Sequoia

Frank Li

What is Sequoia?

Sequoia is a static-site generator in part inspired by Hugo, Jekyll, and others.

Right now, it isn’t as effective, but it can generate a rough sketch of what a static website could look like, and all from a single YAML file!

Inspiration

At first, I wanted to create a mark-and-sweep garbage collector. However, this meant that I also wanted to create a basic scripting language, in which I ran into several pitfalls regarding function composition and good ways of passing functions in Java. Naturally, as I was planning on using YAML already, I shifted towards the static site generator. My personal website is made using Hugo, and I’ve always been interested in what kind of work goes into it

Packages

With Maven to manage the build:

- Zt-zip to expedite the zip-file unzipping process
- Apache commons for some helpful file utilities (I don’t think I ended up using any in the final version, besides the HTML unescaping method of StringUtils)
- Picocli as the Cli framework.
- Mustache.java as the templating engine
- SnakeYaml as the Yaml parser

Core Algorithm

Basically, most of the computation on my part is a depth-first traversal of the YAML document. Most of the difficulties arose in making everything work together, and in the end, there is still this tiny little bug regarding an old Reflection library that I don’t think is compatible with the current version.

It was very pleasant to use Mustache.js, and besides dealing with the nesting, it did a lot of the heavy-lifting regarding substitutions and etc.

SnakeYaml was horrifyingly difficult to work with, mostly dude to the nature of serialized documents not working well with Java’s strong typing. This resulted in a *lot* of safe but ugly typecasting.

Maven

Using Maven was AMAZING. Compared to the package managers I’ve used before (mainly npm and pip), Maven as just a package manager is so much more reliable and trustworthy. Together with the build and testing tools. It makes such a good development tool, and explains why Java is the most-used language in the business.

Demo

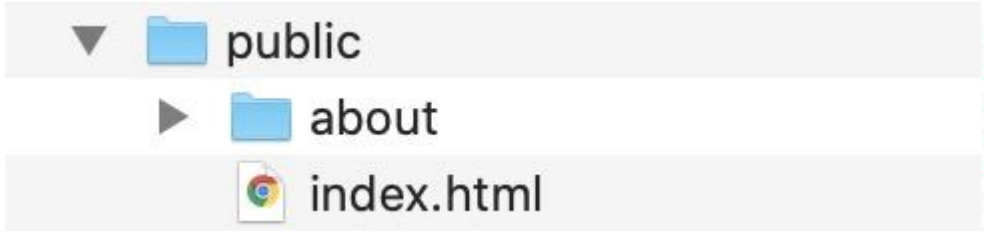
The sample base project's source.yml file:

```yaml
Home:
  type: base
  url: /
  content:
    pages:
      type: navbar
    line1: Hi guys! How are you? This is an example!
    line2: this is another line!

About:
  type: base
  url: /about
  content:
    pages:
      type: navbar
    line1: Not much about me!
```

Demo

The result:




```
<!DOCTYPE HTML>

<html lang="en">

<head>
    <title>
        Home
    </title>
</head>

<body>


<nav>
    <ul>
        <li>
            <a href="/">
        </li>
    </ul>
    <ul>
        <li>
            <a href="/about">About</a>
        </li>
    </ul>
</nav>
 <br>
    Hi guys! How are you? This is an example! <br>
    this is another line! <br>
```

```
<!DOCTYPE HTML>

<html lang="en">

<head>
    <title>
        About
    </title>
</head>

<body>


<nav>
    <ul>
        <li>
            <a href="/">Home</a>
        </li>
    </ul>
    <ul>
        <li>
            <a href="/about">About</a>
        </li>
    </ul>
</nav>
 <br>
    Not much about me! <br>
```

Experience

It was a really cool experience learning how to use Java in a full-size projects. One of the biggest difficulties of this project was learning how to retrieve static resources from a flat JAR file (hint: that's why I used zip files), and it's really nice to have found a way to do it. It was really weird typecasting so much, but AFAIK there really isn't a better way for serialized data of an unknown form (indefinite nesting). While I was working on the language still, I learned a lot about reflection and annotations but it didn't really transfer over to this iteration.

I also hard coded some parameters in towards the end, but I think there may be a better way using interfaces.

References

http://mustache.github.io/mustache.5.html

https://github.com/spullara/mustache.java

https://github.com/zeroturnaround/zt-zip

https://commons.apache.org/proper/commons-text/

https://picocli.info/

https://bitbucket.org/asomov/snakeyaml/wiki/Documentation

Future Directions

In the future, I think this project can move forward by introducing more factors. Maybe allowing users to define their own templates, as well as developing a comprehensive config and theming system. I have seen other static site generators focus a lot of concurrency and optimization, so that could be a future direction, potentially.

All in all, I think this was a cool way to use a lot of the techniques and ideas we learned in class this year on a creative project. Using objects and different data structures made this project a looooottt easier.