Silhouette

Java Web Framework

Håkon Marthinsen, Mats Engélien, Lars Erik Faber

GROUP 12 Second Delivery

Table of Contents

Link to our GitHub: https://github.com/norwen3/Silhouette

Intro

Silhouette is a Java web framework that aims to let users easily create websites in java and enables people who are not web designers to generate HTML and CSS from a range of classes and methods. The following report will give insight to our design decisions and the current state of our framework.

API Design Specification

Scenarios

Scenario 1

Make CSS two rulesets and a grid. Give one of the rulesets a class to target and change the background-color of one ruleset. Add the grid to the same ruleset and set it's columns and rows. On the second ruleset, set the text-color, add it to the first ruleset, create a Container of type "header" and give it the same class as the ruleset.

Scenario 2

Make a table whose size changes dynamically, add values to the header row and add values to the rest of the rows as they are generated. Apply a class to the table and set a header color for the table.

Client-Code

Scenario 1

```
RuleSet colorBlue = new RuleSet();
colorBlue.setClass("BlueClass");
colorBlue.addRule("background-color:blue");

Grid blueGrid = new Grid();
colorBlue.addRule(blueGrid.setColumns("10px", "12px"));
colorBlue.addRule(blueGrid.setRows("10px", "12px"));

RuleSet colorGreen = new RuleSet();
colorGreen.addRule("font-color", Color.colorRGB(23, 52, 12));


colorBlue.addRule(colorGreen);

Container myHeader = new Container("header");
myHeader.applyClass("BlueClass");

-->
<header class="BlueClass" ></header>
```

Scenario 2

```
Table table = new Table();
int col = headerArray.length;
int row = array.length;
RuleSet color = new RuleSet();
color.addRule("background-color", Color.colorRGB(255,255,255));
```

```
ColGroup headerCol = new ColGroup();

table.setSize(row, col);
table.applyClass("tableClass");

for(int i= 0; i<row; i++){
    for(int j= 0; j<col; j++){
        while(i<1){
            table.insertHead(i,j, headerArray[j]);
        }
        table.insertValue(i,j, array[j])
        if(i<1){
            headerCol.addCol(col, color.toString());
        }
    }
}
```

Project structure for Silhouette

Below is the resulting api from the design specification.

```
Silhouette
|
|---CSS.LowLevel
|   |
|   |---ILowLevel
|   |   |---IAlignment.java
|   |
|   |---Style
|   |   |---Animation.java
|   |   |---FlexBox.java
|   |   |---Grid.java
|   |   |---GridChild.java
|   |   |---RuleSet.java
|   |
|   |---Values
|   |   |---Color.java
|   |
|   |---StyleSheet.java
|
|---HTML
    |
    |---HighLevel
    |   |
    |   |---Component
    |   |   |---Page.java
    |   |
    |   |---Feature
    |       |---Article.java
    |       |---Footer.java
    |       |---Graphic.java
    |       |---Header.java
    |       |---Schema.java
    |       |---Table.java
    |
    |---LowLevel
        |
        |---Component
        |   |---HTML.java
        |
        |---Element
        |   |---Anchor.java
        |   |---Audio.java
        |   |---ColGroup.java
        |   |---Container.java
        |   |---ContainerElement.java
        |   |---Element.java
        |   |---EmptyElement.java
        |   |---FieldSet.java
        |   |---Figure.java
        |   |---Form.java
        |   |---Format.java
        |   |---Heading.java
        |   |---Image.java
        |   |---ListElement.java
        |   |---Paragraph.java
        |   |---Picture.java
        |   |---Table.java
        |   |---Video.java
        |
        |---ILowLevel
            |---IDimensions.java
            |---ISource.java
```

Principles of the Design

This section covers the design principles behind our framework, the naming of the classes, the abstractions between classes and the reasoning behind its structure. Let's start with the main APIs.

The low-level APIs in our framework has a direct corelation to HTML and CSS, with some abstractions and exclusions. For example, while Form object translates to the Form tag in HTML, some classes generalised tags such as Aside, Article and Header into a "Container" object. That way, the user only needs to specify the semantic tag as an argument.

We've only allowed high-level APIs to work with low-level APIs to avoid complication in the code generation process. For example, the HTML low-level API will never be able to directly influence the CSS low-level API, this is also further enhanced in our package structure. For the high-level APIs we plan to simplify the process by cutting out tags and styles that can be automatically generated, such as meta tags. We want the low-level and high-level APIs to be completely independent, but able to work with each other, so that the user is not constrained to either one of them. This let's anyone make a simple website with the high-level API and gives them the opportunity to change nitty-gritty details with the low-level API.

The class and method names follow the standard Java naming conventions. For example, the classes that are a one to one corelation to HTML and CSS are simply named after their HTML/CSS counterpart. Other classes that act as a generalization of other HTML tags are assigned a describing name, for example Container class (Temporary name).

User Testing

We were given the opportunity to cooperate with Group 5 when doing user testing, where they tested our framework, and later we tested theirs. The following section covers the results from our session.

Description of the setup

We sent a jar file containing the framework to Group 5 and started with some simple instructions. Firstly, they needed to start a new project and add our framework/library to their project from the project structure menu. Once this was done, they were ready to start coding inside the main method.

We provided them with two sample files, one called index.html and one called style.css, their task was to simply reproduce the structure and the code inside these files using our Java framework.

index.html

```html
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="style.css" type="text/css"/>
</head>
<body>
    <header>
        <h1 class="myH1">Title</h1>
        <nav>
            <a href="www.google.com">Google</a>
        </nav>
    </header>
    <main>
        <article>
            <p id="myParagraph">Lorem Ipsum</p>
        </article>
    </main>
    <footer>

    </footer>
</body>
</html>
```

style.css

```css
body {
    background-color: rgb(200, 245, 240);
    display:grid;
}

nav {
    display: flex;
}

.myH1 {
    font-size: 3em;
}

#myParagraph {
    background-color: #179421;
}
```

At first, we let them figure out the process themselves, but once they met issues and got stuck, we gave explanations to what each code block represented and guided them through the process. The next section will cover the resulting code and feedback from our testing.

The Code

The following is the client code created by Group 5 using our framework to make a simple HTML and CSS file:

```
public class Main {

    public static void main(String[] args) {
    // write your code here
        HTML html = new HTML();
        html.setLinkType("stylesheet","style.css" ,"css" );

        Container head = new Container("head");
        Container body = new Container("body");

        html.append(head);

        Container header = new Container("header");
        Container main = new Container("main");
        Container footer = new Container("footer");

        Heading heading = new Heading("1", "asdasdsada");
        heading.applyClass("h1 class name");

        header.addElement(heading);

        body.addElement(header);
        body.addElement(main);
        body.addElement(footer);

        Container nav = new Container("nav");
        header.addElement(nav);

        Anchor anchor = new Anchor("google","google.com");
        nav.addElement(anchor);

        Container article = new Container("article");
        Paragraph p = new Paragraph("YESYEYES");

        article.addElement(p);
        p.applyId("paragraph");

        main.addElement(article);

        html.append(body);
```

```
        html.initialize();

        StyleSheet css = new StyleSheet();
        RuleSet rs = new RuleSet();

        rs.setSelector("body");
        rs.addRule("background-color", "red");
        rs.setDisplay("grid");

        RuleSet rsNav = new RuleSet();
        rsNav.setSelector("nav");
        rsNav.setDisplay("flex");

        RuleSet rsfont = new RuleSet();
        rsfont.setClass("header");
        rsfont.addRule("font-size", "12px");

        RuleSet pC = new RuleSet();
        pC.setId("paragraph");
        pC.addRule("background-color", "blue");
    }
}
```

(Note that this code section excludes all errors)

Feedback

Group 5:

Right after they had tested our framework and completed the setup, we asked them to give their initial reaction and opinion. They said, “It was pretty good once we got the hang of the class names” which was a relief to hear. It seemed to us that they found the APIs to be intuitive, as they quickly grew familiar with the main classes. We then followed up with a few questions regarding their experience, and they told us they could easily expand further on their website with enough practice.

Group 5 later sent us a short report on our program, starting with the setup: “The setup was pretty good, since we were ordered to reproduce a given webpage along with its styling, which is pretty straightforward. The setup structure was good”. And to the actual framework they said: “The code was semantic and well structured. It is also easily readable and applicable to making websites with java.”. They thought our framework took a good direction for forming webpages, as they found Java to be challenging for that use.

“The only nit-pick is that some parts of the framework produced errors, or they wouldn’t work like they did in the example code.”, meaning that some classes didn’t extend correctly, and didn’t have access to certain methods. “For example, when we tried to add a paragraph to an article element, java didn’t recognize it. Other than that, the code structure was clean”.

The Revised API

After our session with Group 5, we ended up with a log of issues that needed fixing. Below are the most important changes to our API.

- The setLinkType method is now accessible within the Container class.

- The setLinkType method now has an overload with two parameters.
- The Heading class now takes an integer as the first parameter.
- The Heading class now extends correctly, as it didn't have access to the applyClass method.
- The addElements method now has an overload that uses varargs in parameters.
- The Paragraph class now extends correctly, as it didn't have access to the addElement, applyId and applyClass methods.

Examples from the API

Example 1 – General use of Silhouette Framework

```
HTML html = new HTML();

//This will originally go into a head tag
html.setTitle("My first webpage!!!");
html.setMetaCharset("UTF-8");
html.setLinkType("stylesheet", "myCss.css", "text/css");
Container head = new Container("head");
html.append(head);
//head takes nothing at the moment

Container body = new Container("body");
html.append(body);

//Start header
Container header = new Container("header");
body.addElement(header);

Heading h1 = new Heading("h1", "Welcom!!!");
h1.applayId("h1");
header.addElement(h1);

Container nav = new Container("nav");
nav.applyId("nav");
nav.addElement(
        new Anchor("google", "google.com").applyClass("a1"),
        new Anchor("youtube", "youtube.com").applyClass("a2"),
        new Anchor("123Spill", "123spill.no").applyClass("a3"));
//End of header
//Start main
Container main = new Container("main");
body.addElement(main);

Heading h2 = new Heading("h2", "About me :D");
main.addElement(h2);

Image me = new Image("picture/Of/Me.png");
main.addElement(me);

Paragraph p = new Paragraph("Lol, I fold you bro.");
```

```
main.addElement(p);
//end of main

Container footer = new Container("footer");
body.addElement(footer);

StyleSheet css = new StyleSheet();
css.setFileName("myCss.css");

RuleSet h1css = new RuleSet();
h1css.setId("h1");
h1css.addRule("background-color:", "blue");
css.applayStyle(h1css);


RuleSet navcss = new RuleSet();
navcss.setId("nav");
navcss.addRule("display:", "flex");
css.applayStyle(navcss);
```