Design Pattern

Fall, 2019

CAU

**<Team Project – Team 8>**

**Function Extension and Design Improvement**

**in JSoup parser**

20121067 한재현

20166627 김민하

20165167 이서현

20172774 송정우

**<목차>**

**1. 각 멤버의 역할 분담**

**2. Jsoup 개요**

**3. Jsoup 설계 및 구현 조사**

**4. 기능 확장 및 설계 개선**

**5. 테스트 수행 내역**

**6. GitHub 프로젝트 활동 요약**

1. **각 멤버의 역할 분담**

**공통 : Unit test, Jsoup 설계 분석,**

* 1. 한재현 – Github Management, Function Extension, Presentation
  2. 송정우 – Function Extension, System test, Presentation
  3. 이서현 – Design Improvement, Project Report
  4. 김민하 – Design Improvement, Project Report

1. **Jsoup 개요**
   1. **What is Jsoup?**

Jsoup은 자바 라이브러리로, 실세계의 HTML을 분석하는 파서이다.

HTML이 hierarchy가 있다는 점에 착안하여 이에 따라 비교적 쉽게 파싱할 수 있도록 한다. DOM, CSS, jquery method를 최대한 활용하여 데이터를 추출하고 다루기에 편리한 API를 제공한다.

* 1. **What Jsoup does?**
     + WHATWG HTML5 사양을 구현하고 최신 브라우저들과 같은 DOM으로 HTML을 파싱한다.
     + URL, 파일 또는 문자열을 소스로 하여 HTML을 스크랩 및 파싱한다.
     + DOM 구조를 추적하거나 CSS selector을 이용하여 데이터를 찾고 추출한다.
     + HTML 문서 내의 element, attribute 및 텍스트를 조작한다.
     + 사용자의 대상 콘텐츠를 지정된 태그만 남기고 나머지는 제거 가능한 화이트 리스트 방식을 통해 안전하게 관리한다. (head 내부에는 태그만 둔다)
     + MIT라이선스에 따라 배포되는 오픈소스 프로젝트로 GitHub에서 구할 수 있다.
  2. **Advantages**
* 긴 HTML Document를 파싱하여 추출하면 재사용이 편리하다.
* 크롤링을 할 때 어떤 구조로 되어있는지 알아보기 용이하며 파싱하기 쉽다.
  1. **Weak point**
* HTML이 변경될 시 파싱 결과에 반영되지 않는다.

1. **Jsoup 설계 및 구현 조사**
   1. **설계 Overview**

**<기능>**

**문서의 파싱, 문서 내부 크롤링, 데이터 수정, CSS selecting**

**<Package와 역할>**

|  |  |
| --- | --- |
| **Package 이름** | **역할** |
| [**org.jsoup**](https://jsoup.org/apidocs/org/jsoup/package-summary.html) | 대상 문서에 대한 정적 엑세스를 제공하고  결과를 얻어오는 Jsoup 기본 클래스를 포함  : URL이나 로컬 HTML에 연결하는 방법과 Response, Request 등을 가지고 있는 Connection 관련 클래스 포함. |
| **org.jsoup.examples** | 예제 프로그램들과 Jsoup 사용 예 |
| [**org.jsoup.helper**](https://jsoup.org/apidocs/org/jsoup/helper/package-summary.html) | Jsoup 파싱 작업을 도와주는 클래스  : 내부 클래스는 private으로 선언하며, helper class간의 의존성 생기지 않는 것이 좋음 |
| [**org.jsoup.internal**](https://jsoup.org/apidocs/org/jsoup/internal/package-summary.html) | Jsoup이 사용하는 Util 메소드 : 정적 메소드 만으로 이루어짐 |
| [**org.jsoup.nodes**](https://jsoup.org/apidocs/org/jsoup/nodes/package-summary.html) | connectin을 통해 얻어온 document의 구조를 위한 노드 |
| [**org.jsoup.parser**](https://jsoup.org/apidocs/org/jsoup/parser/package-summary.html) | HTML 파서, 태그 특정화 및 HTML 토크나이저 |
| [**org.jsoup.safety**](https://jsoup.org/apidocs/org/jsoup/safety/package-summary.html) | jsoup HTML 클리너 및 화이트리스트 정의 |
| [**org.jsoup.select**](https://jsoup.org/apidocs/org/jsoup/select/package-summary.html) | CSS 스타일 element selector를 지원하는 패키지 |

**<주요 요소>**

**1. Document : Jsoup get 방식으로 얻어온 HTML 문서 전체**

**2. Element : Document 내의 HTML element**

**3. Elements : Element들로 이루어진 자료형**

**4. Connection : Document를 얻기 위한 연결 관련 정보를 담은 객체**

**5. Response : Jsoup이 URL에 접속해 얻어온 결과,**

**status 코드, status 메시지, charset과 같은 헤더 메시지와 쿠키 등을 가짐**

**<동작 흐름>**

**<Document의 객체 모델>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fig.1 | | Document는 Element를 상속받고  Element는 Node를 상속받는다. | | |
| Fig.2 | | | 하나의 Element는 자식 Nodelist로 구성되어있고 하나의 부모 Element를 가진다. | |
|  | | |  | |

* 1. **이미 적용되어 있는 설계패턴 분석**

|  |  |
| --- | --- |
| 1. strategy 패턴 | *Parser.java* |
|  |  |
| 클래스 다이어그램 | 판단근거 |
| A black and silver text on a screen  Description automatically generated  A close up of a clock  Description automatically generatedA screenshot of a cell phone  Description automatically generated | |
| 소스코드와 매핑 | |

|  |  |
| --- | --- |
| 2. facade 패턴 | *Parser.Java* |
|  |  |
| 클래스 다이어그램 | 판단근거 |
|  | |
| 소스코드와 매핑 | |

|  |  |
| --- | --- |
| 3. visitor 패턴 | *Safety.Cleaner.Java* |
|  | 일반적으로 객체지향 프로그래밍 환경에서는 객체내부에 멤버변수와 메소드를 함께 작성한다.  그러나 visitor 패턴에서는 변수와 메소드를 분리하여 작성한다.  아래 코드를 보면, NodeVisitor 인터페이스를 상속받은 클래스 내에서 visitor 패턴을 호출하고 있다. |
| 클래스 다이어그램 | 판단근거 |
|  | |
| 소스코드와 매핑 | |

|  |  |
| --- | --- |
| 4. composite 패턴 | *CombiningEvaluator.java* |
| A circuit board  Description automatically generated | |
| 클래스 다이어그램 | |
| * **Component** : Evaluator * **Composite** : CombiningEvaluator * **Leaf** : CssNthEvaluator, IndexEvaluator | |
| 판단근거 | |
| A screenshot of a cell phone  Description automatically generated  A screenshot of a cell phone  Description automatically generated  A screenshot of a cell phone  Description automatically generatedA screenshot of a cell phone  Description automatically generated | |
| 소스코드와 매핑 | |

|  |  |
| --- | --- |
| 5. Decorator 패턴 | *nodes.Element.java* |
|  | |
| 클래스 다이어그램 | |
| Node package는 HTML DOM Tree에 기초하여 작성된 클래스들로 각각의 클래스들이 특정 요소들을 가리키고 있다. Form Element는 Form태그와 관련된 부분을, DataNode는 style과 script 태그의 내용을 담고 있는 것과 같이 각각의 태그의 대한 Node라고 할 수 있겠다.    Node pakage에서 Composite pattern의 흔적을 찾았는데 이유는 다음과 같다.  **Component** : Node, **Leaf** : LeafNode, **Composite** : Element  Element는 부모 클래스로 Node를 가지고 있으며, 여러 개의 Node들을 List 형식으로 가지고 있다. 또한 Element 클래스는 부모 클래스의 operation(@override) 뿐 아니라 새로운 기능들을 가지고 있다.  아래 코드를 통해 Node를 추가하거나 빼는 메소드나 LeafNode의 자식들의 리스트를 반환하는 메소드를 확인할 수 있다. 또한 LeafNode 클래스는 부모 클래스로Node를 가지고 있다. | |
| 판단근거 | |
|  | |
| 소스코드와 매핑 | |

**4. 기능 확장 및 설계 개선**

**A. 확장된 기능**

- HTML 로 작성된 문서를 시각화 하여 브라우저를 띄워 결과를 바로 보여주는 기능을 구현하였다.

이를 통해 사용자는 보다 쉽게 코드를 수정 및 분석이 가능하다.

다음과 같은 코드로 구현하였다.

|  |
| --- |
|  |
| Fig.3 기능확장 코드 |

|  |
| --- |
|  |
| Fig4. 기능확장 코드 |
| A screen shot of a computer  Description automatically generated |
| Fig5. 기능확장 코드에 대한 테스트 결과 |

**B. 설계 개선 내용**

- 기존에 존재하던 Element 메소드는 tag, baseUri, Attributes 세개의 매개 변수를 가진다. 생성자를 통해 데이터를 받기 때문에 String 또는 baseUri에 인자가 할당될 필요가 없을 경우, 해당 매개 변수를 제외한 새로운 Element 메소드를 만들고, this.을 이용하여 기존 Element메소드를 다시 불러오는 형태이다.

이러한 형태는 새로운 인자를 받아야 할 경우 생성자를 수정해 줘야할 뿐 아니라 코드 좌우 길이가 길어져 가독성을 저해한다. 또한 어떤 위치에 어떤 타입의 데이터가 들어가는지 사용자가 정확히 기억해야하기에 코드 작성 시 실수할 확률이 높아진다. 미래에 웹 관련 새로운 언어의 등장 또는 다양화가 이루어질 경우 Jsoup의 기능확장에 있어 효율 저하를 초래할 것이라고 판단하여 Builder Pattern을 적용시켜보았다.

|  |
| --- |
|  |
| Fig.4 기존 Element 메소드 |

Builder 패턴은 인스턴스 생성자를 통해 직접 생성하는 대신, 내부 클래스를 이용하여 간접적으로 객체를 생성한다. 생성과 표기를 분리하여 복잡한 객체 생성을 깔끔하고 유연하게 하며 한 번에 객체를 생성하므로 객체 일관성이 깨지지 않는다. Private을 이용하여 외부의 접근을 막아 클래스와 사용 대상의 결합을 느슨하게 한다. 또한 앞서 언급한 것과 같이 매개 변수가 다양해질 경우 기존 생성자 패턴으로 인수를 전달하는 것은 비효율적인데, 이에 반해 builder 패턴을 이용 시 인수 파악이 용이하다. 전반적으로 SRP가 잘 지켜지도록 하는 패턴이다.

|  |  |  |
| --- | --- | --- |
|  | 다음과 같이 Builder 클래스를 작성하고 필수 인자로 tag를 선언해준다.  attributes와 baseUri는 선택적 인자로 선언해준다.  tag는 String 타입과 Tag 타입이 경우에 따라 결정되어 사용되기 때문에, 두 가지 모두 적용 가능하도록 만들었다. | |
| Fig.5 builder pattern 적용 | |  |

위와 같이 builder 클래스를 작성하고, 전체 코드에서 Element 메소드가 적용된 곳을 아래와 같이 변경해주었다.

|  |
| --- |
|  |
| Fig.6 설계 개선 코드 |

**5. 테스트 수행 내역**

**A. 기능 확장 테스트**

|  |
| --- |
|  |
| **확장된 기능 테스트 코드** |

|  |
| --- |
|  |
| **확장된 기능 테스트 결과** |

**B. 설계 개선 테스트**

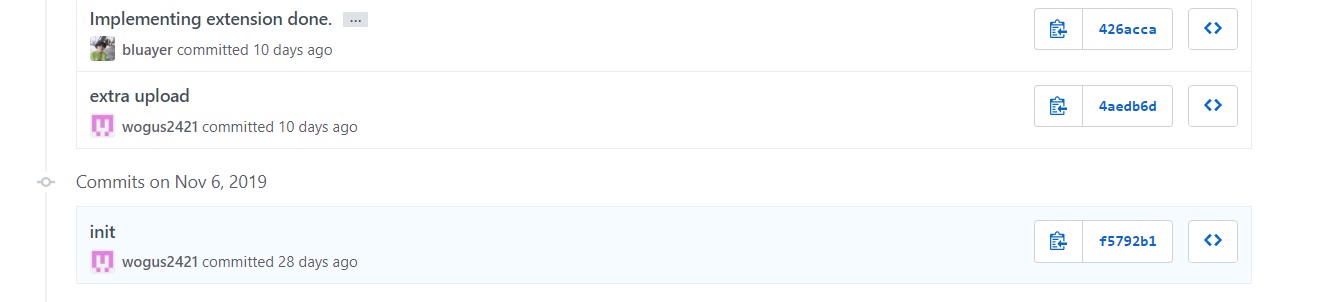
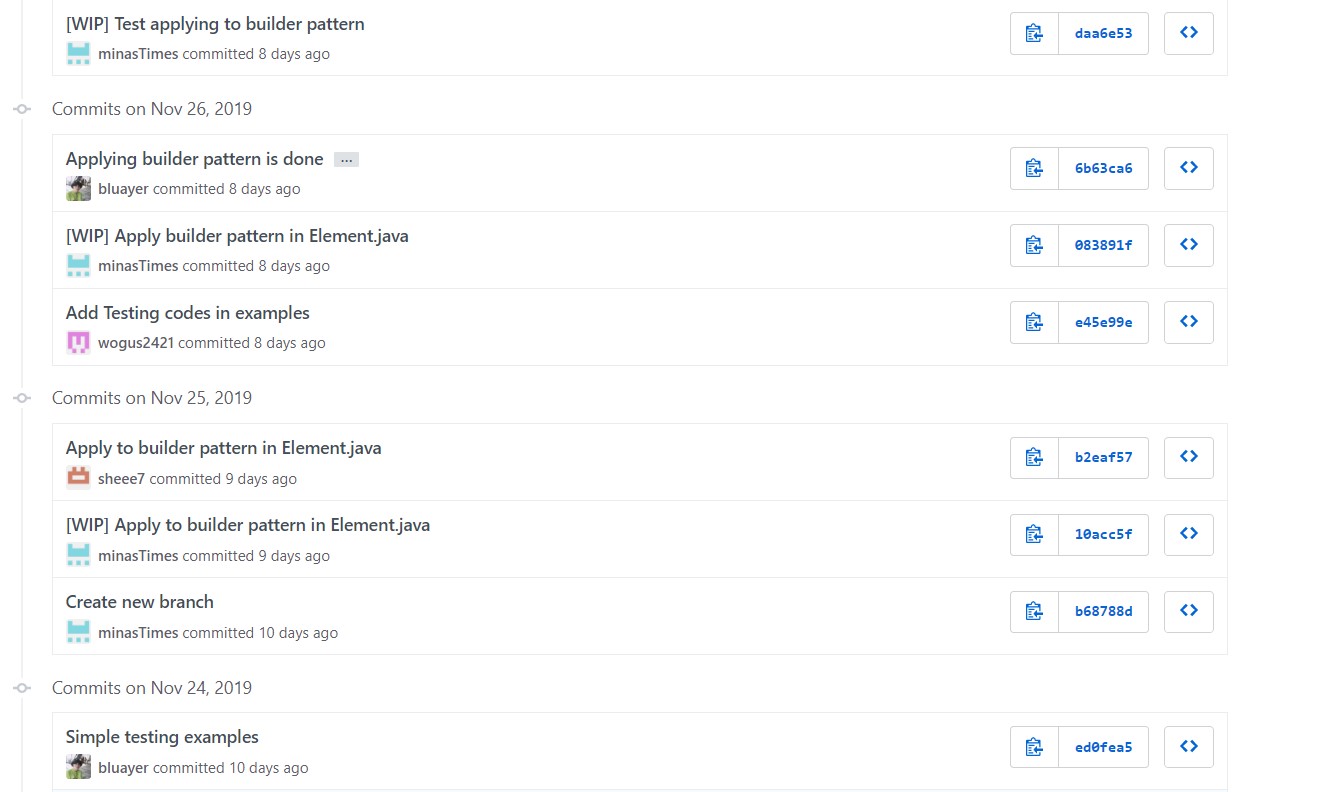
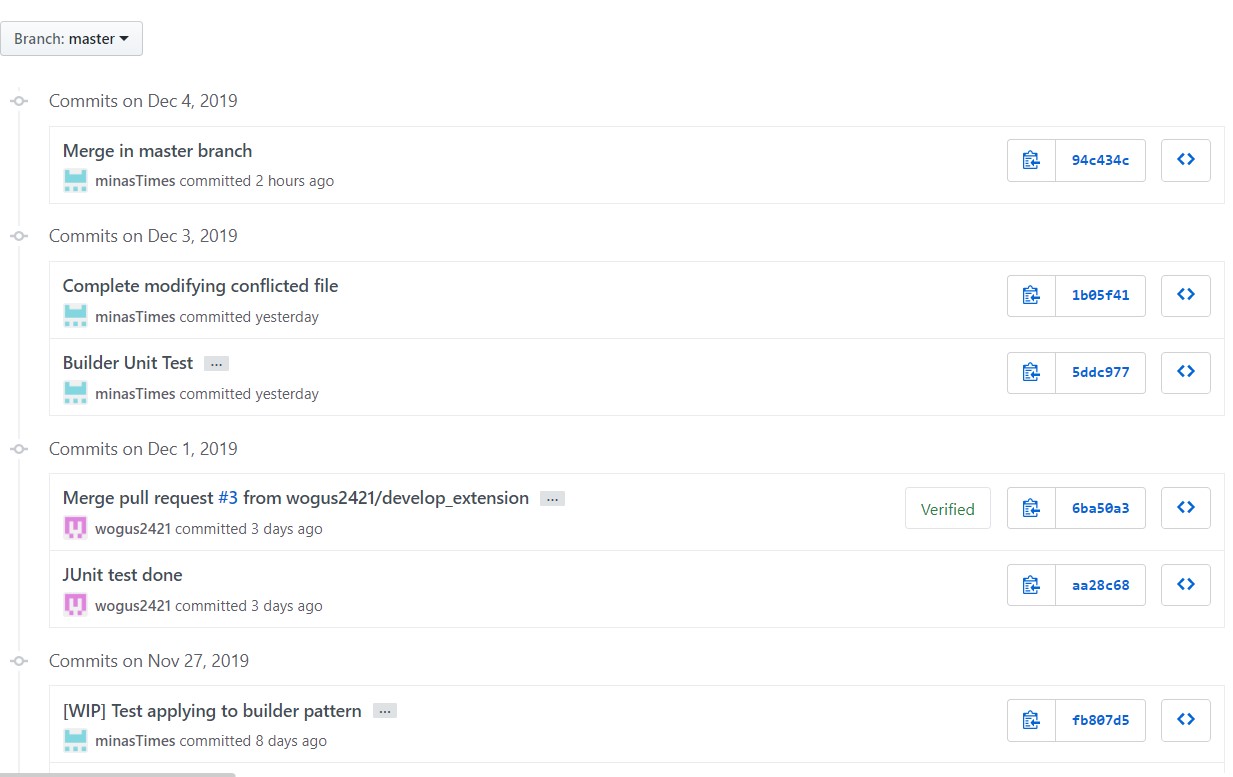
|  |
| --- |
|  |
| **설계 개선 테스트 코드** |

|  |
| --- |
|  |
| **설계 개선 테스트 결과** |

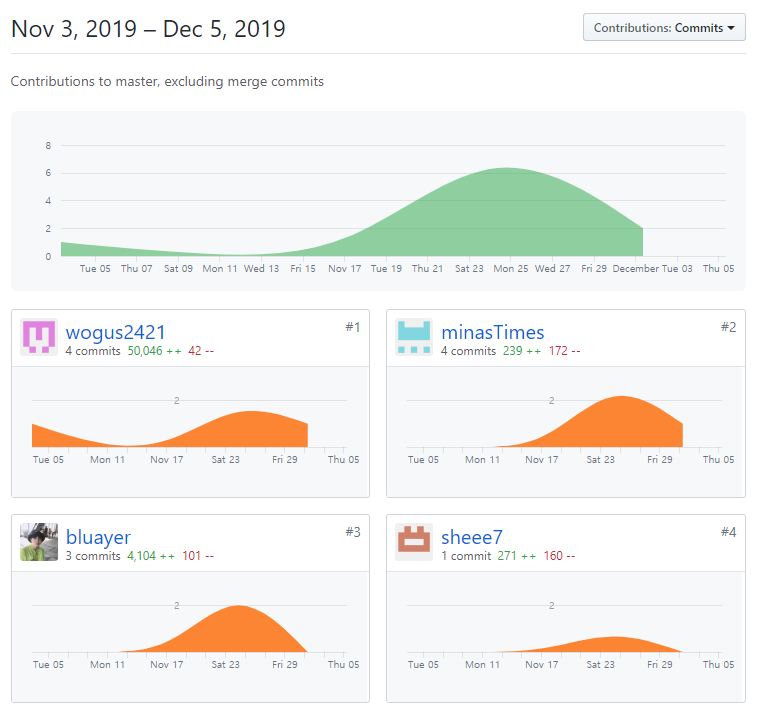
1. **GitHub 프로젝트 활동 요약**
   1. **Github 프로젝트 주소**

<https://github.com/wogus2421/WildHorse>

* 1. **프로젝트 progress history**

****

* 1. **Github insight Contributors**

****