "top" : None,
        "bottom" : None,
        "left" : None,
        "right" : None
    }

def __str__(self):
        return self.sentence
```

Listing 1: Story_Node Object

I developed the app on macOS and I installed the flask dependency through PIP. In order to run it, from the directory containing the "stories-web-app.py" file you need to run:

```
> python3 stories-web-app.py
```

and go to http://127.0.0.1:5000/. The interface and functionally are the same as the ones from the specification.

2 Shortcuts

One of the shortcuts I took is not keeping a reference to the individual nodes of the Tree, and I did not use Sass with a separate file for referencing the form tags within ”index.html”, which would alleviate a bit of the reduncancy of the file.

I believe there is a better way of changing the content within the grid cells using some version:

```
for key in current_node.keys():
```

using Jinja2 syntax within the ”index.html”, but I could not find a better way of creating the ”grid-item” elements.

Finally, the application stores only a single state, and can change it only based on the current one or go to the start. I would have implemented a post request from the sentence links together with a List of Dictionaries, instead of a Tree, in order to bypass the problem, but it didn't seem necessary.

3 The test

I believe that the test is used to examine the candidate's knowledge of frameworks, or alternatively their ability to learn new frameworks to a usable degree given a short amount of time. Knowledge of data structures is a plus in this situation, given that using a Dictionary or a List would be just as viable for this app in particular.

It is also used to evaluate their approach, and their reverse engineering skills, given a completed product, which differs from the conventional Codility or HackerRank challenge.