**창의 소프트웨어**

**동계훈련 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | Seat Finder |
| 팀 명 | - |
| 문서 제목 | 프로젝트 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.3 |
| **Date** | 2015. 3. 2 |

|  |  |
| --- | --- |
| **팀원** | 김윤호 |
|  |
|  |
|  |
|  |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 비주얼 프로그래밍 수강 학생 중 프로젝트 “도전! 카드 뒤집기”를 수행하는 팀 “더블 플레이”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “더블 플레이”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 프로젝트 결과보고서.docx |
| **원안작성자** | 김윤호 |
| **수정작업자** | 김윤호 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2015. 2. 27 | 김윤호 | 1.0 | 최초 작성 |  |
|  |  |  |  |  |

**목 차**

[**1** **프로젝트 목표** 4](#_Toc413004727)

[**2** **프로그램 개요** 5](#_Toc413004728)

[**2.1** **구현 배경** 5](#_Toc413004729)

[**2.2** **구현에 사용된 데이터와 언어** 5](#_Toc413004730)

[**3** **본문** 6](#_Toc413004731)

[**3.1** **프로그램 구조도 및 간략한 설명** 6](#_Toc413004732)

[**3.2** **데이터 입력** 7](#_Toc413004733)

[**3.3** **혼잡도 계산 알고리즘** 7](#_Toc413004734)

[**3.4** **이벤트 처리 및 웹 페이지 구현** 8](#_Toc413004735)

[**3.5** **UI (Bootstrap)** 9](#_Toc413004736)

[**4** **개선/보완해야 할 점** 10](#_Toc413004737)

[**5** **자기평가 및 소감** 10](#_Toc413004738)

[**6** **참고문헌** 11](#_Toc413004739)

# **프로젝트 목표**

* **다양한 웹 프로그래밍 언어를 사용하여 사용자들에게 편의를 제공하는 웹 사이트를 구축한다.**
* **오픈 소스를 활용하여 만들고자 하는 기능을 더욱 편리하고 효과적으로 제작한다.**
* **오픈 데이터를 활용하여 실생활 밀접한 프로그램을 제작한다.**
* **웹 서버와 연동하여 실제 웹 브라우저에서 호스팅한다. (크로스 브라우저 문제 포함)**

# **프로그램 개요**

## **구현 배경**

지하철을 타면 어떤 칸에는 사람이 꽉 차있는 반면 어떤 칸에는 비교적 한산한 칸이 있다는 것을 볼 수 있다. 만약 이러한 차량 내 혼잡도를 알려주는 서비스를 개설한다면 지하철을 이용하는 수많은 승객들에게 편의를 제공할 수 있을 것이다.

## **구현에 사용된 데이터와 언어**

1. **데이터**

서울시 열린 데이터광장(http://data.seoul.go.kr)에서 제공하는 시간대별 역 승하차량, 환승객 통계 데이터를 토대로 프로그램을 제작하였다. 각 파일은 엑셀 파일과 JSON파일로 제공되며 이 중 특별한 편집 작업 없이 그대로 쓸 수 있는 시간대별 역 승하차량은 JSON파일을 사용했으며 그 이외의 필요한 데이터들은 텍스트 파일로 편집하여 사용하였다.

1. **언어**
2. **JSP**

텍스트 파일과 JSON 파일 읽기, 열차 내 혼잡도를 계산하는 알고리즘은 다양한 연산과 기능들이 구현 가능한 Java로 구현했으며 이를 웹 페이지와 연계하기 위하여 JSP와 Tomcat 서버를 사용했다.

1. **JavaScript**

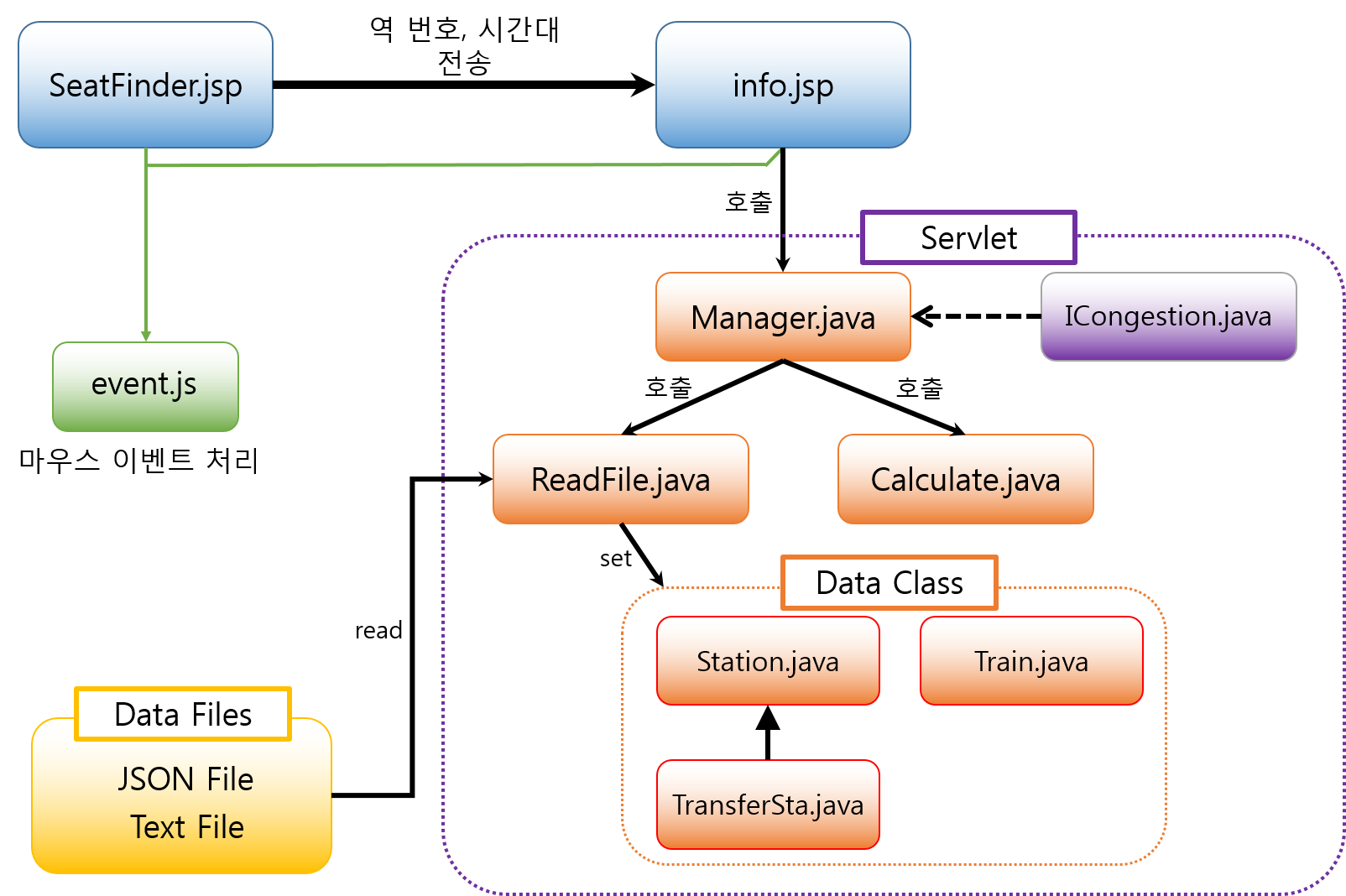
웹 페이지 상에서 일어나는 마우스 관련 이벤트 및 적합한 이미지 출력을 위하여 JavaScript를 사용하였다.

1. **HTML**

웹 페이지 구축 및 디자인 제작을 위하여 HTML과 CSS(Bootstrap)을 사용하였다.

# **본문**

## **프로그램 구조도 및 간략한 설명**



MVC2 모델을 기준으로 만들기 위해 View 담당인 JSP와 Controller 담당인 Servlet(Java)를 분리했고 서블릿들은 ICongestion 인터페이스를 구현한 Manager 클래스에 의해서만 JSP와 통신할 수 있도록 만들었다. 이번 프로젝트에서 DB구축은 못한 관계로, 서블릿에서 생성된 자료들이 웹 상에서 계속 호출되고 삭제되는 형태를 반복하는 방식으로 만들게 되었다.

서블릿 파일은 크게 정보를 담고 있는 데이터 클래스와 여러 가지 작업을 담은 실행 클래스로 구분했으며 데이터 클래스에서 환승역 관련 정보를 담고 있는 클래스는 상위 속성인 Station 클래스의 상속관계에 놓음으로써 관리가 편리하도록 만들었다.

진행 순서로, 먼저 SeatFinder.jsp를 불러와 프로그램을 시작한다. 이 페이지에서 검색할 역 번호와 시간대를 info.jsp에게 넘겨주면 JSON과 텍스트 파일에서 데이터를 읽어 ArrayList<Station>의 인스턴스를 생성한 다음 각 역에서의 시간대 열차 혼잡도를 계산한다. 이 결과를 선택했던 역 번호와 시간대에 매칭하여 사용자가 원하는 결과만 보이도록 출력한다.

또한 각각의 JSP파일에서 필요한 이벤트들을 처리하기 위하여 이벤트 핸들러 자바스크립트 파일을 만들었다.

## **데이터 입력**

서울시 오픈 데이터광장에서 받은 역 시간대별 승하차량과 환승객 수 자료로 만든 JSON, 텍스트 파일을 입출력 함수로 프로그램에 대입한다.

1. **텍스트 파일**

일반 자바 파일은 현재 폴더에서 컴파일이 진행되어 별다른 경로 설정 없이 파일 주소만 적어 호출할 수 있었지만 JSP에서는 코드가 실행되는 장소가 웹이기 때문에 tomcat 서버의 특정 폴더로 절대경로가 지정되기 때문에 이 경로 뒤에 파일명을 붙여야 파일 입출력이 가능하다. 기본 절대경로를 파악하기 위해 JSP에서 ServletContext 클래스의 getRealPath를 사용하여 주소를 얻고 Servlet에서 파일명을 붙여 전체 파일주소를 얻을 수 있다.

파일에서 데이터를 읽는 방법은 기본 자바 파일 입출력과 마찬가지로 BufferedReader와 StringTokenizer를 사용하여 파일을 읽었다.

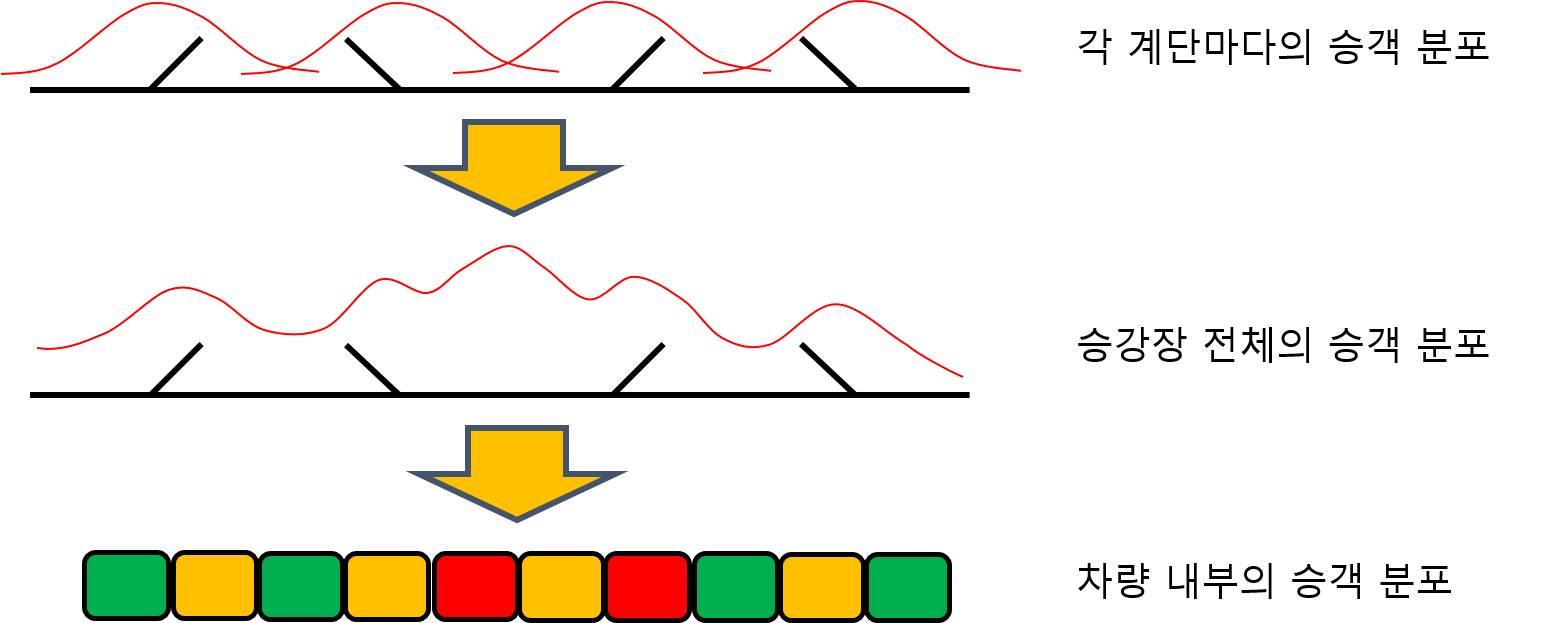
1. **JSON 파일**

JSON 파일을 파싱하고 각 인스턴스에 데이터를 집어넣기 위하여 Google에서 제공하는 json-simple(https://code.google.com/p/json-simple/) 라이브러리를 사용하여 데이터를 읽는다. 라이브러리의 JSONParser 클래스를 사용하여 JSON파일을 파싱한 다음 JSONArray와 JSONObject 클래스를 사용하여 원하는 데이터를 얻어낸 다음 데이터 클래스에 집어 넣는다.

JSON파일의 데이터 중 한글, 특수문자, 숫자가 혼합된 데이터가 있는데 (예 : 당고개(409)) 이러한 데이터에서 숫자만 뽑아내기 위하여 정규표현식을 사용하여 token을 떼어냈다.

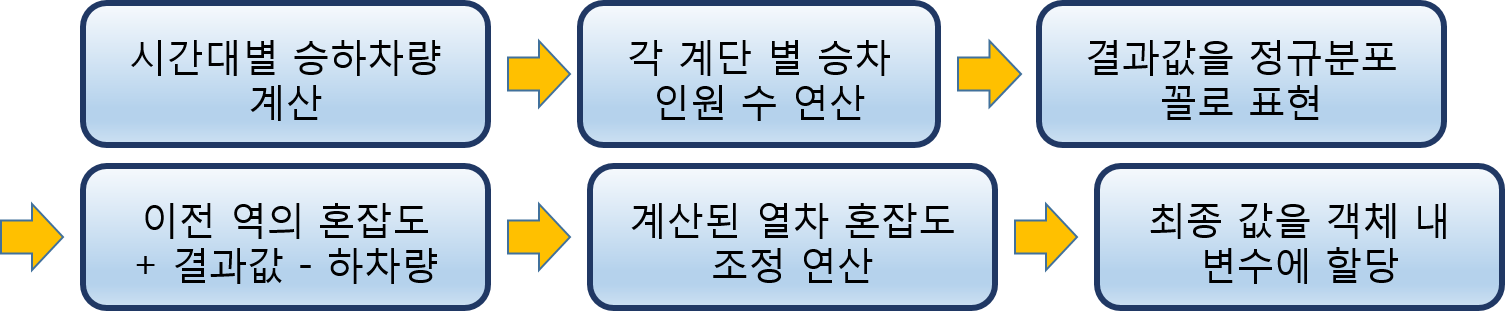
## **혼잡도 계산 알고리즘**

1. **기본 아이디어**



일반적으로 사람들이 지하철을 타러 승강장에 내려오면 자신이 내려온 계단 근처에서 지하철을 타게 된다. 또한 지하철을 탄 이후에도 많이 열차 안에서 많이 이동하지 않는 편이다. 따라서 계단 위치로부터 승객의 분포 정도를 정규분포 꼴로 나타낸 다음 각 열차 칸에 대입을 하면 차량 내부 혼잡도가 나올 것이라는 예상 하에 프로그램을 제작했다.

1. **알고리즘 설명**

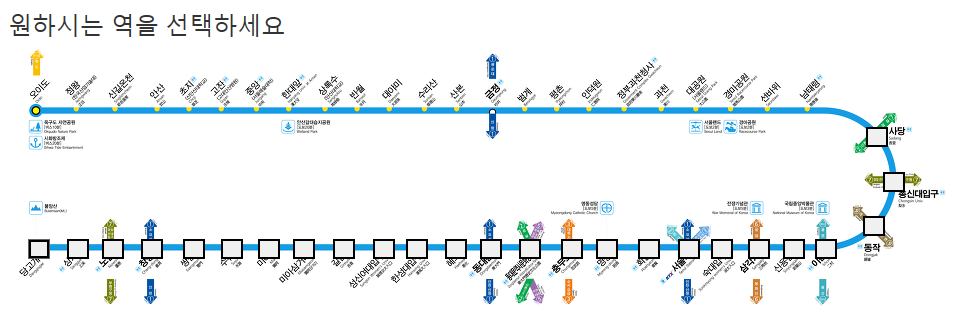


역의 시간대별 승하차량을 시간대별 배차간격으로 나눈 값이 한 열차 편성에 탈 승객이 된다. 이를 승강장 계단 수로 나눠 각 계단으로 몇 명이 내려올 지를 예상한 다음 이 값을 정규분포 형태로 분산시킨다. 이렇게 각 계단 별 분산된 인원 수를 이전 역의 열차 혼잡도와 하차 인원을 계산하여 최종 열차 혼잡도를 구한다.

아무리 열차 내에서 사람들이 잘 움직이지 않는다고 해도 옆 칸이 지금 칸에 비해 현저히 한적하다고 생각되면 사람들은 그 칸으로 이동하게 된다. 이를 반영하기 위해 최종 결과값에서 나온 혼잡도 중 옆 칸과 차이가 심한 경우는 임의의 수만큼 인원을 이동시키는 연산을 한다.

## **이벤트 처리 및 웹 페이지 구현**

1. **이벤트 처리 – 메인 화면**



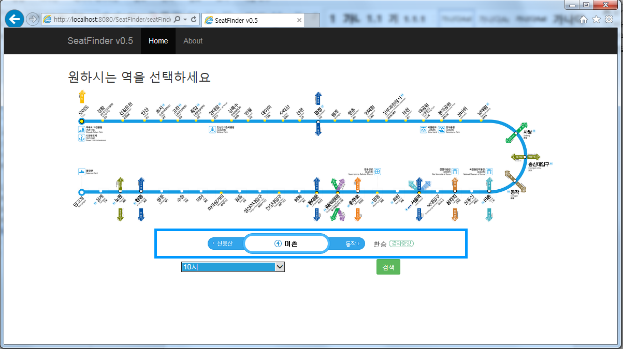
메인 화면으로부터 사용자가 역을 입력할 때 텍스트 박스 대신 위의 사진과 같이 노선도 이미지와 자바스크립트 버튼을 사용하여 역을 선택하도록 만들었다. 버튼은 정해진 위치에 배치 된 다음 opacity를 조정하여 화면에서 보이지 않게 만든다. 이로써 사용자의 의도치 않은 입력 오류를 차단할 수 있으며 직관적인 사용 방법에서 오는 편리함도 얻을 수 있다.

1. **이벤트 처리 – 결과 화면**



계산된 차량 내 인원 수를 토대로 혼잡도를 색깔로 표현한다. 각 열차에 마우스를 올리면 그 열차의 예상 인원 수를 보여준다.

1. **이벤트 처리 – Cross Browser**

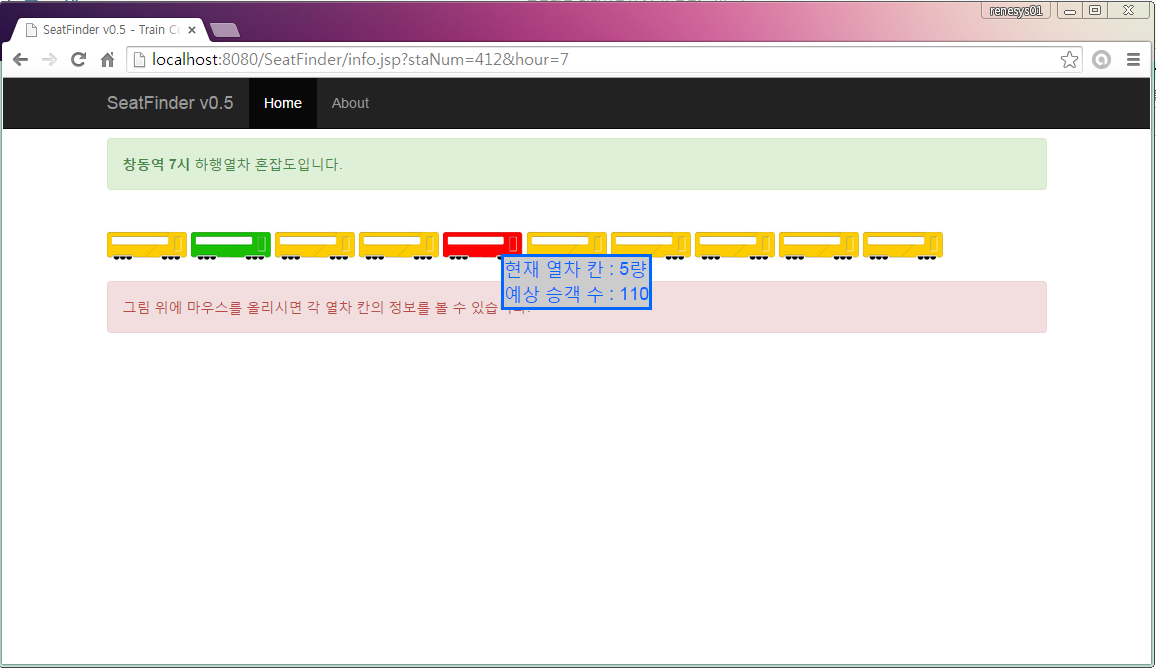
** **

자바스크립트 이벤트를 인터넷 익스플로러, 구글 크롬에서 모두 실행할 수 있도록 만들기 위해 이벤트를 호출하는 JSP파일에서 브라우저 별 이벤트 처리 코드를 넣었다.

1. **한글처리**

웹 페이지 상에서 한글 깨짐을 막기 위해 request 인스턴스의 setCharacterEncoding을 사용했으며 문자형을 EUC-KR에서 UTF-8로 바꿨다.

## **UI (Bootstrap)**

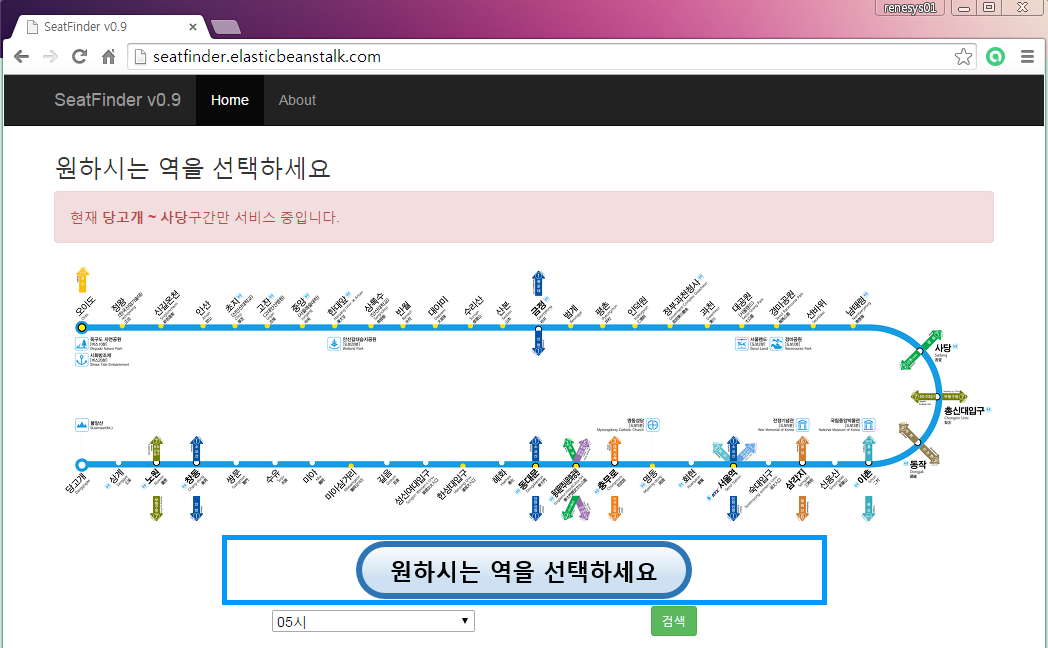


웹 페이지 디자인은 오픈 소스인 Bootstrap을 사용했다. Bootstrap은 JavaScript와 CSS 를 기반으로 만들어진 웹 디자인 라이브러리로JQuery가 설치되어야 사용할 수 있다. Bootstrap을 사용함으로써 깔끔한 헤드라인 디자인, 글꼴, 버튼, 안내 박스를 사용할 수 있었으며 라이브러리 특성상 크로스 브라우저 문제는 물론, 모바일에서도 구동이 가능하여 브라우저 환경에 구속 받지 않는 웹 페이지를 만들 수 있었다.

## **호스팅(AWS Elastic Beanstalk)**

완성된 웹 페이지를 실제 웹 상에 호스팅하기 위해 AWS(Amazon Web Service)의 Elastic Beanstalk를 사용하였다. Elastic Beanstalk는 여러 가지 언어들(JSP, PHP, Python)로 만든 웹 페이지코드를 원하는 서버환경을 선택하여 호스팅하는 방법이다. 이번 프로젝트의 코드는 JSP이므로 Apache Tomcat환경을 사용하였고 웹 인스턴스는 EC2를 사용하였다. 인스턴스와 구동 환경을 구축한 다음 완성된 프로젝트를 war파일로 생성하여 Elastic Beanstalk에 올림으로써 실제 웹 브라우저에서 접속할 수 있는 도메인 주소를 얻게 되었다.

완성된 웹 페이지 URL : http://seatfinder.elasticbeanstalk.com/

** **

# **개선/보완해야 할 점**

* 데이터베이스를 사용하지 않아 프로그램을 실행하는 방식이 효율적이지 못하다. 위에서도 한 번 언급했었지만 서블릿에서 만든 데이터를 DB에 저장하면 매번 Station 인스턴스를 생성하지 않고도 프로그램을 구동할 수 있을 것이다.
* 시간대별 승하차량 중 딱 하루의 데이터를 사용해서 계산 값의 결과가 실제 혼잡도랑은 크게 상이한 시간대가 존재한다. 특히 Normal Hour 시간대의 결과가 많이 어긋난다. 또한 혼잡도를 계산하는 알고리즘도 부족한 점이 있다. 이는 교통량 분석과 관련된 전문지식을 더 습득해야 해결될 문제로 생각된다.
* 현재 프로그램은 서울지하철 4호선(당고개~사당)의 하행열차의 혼잡도만 서비스하고 있다. 모든 노선의 혼잡도를 계산하기 위해서는 다른 회사의 데이터(특히 코레일)가 필요하고 서울 지하철 1호선이나 서울 지하철 9호선의 경우 급행열차도 있기 때문에 별도의 계산 알고리즘이 필요할 것으로 예상된다.

# **자기평가 및 소감**

작년에 웹 프로그래밍 과목을 수강했었지만 그 때 했던 과제들은 전부 HTML문서 기반으로 돌아가던 웹 프로그램이었다. 서버와 연동하여 실제 웹 서비스를 제작한 것은 이번이 처음이었다. 교통분야에 관심이 많아서 이쪽 분야의 어떤 것을 만들겠다는 생각은 프로젝트 계획 초창기부터 있었지만 문제는 ‘무엇을 써서 어떻게 만드는가’였다. 프로그램 특성상 복잡한 연산이 많이 필요했고 특히 웹 페이지와 하부 프로그램(혼잡도 계산)이 계속 연동돼야 했기 때문에 JSP를 사용하게 되었다.

무엇을 써서 만들지를 결정한 다음에도 문제는 끊임없이 나왔다. 특히 텍스트와 JSON 원본 데이터를 읽는 방식이 Java와 달라서 애를 먹었다. 이를 해결하기 위해 인터넷에서 서블릿과 MVC모델의 개념, request와 response 인스턴스 등, 많은 지식들을 찾아보고 공부했다.

제작 도중 필요한 기능과 부합하는 오픈 소스를 찾아내면 최대한 사용해보려고 노력했다. 덕분에 JSON 파일을 쉽게 사용할 수 있었으며 Bootstrap을 통해 실제 웹 상에서 서비스하는 사이트들의 분위기를 나름대로 낼 수 있었다.

완성된 프로그램을 AWS에 호스팅하여 실제로 서비스한 다음 다른 컴퓨터에서 돌아가는 것을 보았을 때 느낌 기쁨은 이루 말할 수 없었다. 그리고 지하철 역에서 스마트폰으로 내가 만든 웹 페이지에 접속하여 실제 혼잡도를 확인했을 때 (그게 맞는지 틀린지를 떠나) 오묘한 희열을 느꼈다. 비록 서툰 면이 있지만 다른 사람들에게 서비스 가능한 무엇인가를 만들었다는 것이 정말로 기분 좋았다.

여태까지 했던 프로그램 프로젝트 중 가장 막막하고 어렵게 시작했지만 그만큼 완성의 보람이 가장 컸던 프로젝트였으며 진행하면서 관련 지식들을 많이 공부했고 또한 이것 이외에도 공부할 것이 많이 있다는 것을 느꼈다.

# **참고문헌**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처(웹 주소) | 발행년도 | 저자 | 기타 |
| 1 | 웹문서 | 서울시 열린 데이터 광장 | http://data.seoul.go.kr |  |  |  |
| 2 | 웹문서 | Bootstrap | http://getbootstrap.com/ |  |  |  |
| 3 | 웹문서 | json-simple | https://code.google.com/p/json-simple/ |  |  |  |
| 4 | 웹문서 | AWS 시작 설명서 | http://aws.amazon.com/ko/documentation/gettingstarted/ |  |  |  |
| 5 | 웹문서 | tutorialspoint | http://www.tutorialspoint.com/json/json\_java\_example.htm |  |  |  |
| 6 | 웹문서 | StackOverflow | http://stackoverflow.com/ |  |  |  |