Part II: Functional Models and Finite State Machines

Introduction to Finite State Machines

The finite state machine (or FSM) can be constructed before the source code or independently of the source code. A finite state machine (or FSM) can be used as a specification for allowed behavior.

A finite state machine is a set of states and a set of transitions.

A finite state machine is a directed graph.

A finite state machine is a node that represents the state of a program.

Edge represents the operation of transforming one program state into another program state. Usually marked with program operations, conditions or events.

Due to countless states, FSM must be abstract.

The reason why finite models are useful for testing

Using finite models, we can draw a state transition tables. These transition tables can help us check the completeness of the program. These completeness can help us do the followings conditions.

1. Help analyze the original state of program.
2. Help analyze the complete process of program.
3. Help test potential bugs.
4. Testing might pass all the branches
5. After analyzing the branches, we can test in more detailed and more targeted way.
6. When encountered with bug, we can target at which branch has the bugs.

Choose a feature

In our project `JSoup`, it is a Java library for processing actual HTML/XML. In the functional model, we extracted two features from `JSoup`.

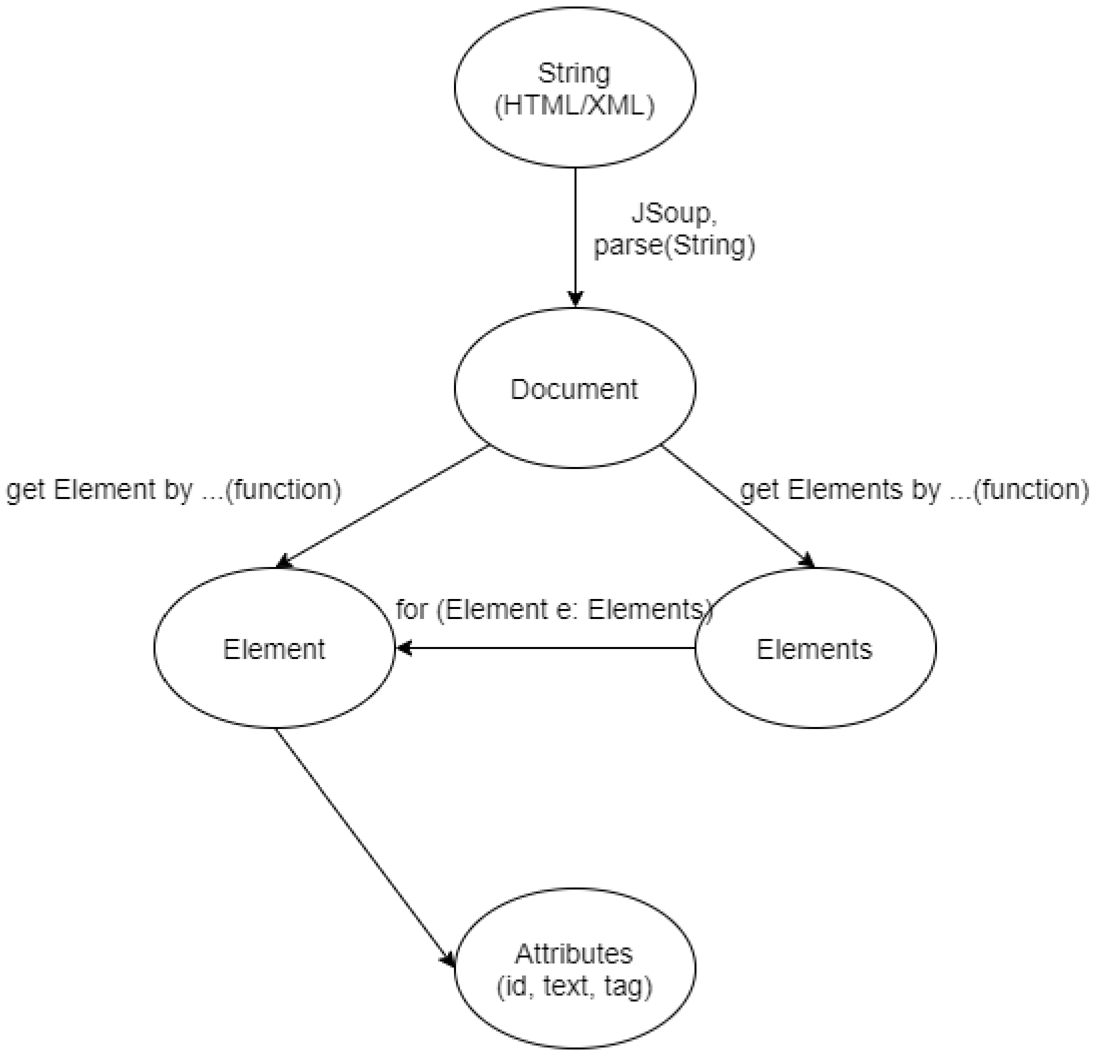
- scrape and parse HTML/XML

- manipulate the HTML/XML elements, attributes.

Create, draw, and describe that functional model, how it works

In `JSoup` progress, you can see the process through the picture below.

1. Firstly, `parse` HTML or XML Strings to `Document`.
2. Then get `Element` or `Elements` by `JSoup` functions.
3. Using `for (Element e : Elements)`, `Elements` can transfer to `Element`.
4. Using `Element`, we can see the `Attributes`.



How to use functional models in `JSoup`

Write test cases

The test cases are stored in the directory - `/src/test/java/org.jsoup/swe261/FiniteStateMachinesTest.java`

The files within are written here.

To see the specific `input_html`, please see our github document.

```
public class FiniteStateMachinesTest {

    @Test
    public void String2Document() {
        String html = input_html;
        String expStr = "<body>\n" +
                "  <p>First post! <img src=\"foo.png\"></p>\n" +
                "  <p>Second post! <img src=\"foo2.png\"></p>\n" +
                "</body>";
        System.out.println(doc.body());
        assertEquals(expStr, doc.body().toString());
    }

    @Test
    public void Document2Element() {
        String html = input_html;
        Document doc = Jsoup.parse(html);
        Element ele = doc.body();
        String expStr = "<p>First post! <img src=\"foo.png\"></p>";
        //System.out.println(ele.children());
        assertEquals(expStr,ele.child(0).toString());
        expStr = "<p>Second post! <img src=\"foo2.png\"></p>";
        assertEquals(expStr,ele.child(1).toString());
    }

    @Test
    public void Element2Elements() {
        String html = input_html;
        Document doc = Jsoup.parse(html);
        Element ele = doc.body();
        Elements eles = ele.children();
        int exp = 2;
        assertEquals(exp, eles.size());
    }

    @Test
    public void Document2Elements() {
        String html = input_html;
        Document doc = Jsoup.parse(html);
        Elements eles = doc.getElementsByTag("p");
        int exp = 2;
        assertEquals(exp, eles.size());
    }

    @Test
    public void Elements2Element() {
        String html = input_html;
        Document doc = Jsoup.parse(html);
        Elements eles = doc.getElementsByTag("p");
        String expStr = "<p>First post! <img src=\"foo.png\"></p>";
        assertEquals(expStr,eles.get(0).toString());
        expStr = "<p>Second post! <img src=\"foo2.png\"></p>";
        assertEquals(expStr,eles.get(1).toString());
    }

    @Test
    public void Element2Attr() {
        String html = input_html;
        Document doc = Jsoup.parse(html);

        // need a better way to verify these:
        Element p = doc.body().child(0);
        assertEquals("p", p.tagName());
        assertEquals("foo > bar", p.attr("class"));
    }


}
```

To explain these code,

1. `String2Document` is the first process. This one `parse` HTML or XML Strings to `Document`

```
    @Test
    public void String2Document() {
        String html = input_html;
        Document doc = Jsoup.parse(html);
        String expStr = "<body>\n" +
                " <p>First post! <img src=\"foo.png\"></p>\n" +
                " <p>Second post! <img src=\"foo2.png\"></p>\n" +
                "</body>";
        System.out.println(doc.body());
        assertEquals(expStr, doc.body().toString());
    }
```

Reversely, `Document` can transfer to HTML/XML

```
@Test
    public void Document2Element() {
        String html = input_html;
        Document doc = Jsoup.parse(html);
        Element ele = doc.body();
        String expStr = "<p>First post! <img src=\"foo.png\"></p>";
        //System.out.println(ele.children());
        assertEquals(expStr,ele.child(0).toString());
        expStr = "<p>Second post! <img src=\"foo2.png\"></p>";
        assertEquals(expStr,ele.child(1).toString());
    }
```

2. Using `for (Element e : Elements)`, `Elements` can transfer to `Element`.

```
@Test
    public void Element2Elements() {
        String html = input_html;
        Document doc = Jsoup.parse(html);
        Element ele = doc.body();
        Elements eles = ele.children();
        int exp = 2;
        assertEquals(exp, eles.size());
    }
```

Reversely, `Elements` can transfer to `Element`.

```java
@Test
    public void Elements2Element() {
        String html = input_html;
        Document doc = Jsoup.parse(html);
        Elements eles = doc.getElementsByTag("p");
        String expStr = "<p>First post! <img src=\"foo.png\"></p>";
        assertEquals(expStr,eles.get(0).toString());
        expStr = "<p>Second post! <img src=\"foo2.png\"></p>";
        assertEquals(expStr,eles.get(1).toString());
    }
```

3. Using `Element`, we can see the `Attributes`.

```java
@Test
    public void Element2Attr() {
        String html = input_html;
        Document doc = Jsoup.parse(html);

        // need a better way to verify these:
        Element p = doc.body().child(0);
        assertEquals("p", p.tagName());
        assertEquals("foo > bar", p.attr("class"));
    }
```