

인터넷프로그래밍 기말 프로젝트 보고서

12131819 육동현

1-1. 프로젝트 기획

요구사항 분석

- 사용자 요구사항 분석 (스터디그룹 의견 반영)

- 1) 전날 무엇을 어떻게 공부를 했는지 기억이 나지 않는다.
- 2) 내가 공부를 얼마나 하는지 언제 하는지 공부패턴을 알고 싶다.
- 3) 다른 멤버는 어느 정도 공부하는지 궁금하다.
- 4) 스터디그룹을 관리할 수 있는 편리한 수단이 있으면 좋겠다.

프로젝트 설계

- 프로그램 이름: Study Logger (스터디로거)
- 프로그램 목적: 자신의 학습로그를 기록하고 자신이 공부한 시간을 스터디그룹 멤버와 공유하며 상호 피드백을 할 수 있는 플랫폼 제공
- 개발언어: Java, JSP, Servlet, Firebase, HTML5, CSS3, JS(jQuery)

1-2. 비즈니스로직 설명

비즈니스로직 설계

- 제공기능 명세

- 1) 하루 동안의 학습을 기록하는 기능을 제공
- 2) 입력된 로그의 수정 및 삭제를 할 수 있도록 UI 구성 (실수 정정)
- 3) 학습인증게시판을 통해 다른 사람들과 학습시간을 비교할 수 있고
서로의 학습시간을 보면서 상호 동기부여
- 4) 당일 뿐만 아니라 과거의 학습로그를 조회할 수 있는 페이지 제공

첫 번째 기능 설계

- 하루 동안의 학습을 기록하는 기능

1. 기록시작 버튼을 누른다
2. 시작시간이 기록된다
3. 기록완료 버튼을 누른다
4. 종료시간, 학습명이 기록된다

2018년 12월 5일 (수)

08:54~09:12 // 등교준비

09:12~10:00 // 등교

10:00~10:16 // 은행업무

10:16~11:00 // 티켓예약

11:00~12:25 // 인프과제 <3>

12:25~12:42 // 점심식사

12:42~13:30 // 인프과제 <3>

13:30~14:40 // 소공수업

14:40~ // 인프과제

휴대폰의 메모장을
이용하여 기록하던 방식을
그대로 웹에서 구현하고자 함

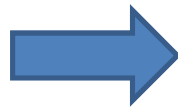
첫 번째 기능 구현 #1

1) 버튼을 통한 제어기능 구현 - ctrl()

ctrl() -> 사용자에게 제어할 수 있는 인터페이스 제공 (wrapping)

```
if(flag)
    flip the flag
    버튼에 "기록완료"라고 표시
    insertBegin( ) 호출
else
    if(input에 입력 있음)
        flip the flag
        버튼에 "기록시작"이라고 표시
        insertEnd( ) 호출
        다음 입력을 위해 input 초기화
    else
        입력이 없다고 경고 메시지 출력
```

의사코드



```
/* 기록제어 */
function ctrl()
{
    if (flag)
    {
        flag = !flag; // flip
        document.getElementById("btn").value = "기록완료";
        insertBegin();
    }
    else
    {
        if (document.getElementById("task").value) // 입력이 NULL인지 확인
        {
            flag = !flag; // flip
            document.getElementById("btn").value = "기록시작";
            insertEnd();
            document.getElementById("task").value = ""; // 입력박스 초기화
        }
        else
            alert("내용이 입력되지 않았습니다!");
    }
}
```

구현된 코드

ctrl() 메소드를 통해서 사용자가 입력을 제어할 수 있는 인터페이스를 제공하며
ctrl() 메소드에서는 시작시간을 입력하는 insertBegin() 메소드와
종료시간을 입력하는 insertEnd() 메소드를 호출합니다.

첫 번째 기능 구현 #2-1

2) 시작시간이 기록 – insertBegin()

insertBegin() -> 시작시간을 입력받고 DB에 기록
현재시간을 받아옴 // ex) 15:21:10 ~
받은 시간을 화면에 출력
데이터베이스에 기록

의사코드



1단계 : 현재시간을 받아와서 동적으로 요소를 생성

```
var now = new Date(); // 현재시간

var log = document.createElement("div"); // 컨테이너
log.setAttribute("id", "log" + beginCnt);
log.setAttribute("style", "margin: 5px;");

var begin = document.createElement("span"); // 시작시간
begin_time = ((now.getHours() < 10) ? "0" : "") + now.getHours() + ":"
              + ((now.getMinutes() < 10) ? "0" : "") + now.getMinutes()
              + ":" + ((now.getSeconds() < 10) ? "0" : "") + now.getSeconds();
begin.setAttribute("id", "begin" + beginCnt);
begin.innerHTML = ((now.getHours() < 10) ? "0" : "") + now.getHours() + ":";
begin.innerHTML += ((now.getMinutes() < 10) ? "0" : "") + now.getMinutes();
begin.innerHTML += ":" + ((now.getSeconds() < 10) ? "0" : "") + now.getSeconds();
begin.setAttribute("style", "font-weight:bold;color:black;");
```

첫 번째 기능 구현 #2-2

2단계: 동적으로 생성된 요소를 페이지에 삽입하여 출력

```
// 로그 한 줄에 삽입
log.appendChild(begin);
log.appendChild(tilde);

// 전체 로그에 삽입
document.getElementById("taskLog").appendChild(log);
```

3단계 : 데이터베이스 업데이트

```
var id = getParameterByName('user_id');
var today = getTimestamp();

/* 세팅 */
var user_log = firebase.database().ref('user_log/' + id + '/' + today + '/log' + beginCnt);
user_log.set
({
  begin_time : begin_time,
  end_time : end_time,
  task_name : task_name,
  study_duration : study_duration
});
```

첫 번째 기능 구현 #3-1

3) 종료시간이 기록 – insertEnd()

insertEnd() -> 종료시간을 입력받고 DB에 기록
현재시간을 받아옴 // ex) 15:25:20
받아온 시간과 입력된 학습명을 화면에 출력
데이터베이스에 기록
학습시간을 구함 // (종료시간 - 시작시간)
학습시간을 데이터베이스에 기록

의사코드



1단계 : 현재시간과 학습명을 받아와서 동적으로 요소를 생성

```
var now = new Date(); // 현재시간

var end = document.createElement("span"); // 종료시간
end_time = ((now.getHours() < 10) ? "0" : "") + now.getHours() + ":"
            + ((now.getMinutes() < 10) ? "0" : "") + now.getMinutes()
            + ":" + ((now.getSeconds() < 10) ? "0" : "") + now.getSeconds();
end.setAttribute("id", "end" + endCnt);
end.innerHTML = ((now.getHours() < 10) ? "0" : "") + now.getHours() + ":";
end.innerHTML += ((now.getMinutes() < 10) ? "0" : "") + now.getMinutes();
end.innerHTML += ":" + ((now.getSeconds() < 10) ? "0" : "") + now.getSeconds();
end.setAttribute("style", "font-weight:bold;color:black;");

var cmt = document.createTextNode(" // "); // 주석표시

var task = document.createElement("span"); // 업무설명
task.setAttribute("id", "task" + endCnt);
task.innerHTML = document.getElementById("task").value + "&nbsp;&nbsp;&nbsp;";
task_name = document.getElementById("task").value;
```

첫 번째 기능 구현 #3-2

2단계: 동적으로 생성된 요소를 페이지에 삽입하여 출력

```
// 로그 한줄에 삽입
document.getElementById("log" + endCnt).appendChild(end);
document.getElementById("log" + endCnt).appendChild(cmt);
document.getElementById("log" + endCnt).appendChild(task);
document.getElementById("log" + endCnt).appendChild(nbsp);
document.getElementById("log" + endCnt).appendChild(rmv);
document.getElementById("log" + endCnt).appendChild(nbsp);
document.getElementById("log" + endCnt).appendChild(edit);
document.getElementById("log" + endCnt).appendChild(hr);
```

3단계: 종료시간과 학습명을 데이터베이스에 기록

```
var id = getParameterByName('user_id');
var user_log = firebase.database().ref('user_log/' + id + '/' + getTimestamp() + '/log' + endCnt);

user_log.update
({
  end_time : end_time,
  task_name : task_name
});
```

첫 번째 기능 구현 #3-3

4단계: 학습시간을 구하는 처리를 수행

```
// parsing을 함
var beginParse = begin_time.split(":");
var endParse = end_time.split(":");

// Date 객체 생성하여 시간차이를 구함
var b = new Date("0", "0", "0", beginParse[0], beginParse[1], beginParse[2], "0"); // 시작시간
var e = new Date("0", "0", "0", endParse[0], endParse[1], endParse[2], "0"); // 끝 시간
var diff = (e.getTime() - b.getTime()) / 1000; //sec 단위로 시간차이 나타냄

result += diff;
var temp = result;

var sec = result % 60;
result -= sec;
result /= 60;
var min = result % 60;
var hr = result / 60;

result = temp;

var str = (hr.toFixed(0) > 0) ? (hr.toFixed(0) + "시간 ") : "";
str += (min.toFixed(0) > 0) ? (min.toFixed(0) + "분 ") : "";
str += (sec.toFixed(0) > 0) ? (sec.toFixed(0) + "초") : "0초";
acc_time = str;
```

5단계: 학습시간을 데이터베이스에 기록

```
/* 누적학습시간 계산 */
function calculate()
{
    var id = getParameterByName('user_id');
    var user_time = firebase.database().ref('user_time/' + id + '/' + getTimestamp());
    user_time.set
    ({
        user_time : acc_time
    });
}
```

두 번째 기능 설계

- 입력된 로그의 수정 및 삭제를 할 수 있도록 UI 구성 (실수 정정)
 1. 수정버튼을 누른다
 2. 학습명이 수정된다
 3. 삭제버튼을 누른다
 4. 로그가 삭제된다

두 번째 기능 구현 #1

editing() -> 수정할 학습명을 입력받고 DB 갱신
수정할 학습명을 입력받음
데이터베이스 업데이트 **의사코드**

의사코드



```

/* 로그수정 */
function editing()
{
    var target = this.parentNode.getElementsByTagName("span")[2];
    task_name = prompt("변경할 업무이름을 입력해주세요");
    target.innerHTML = task_name + "    ";

    var editId = this.id;
    var parse = editId.split("edit");

    var id = getParameterByName('user_id');
    var user_log = firebase.database().ref('user_log/' + id + '/' + getTimeStamp() + '/log' + parse[1]);

    user_log.update
    ({
        task_name : task_name
    });
}

```

두 번째 기능 구현 #2

remove() -> 삭제버튼을 누르면 로그가 삭제되고 DB에서 제거
정말 삭제할 것인지 사용자에게 물어봄
동의한 경우 동적으로 삭제
DB에서도 삭제

의사코드



```
/* 로그삭제 */  
function remove()  
{  
    var ret = confirm("정말 삭제하시겠습니까?");  
    if (ret)  
    {  
        var c = this.parentNode;  
        var p = c.parentNode;  
        p.removeChild(c);  
  
        var rmvId = this.id; // 삭제할 대상 노드의 id를 받아옴  
        var parse = rmvId.split("rmv"); // 파싱함  
  
        var id = getParameterByName('user_id');  
        var user_log = firebase.database().ref('user_log/' + id + '/' + getTimestamp() + '/log' + parse[1]);  
        user_log.remove();  
    }  
    else  
        alert("삭제되지 않았습니다");  
}
```


세 번째 기능 설계

- 학습인증게시판을 통해 다른 사람들과 학습시간을 비교할 수 있고 서로의 학습시간을 보면서 상호 동기부여
 - 필요사항 : 작성글에 작성자와 학습시간이 표시되도록 함
1. 게시판에 들어간다
 2. 작성글을 출력하는데 작성시점에 반드시 작성자와 학습시간이 포함하게 하여 출력한다

세 번째 기능 구현 #1

1단계: 동적으로 요소를 생성

```
// 동적으로 element 생성
var div = document.createElement("div");
var h3 = document.createElement("h3");
var p = document.createElement("p");
var hr = document.createElement("hr");

// 동적으로 생성된 요소의 CSS 설정
h3.setAttribute("style", "font-family:'Jua',sans-serif;");
p.setAttribute("style", "font-size:20px;font-family:'Jua',sans-serif;");
```

2단계: 작성글의 날짜를 파싱한다

```
// 포스트날짜를 파싱하는 처리
var str = tmp.post_date;
var parse1 = String(str).split("-");
var parse2 = String(parse1[2]).split(" ");
var result = parse1[0] + parse1[1] + parse2[0];
```

세 번째 기능 구현 #2

3단계: 해당되는 정보를 DB로부터 가져와 화면에 출력한다

```
/* 게시판 업데이트 하는 처리 */
var user_post = firebase.database().ref('user_post');
user_post.on('value', function(snapshot) {
  snapshot.forEach(function(childSnapshot) {
    var tmp = childSnapshot.val();

    // DB로 부터 내용 가져옴
    h3.innerHTML = tmp.post_title;
    p.innerHTML = "한 줄 응원 : " + tmp.post_content;
    p.innerHTML += "<br>" + "작성자 : " + tmp.user_id + "<br>" + "작성시간 : " + tmp.post_date;

    // 해당 게시글 작성시점에 학습시간 얻어오기 위한 처리
    var study_time = "";
    var ref = firebase.database().ref('user_time/' + tmp.user_id + '/' + result);
    ref.once("value").then(function(snapshot) {
      study_time = snapshot.child("user_time").val();
      console.log(study_time);
      p.innerHTML += "<br>" + "학습시간 : " + study_time;
    });
  });
  // 정리하자면 {"키 값" : "값"}에서 value를 접근하려면 snapshot.child("키 값").val()을 사용해야 한다
```

네 번째 기능 설계

- 당일 뿐만 아니라 과거의 학습로그를 조회할 수 있는 페이지 제공
 1. 사용자로부터 조회할 날짜를 입력 받음
 2. 해당 날짜에 해당되는 로그에 접근
 3. 로그를 화면에 출력해줌

네 번째 기능 구현 #1

1단계: 사용자로부터 조회할 날짜를 입력 받음

```
<label class="jua" style="font-size:20px;">날짜 선택 : </label>
<%
    Date date = new Date();
    String modifiedDate= new SimpleDateFormat("yyyy-MM-dd").format(date);
%>
<input type="date" id="date" value="<%= modifiedDate %>" class="jua">
<input type="button" value="선택완료" id="select" onclick="generateReport()"
```

2단계: 해당 날짜에 해당되는 로그에 접근

```
var input = document.getElementById("date").value; // input에서 값 받아옴
var parse = input.split("-"); // input값을 파싱함
var dateStr = "" + parse[0] + parse[1] + parse[2]; // 파싱한 문자열을 하나로 묶음
var id = getParameterByName('user_id'); // user_id 받아옴
var user_log = firebase.database().ref('user_log/' + id + '/' + dateStr); // DB에 접근
```

네 번째 기능 구현 #2

3단계: 데이터를 화면에 출력

```
// 동적으로 element 생성
var div = document.createElement("div");
div.setAttribute("style", "font-family:'Jua', sans-serif;font-size:20px;");
var task_name = document.createElement("span");
var colon = document.createTextNode(" : ");
var start_time = document.createElement("span");
var tilde = document.createTextNode(" ~ ");
var end_time = document.createElement("span");

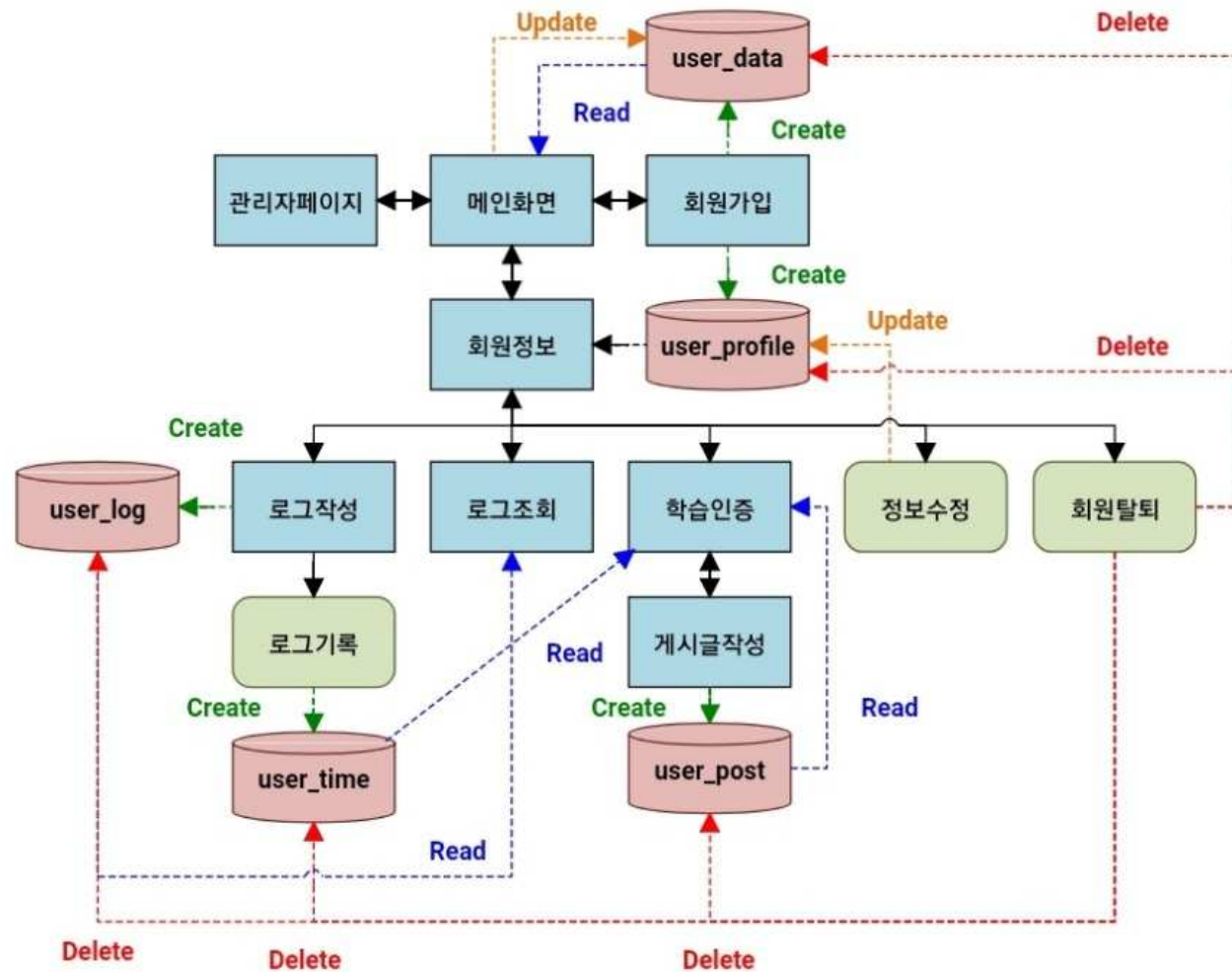
// DB로 부터 내용 가져옴
task_name.innerHTML = tmp.task_name;
start_time.innerHTML = tmp.begin_time;
end_time.innerHTML = tmp.end_time;

// HTML에 요소 삽입
div.appendChild(task_name);
div.appendChild(colon);
div.appendChild(start_time);
div.appendChild(tilde);
div.appendChild(end_time);
document.getElementById("record").appendChild(div);

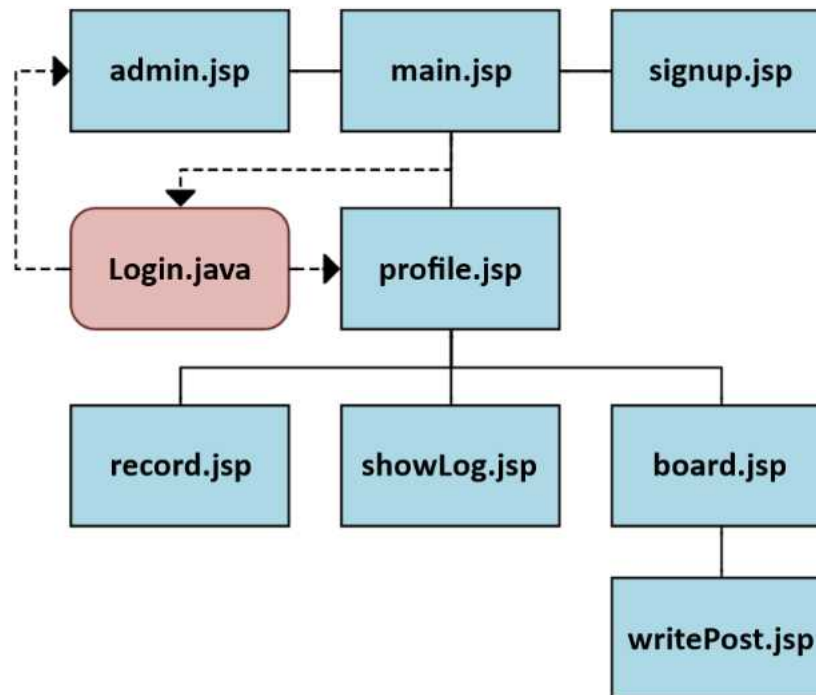
// 조회날짜에 학습시간 얻어오기 위한 처리
var study_time = "";
var ref = firebase.database().ref('user_time/' + id + '/' + dateStr);
ref.once("value").then(function(snapshot) {
    study_time = snapshot.child("user_time").val();
    var study = document.createElement("div");
    study.innerHTML = "학습시간 : " + study_time;
    study.setAttribute("style", "font-family:'Jua', sans-serif;font-size:20px;color:navy;");
    document.getElementById("record").appendChild(study);
});
```

2-1. 웹 페이지 구성

웹 페이지 아키텍처



Web Page 구성



저는 이번 프로젝트를 위해서 8개의 JSP 페이지와 1개의 Servlet을 만들었습니다. 각 페이지의 아키텍처는 위와 같습니다.

메인화면



위의 메인 화면은 입력필드와 버튼을 수정하였고,

구글 폰트를 이용하여 글씨체를 변경하였습니다.

또한 twitter 대신 github 아이콘으로 바꾸고 링크를 연결하였습니다.


관리자 페이지

스터디원 관리

ID	이름	학년	전공	성별	관심사	접속시간	관리
12345	웹마스터	NULL	NULL	NULL	NULL	2018-12-05 16:19:00	삭제
12131819	육동현	4학년	컴퓨터공학	남자	웹 프로그래밍	2018-12-05 13:33:54	삭제

관리자 계정으로 로그인한 경우에만 표시되는 화면입니다.

회원가입/회원정보



저장하기
메인으로

아이디

비밀번호

이름


성별 ☒ 남자 ☐ 여자

전공

학년

관심사

중복확인



로그아웃
로그재입
인증하기
정보수정
회원탈퇴
로그아웃

12131819님 반갑습니다!

아이디 12131819

이름 육동현

전공 컴퓨터공학

학년 4학년

성별 ☒ 남자 ☐ 여자

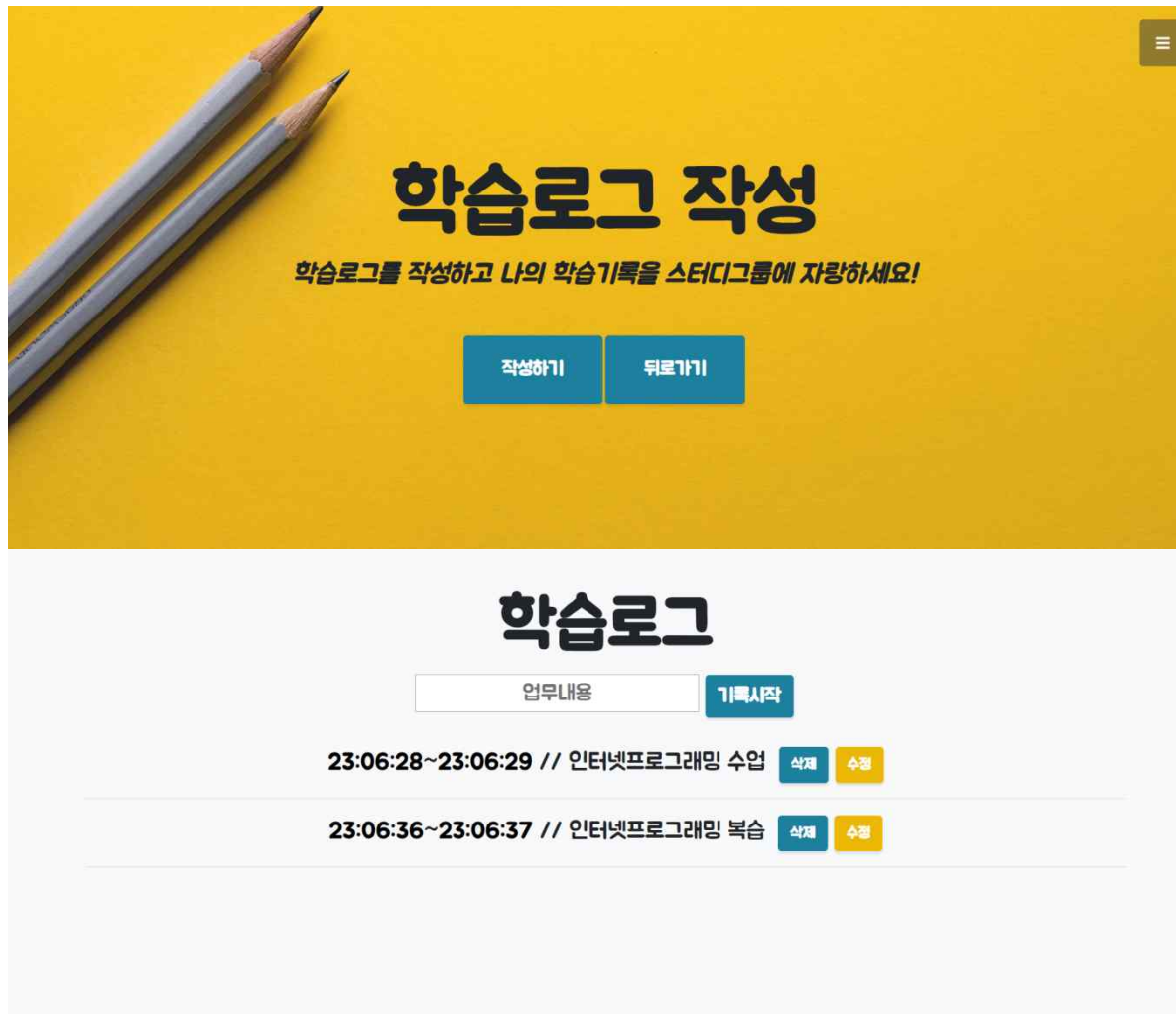
관심사 웹 프로그래밍

학습시간 0초

스터디멤버 12345,12131789,12131819,12131820,12131821

위의 회원가입과 회원정보 페이지는 앞과 마찬가지로 구글 폰트를 적용하였고, 그 외에 사진과 네비게이션 바를 일부 수정하였습니다. 또한 기존에 만들었던 웹 폼을 삽입하여 동작할 수 있도록 하였습니다.

로그작성 #1



- 기존의 코드에서
제가 필요로 하는
버튼을 추가적으로
만들었습니다.

-구글 폰트를 적용했습니다.

- 학습로그는 비즈니스로직을
직관적으로 사용할 수 있게
자체적으로 디자인했습니다.

로그작성 #2

학습로그

기록시작

23:06:28~23:06:29 // 인터넷프로그래밍 수업

삭제

수정

23:06:36~23:06:37 // 인터넷프로그래밍 복습

삭제

수정

×

현재시간



23시 07분 04초

위의 오른쪽 사이드의 현재 시간 보기기능은 HTML5의 canvas를 이용하여 기존의 네비게이션 기능과는 다르게 위젯과 유사한 기능을 할 수 있도록 디자인하였습니다.

학습인증게시판



기말고사 화이팅!

한 줄 응원 : 모두 기말고사 화이팅하세요!

작성자 : 12131819

작성시간 : 2018-12-04 23:08:14

학습시간 : 0초

구글 폰트를 적용하였고,
네비게이션 바를 일부 수정하였습니다.
게시글은 DB를 조회하여 출력하도록 하였습니다.

게시글 작성

Study Logger

포스트 작성하기

제목

한 줄 응원

작성하기

뒤로가기

원본 코드에서 네비게이션 바를 수정하고 나머지 코드는 모두 지우고 지운 자리에 제가 작성한 포스트 작성 폼을 삽입하였습니다.

로그조회

Study Logger

로그조회

날짜 선택 : 12/03/2018 선택완료

<20181203>

인프복습 : 23:48:39 ~ 23:48:39

자구복습 : 23:48:46 ~ 23:48:52

인프복습 : 23:45:44 ~ 23:45:44

학습시간 : 6초

앞과 마찬가지로 원본 코드에서 네비게이션 바를 수정하고 나머지 코드는 모두 지웠습니다. 그리고 지운 자리에 제가 작성한 포스트 작성 폼을 삽입 하였습니다. (자세한 DB 관련 언급은 뒤에서 하겠습니다.)

2-2. 화면 디자인

부트스트랩 적용 #1

```
<!-- Bootstrap core CSS -->
<link href="bootstrap/main/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom fonts for this template -->
<link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:200,200i,300,300i,400,400i,600,600i,700,700i,900,900i" rel="stylesheet">
<link href="https://fonts.googleapis.com/css?family=Merriweather:300,300i,400,400i,700,700i,900,900i" rel="stylesheet">
<link href="bootstrap/main/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">

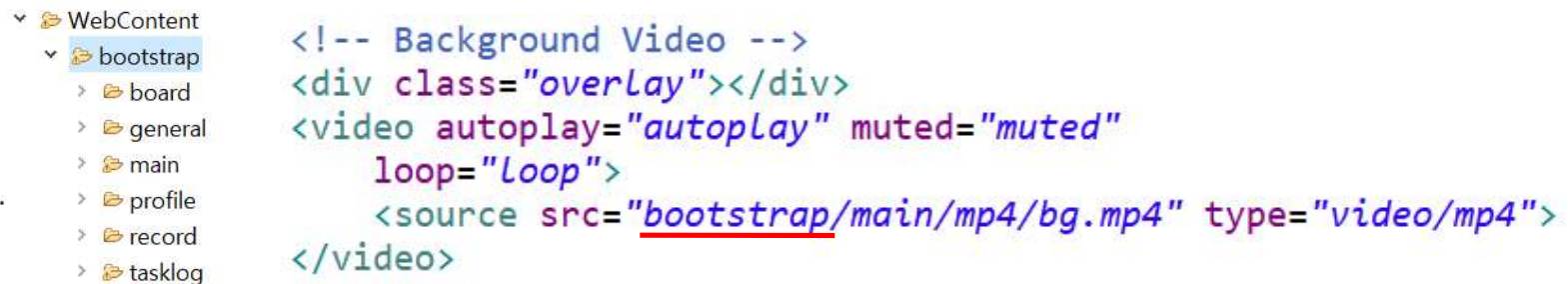
<!-- Custom styles for this template -->
<link href="bootstrap/main/css/coming-soon.min.css" rel="stylesheet">
```

위와 같은 방식으로 부트스트랩을 import하였습니다.

```
<li class="list-unstyled-item">
  <a href="https://github.com/DustinYook">
    <i class="fab fa-github"></i> 원래는 twitter였으나 수정
  </a>
</li>
```

또한 위와 같이 기존 코드를 제 요구사항에 맞게 수정하며 사용했습니다.

부트스트랩 적용 #2



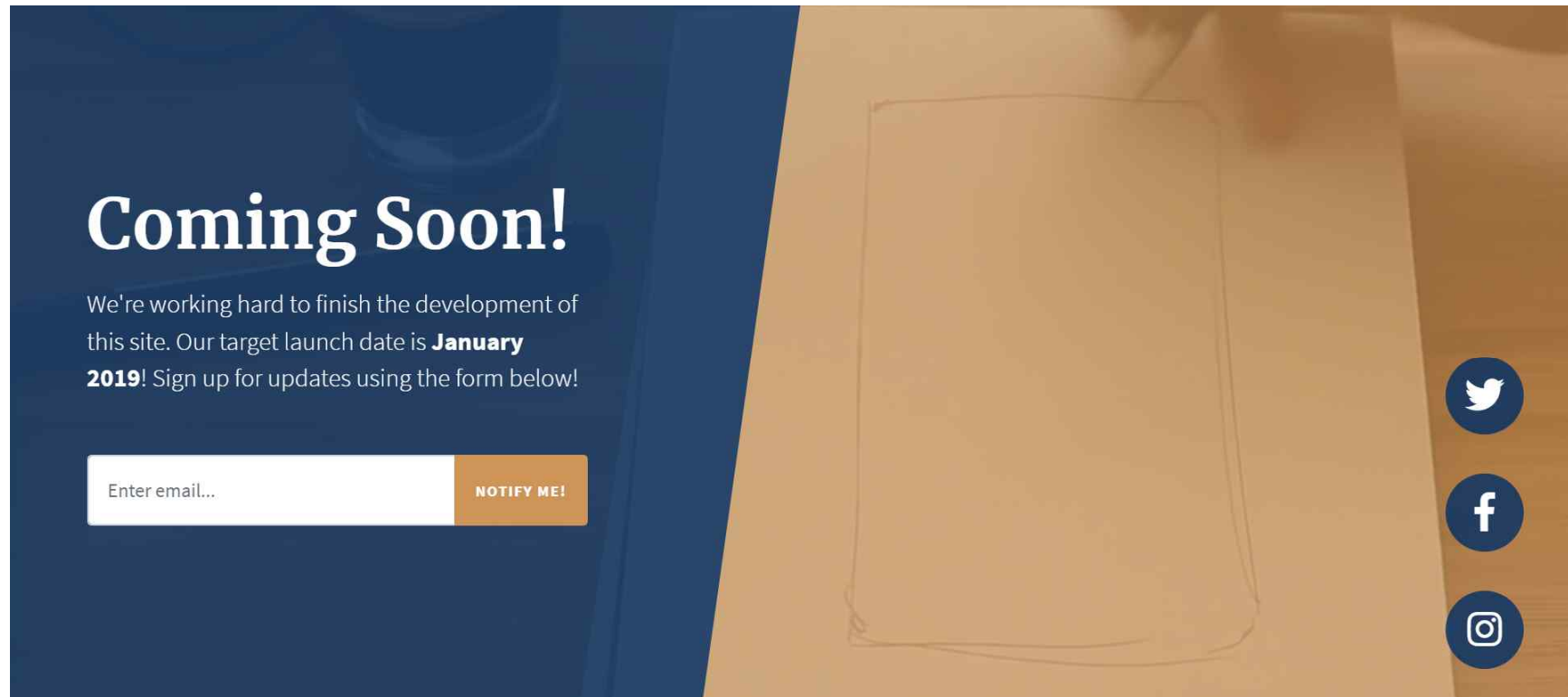
모든 부트스트랩 파일은 관리의 용이성을 위해 bootstrap이라는 폴더를 별도로 생성하여 위와 같은 방식으로 URI를 통해 접근하였습니다.

구글 폰트 이용

```
<!-- CSS Style Sheet -->
<style>
  @import url('https://fonts.googleapis.com/css?family=Black+Han+Sans|Jua');
  #jua1, #jua2 { font-family: 'Jua', sans-serif; }
</style>


// 동적으로 element 생성
var div = document.createElement("div");
div.setAttribute("style", "font-family:'Jua', sans-serif;font-size:20px;");
var task_name = document.createElement("span");
var colon = document.createTextNode(" : ");
var start_time = document.createElement("span");
var tilde = document.createTextNode(" ~ ");
var end_time = document.createElement("span");
```

원본 #1



[출처] <https://startbootstrap.com/template-overviews/coming-soon/>

원본 #2







- ABOUT
- EXPERIENCE
- EDUCATION
- SKILLS
- INTERESTS
- AWARDS

CLARENCE TAYLOR

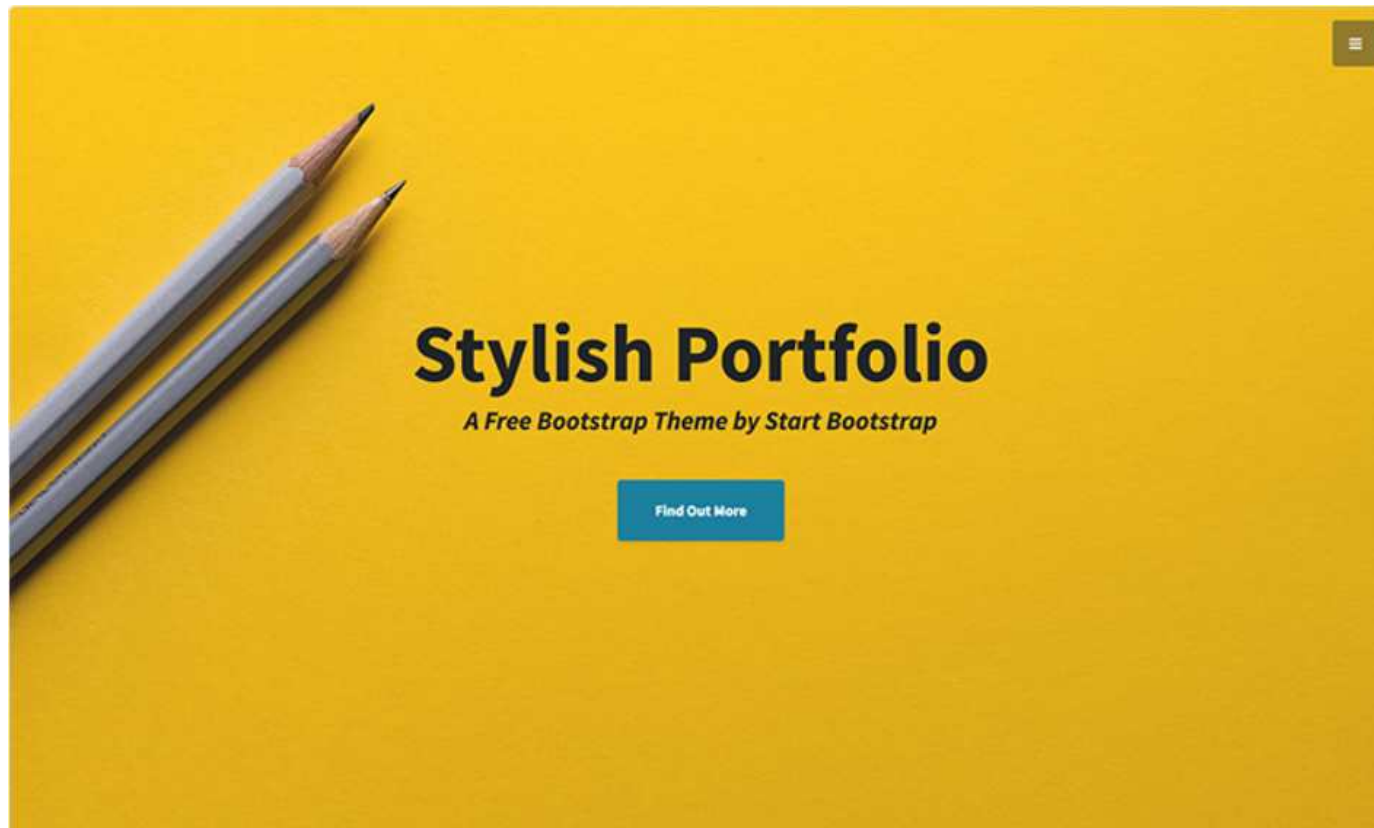
3542 BERRY STREET · CHEYENNE WELLS, CO 80810 · (317) 585-8468 · NAME@EMAIL.COM

I am experienced in leveraging agile frameworks to provide a robust synopsis for high level overviews. Iterative approaches to corporate strategy foster collaborative thinking to further the overall value proposition.



[출처] <https://startbootstrap.com/template-overviews/resume/>

원본 #3-1



[출처] <https://startbootstrap.com/template-overviews/stylish-portfolio/>

원본 #3-2



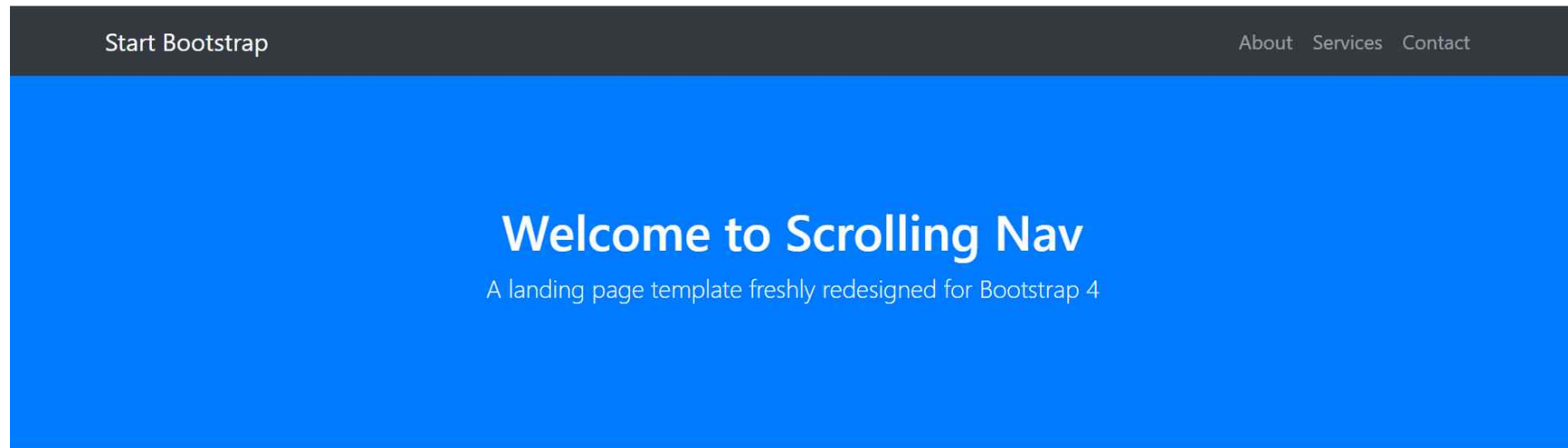
[출처] <https://startbootstrap.com/template-overviews/stylish-portfolio/>

원본 #4



[출처] <https://blackrockdigital.github.io/startbootstrap-clean-blog/>

원본 #5



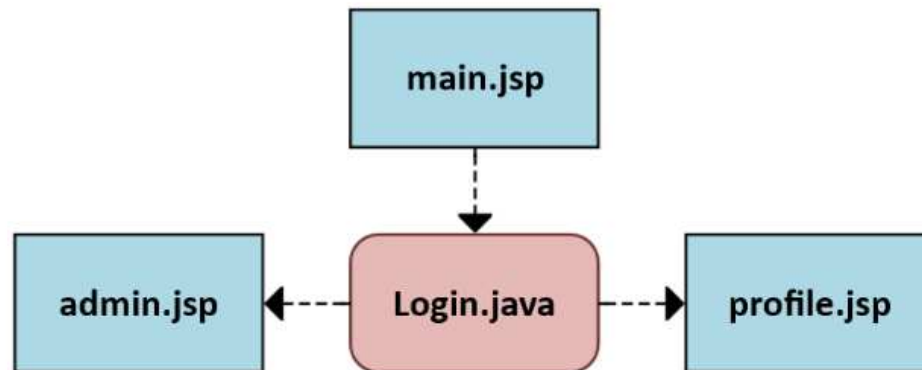
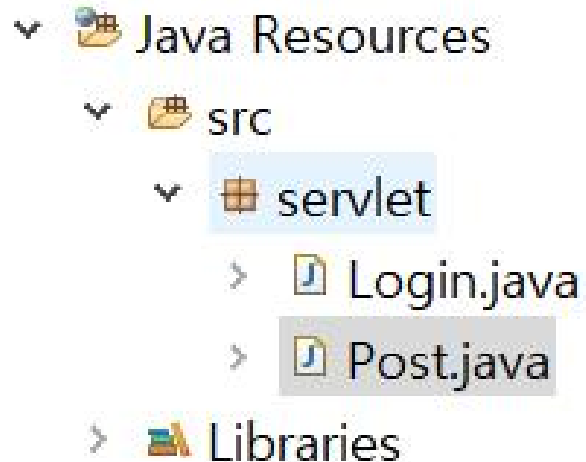
About this page

This is a great place to talk about your webpage. This template is purposefully unstyled.

[출처] <https://startbootstrap.com/template-overviews/scrolling-nav/>

2-3. 서블릿의 사용

서블릿의 역할 정의



main.jsp에서 Login.java 서블릿을 통해 관리자 계정인 경우 admin.jsp로 그렇지 않은 경우 profile.jsp로 이동을 제어하도록 역할을 정의하여 사용하였습니다.

서블릿의 구현 #1

아래와 같이 main.jsp에서 서블릿으로 정보를 전달합니다.

```
<!-- FORM -->
<form id="Login" method="POST">
  <table>
    <tr>
      <td><input type="text" id="id" value="" />
    </tr>
    <tr>
      <td><input type="password" id="pw" value="" />
    </tr>
    <tr>
      <td>
        <input class="btn btn-s" type="button" value="로그인" />
        <input class="btn btn-s" type="button" value="회원가입" />
      </td>
    </tr>
  </table>
</form>
```

```
var form = $("#login");
form.attr("action", "./Login?id=" + id + "&&pw=" + pw + "");
form.submit(); // form 제출 -> Login.java로 이동 -> Servlet에 의해 처리
```

아래와 같이 Login.java에서 어노테이션을 통해 서블릿을 매핑합니다.

이 경우 별도로 xml에 서블릿을 등록하지 않습니다. (충돌방지)

```
import javax.servlet.annotation.WebServlet;

/** Servlet implementation class LoginServlet */
@WebServlet("/Login")
public class Login extends HttpServlet
```

서블릿의 구현 #2

```
/** @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response) */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
    HttpSession session = request.getSession();
    String id = request.getParameter("id");
    String pw = request.getParameter("pw");

    session.setAttribute("id", id);
    session.setAttribute("pw", pw);

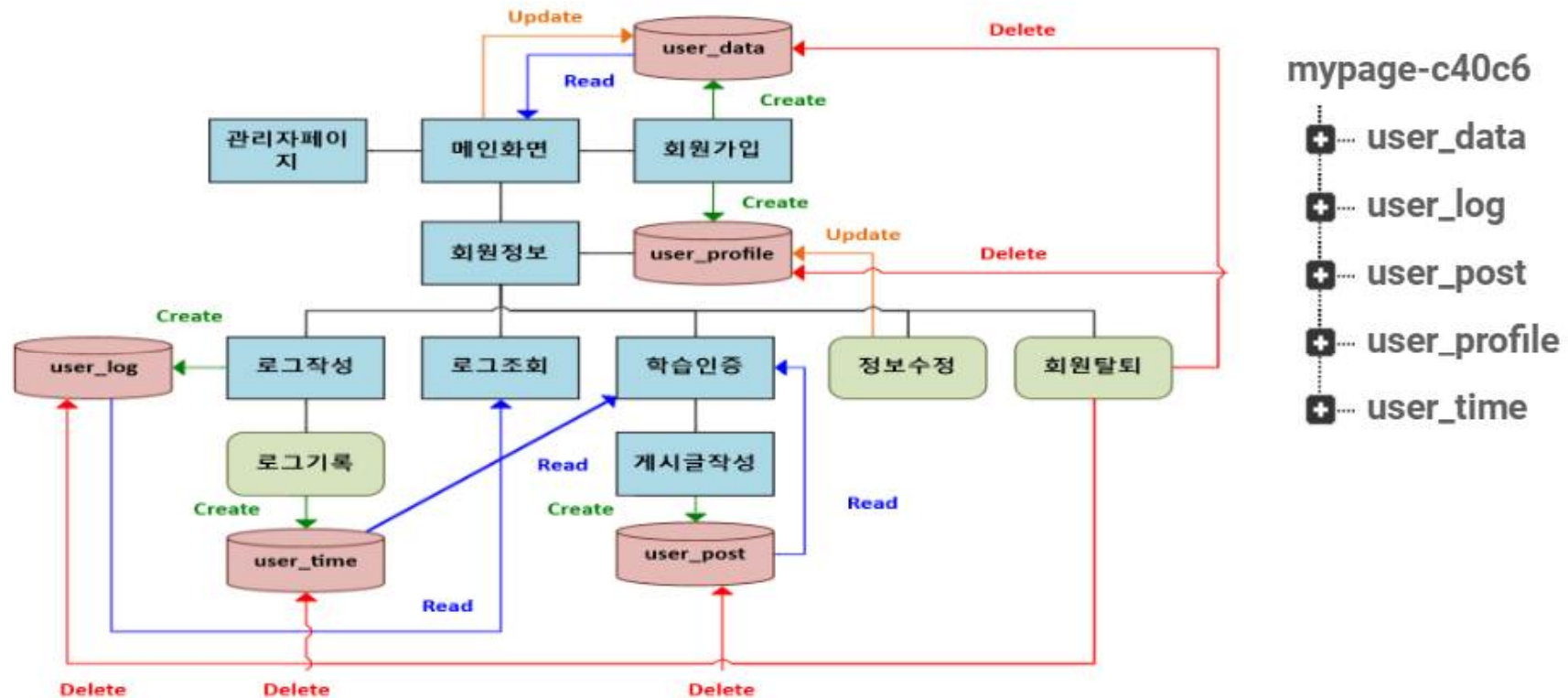
    // 관리자 계정으로 로그인
    if(id.equals("12345") && pw.equals("12345"))
    {
        String admin = "admin.jsp";
        request.getRequestDispatcher(admin).forward(request, response);
    }
    // 일반 계정으로 로그인
    else
    {
        String login = "profile.jsp?user_id=" + id;
        request.getRequestDispatcher(login).forward(request, response);
    }
}

/** @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response) */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
    doGet(request, response);
}
```

앞서 main.jsp에서 form에 정의된 method 방식으로 값을 전달받아 doPost() 또는 doGet() 메소드가 Login.java에서 호출됩니다. 본 서블릿에서는 관리자 계정인지 아닌지에 따라 다른 페이지로 이동시킵니다. 여기서 getRequsetDispatcher()를 통해 처리 속도를 향상시킵니다.

3-1. 데이터베이스 구성

Database 구성



이번 프로젝트에서 사용된 데이터베이스는 총 5개입니다.
모든 데이터베이스는 CRUD 방법을 통해 작동합니다.

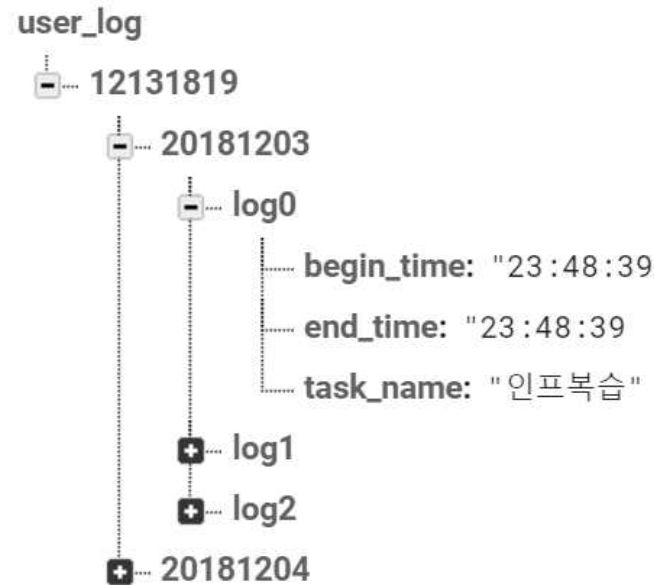
테이블 구성 #1

user_data

- 12345	last_login: "2018-12-04 19:28:5"
	user_id: "12345"
	user_pw: "12345"
+ 12131819	

user_data 테이블은 사용자의 아이디, 비밀번호,
마지막 로그인 시간을 저장합니다.

테이블 구성 #2



user_log 테이블은 사용자의 아이디 별로 로그를 관리하며, 하위에 날짜를 통해 날짜 별로 로그를 관리합니다. 로그는 작성 순서에 따라 0번부터 순서가 매겨집니다. 가장 하위요소인 log는 학습 시작시간, 학습종료시간, 그리고 학습내용을 저장할 수 있도록 구성하였습니다.

테이블 구성 #3

user_post

12131819

```
post_content: "모두 기말고사 화이팅하세요!"  
post_date: "2018-12-04 23:08:1"  
post_title: "기말고사 화이팅!"  
user_id: "12131819"
```

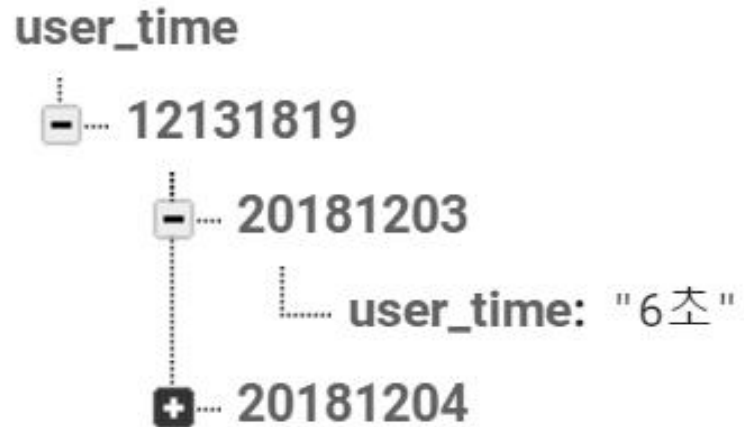
user_post 테이블은 사용자의 아이디 별로 로그를 관리하며,
하위에 포스트 제목, 포스트 내용, 작성자 아이디, 그리고 작성일이
기재되도록 디자인하였습니다. 여기서 날짜 별로 로그를 관리하지 않는
이유는 당일 학습시간 인증을 기능으로 하기 때문에 글 작성을 하면 과거
작성 로그가 새로운 로그로 갱신되도록 하기 위해서 입니다.

테이블 구성 #4



`user_profile` 테이블은 사용자의 아이디 별로 로그를 관리하며,
하위에 이름, 성별, 학년, 전공, 관심사 정보를 저장하도록 설계하였습니다.

테이블 구성 #5



`user_time` 테이블은 사용자의 아이디 별로 로그를 관리하고, 하위에 날짜를 두어 날짜 별로 학습시간을 기록할 수 있게 설계하였습니다.

3-2. 데이터베이스 기능

CRUD 방법 #1

- Create

```
var user_log = firebase.database().ref('user_log/' + id + '/' + today + '/log' + beginCnt);
user_log.set
({
  begin_time : begin_time,
  end_time : end_time,
  task_name : task_name,
  study_duration : study_duration
});
```

먼저 `firebase.database().ref()`를 통해 래퍼런스(테이블)을 받아옵니다.

파이어베이스는 독특하게 디렉토리 구조를 띄고 있어 이 구조를 이용하여 원하는 테이블에 접근할 수 있게 문자열의 연결연산 등을 이용하여 제어합니다.

(' /log '와 같은 방식으로 원하는 래퍼런스에 접근할 수 있게 함)

또한 `set()` 메소드를 통해 해당 래퍼런스(테이블)에 데이터를 설정하는 처리를 합니다.

CRUD 방법 #2

- Read

```
// 조회날짜에 학습시간 얻어오기 위한 처리
var study_time = "";
var ref = firebase.database().ref('user_time/' + id + '/' + dateStr);
ref.once("value").then(function(snapshot) {
    study_time = snapshot.child("user_time").val();
    var study = document.createElement("div");
    study.innerHTML = "학습시간 : " + study_time;
    study.setAttribute("style", "font-family:'Jua', sans-serif;font-size:20px;color:navy;");
    document.getElementById("record").appendChild(study);
});
```

Firestore.database().ref()를 통해 래퍼런스(테이블)을 받아옵니다.

위의 경우 id를 래퍼런스를 받아오는 부분에 작성을 했는데 if문을 이용하여 조건에 맞는 데이터만 읽어오는 것과 동일한 효과를 내게 됩니다.

논리적 흐름은 “대상 래퍼런스 받기 → 해당 래퍼런스의 구성요소를 순회 (조건 있는 경우 만족하는 것만, 아니면 모든 것)

CRUD 방법 #3

- Update

```
var id = getParameterByName('user_id');
var user_log = firebase.database().ref('user_log/' + id + '/' + getTimestamp() + '/log' + endCnt);

user_log.update
({
  end_time : end_time,
  task_name : task_name
});
```

Firestore.database().ref()를 통해 래퍼런스(테이블)을 받아옵니다.

그 후 .update() 메소드를 통해 end_time과 task_name의 값을 처리된 값으로 갱신하는 처리를 수행하도록 하였습니다. set() 메소드와의 차이점은 set()의 경우 전체 테이블을 갱신하지만 update()는 지정된 부분만 갱신을 한다는 차이점이 있습니다.

CRUD 방법 #4

- Delete

```
/* 계정삭제 */  
function removeAccount()  
{  
    var ret = confirm("정말 탈퇴하시겠습니까?");  
    if(ret)  
    {  
        var rmv_data = firebase.database().ref('user_data/' + '<%= id %>');  
        var rmv_profile = firebase.database().ref('user_profile/' + '<%= id %>');  
        var rmv_post = firebase.database().ref('user_post/' + '<%= id %>');  
  
        rmv_data.remove();  
        rmv_post.remove();  
        rmv_profile.remove();  
        alert("탈퇴되었습니다");  
    }  
    else  
        alert("탈퇴가 취소되었습니다");  
}
```

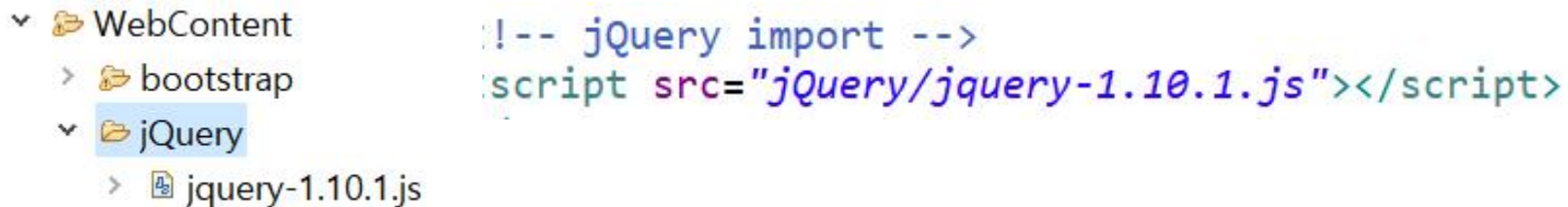
Firebase.database().ref()를 통해 래퍼런스(테이블)를 받아옵니다.

그 후 .remove() 메소드를 통해 대상 테이블을 삭제합니다.

삭제 후에는 돌이킬 수 없기 때문에 확인 다이얼로그를 통해 사용자의 실수로 인해 삭제가 되는 상황을 방지하도록 조치하였습니다.

4. 적용기술 명세

jQuery의 적용



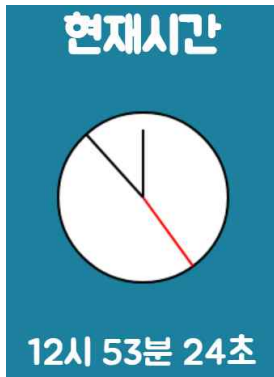
앞서 말씀 드린 부트스트랩의 경우와 마찬가지로 관리의 용이성을 위해 jQuery라는 폴더를 생성하여 관리합니다. 또한 웹에서 import하는 방법도 있지만 그럴 경우 응답속도가 늦어질 수 있는 점을 고려하여 내부에 포함하는 방법을 채택하였습니다.

```
// 입력필드에서 값을 받아옴
var id = $("#id").val(); var form = $("#login");
var pw = $("#pw").val(); form.attr("action", "./Login?id=" + id + "&&pw=" + pw + "");
```

저는 제이쿼리를 코드 작성의 효율성을 위해 사용하였습니다.

예를 들어 document.getElementById("id").value를 \$("#id").val()로 축약하거나 form.setAttribute()대신 form.attr()로 축약하는 등의 방식으로 적용하여 코드 작성의 효율성을 높였습니다.

HTML5 canvas의 적용



```
<!-- ANALOG CLOCK -->
<canvas id="myCanvas" width="300px" height="200px"></canvas>
<br>
<script>
    var canvas = document.getElementById("myCanvas"); // canvas 요소 받아오기
    var context = canvas.getContext("2d"); // context 객체 받아오기

    // 5) 시계판을 그리는 함수
    function drawCircle()
    {
        context.beginPath(); // context 초기화
        context.arc(x, y, r, 0, 2.0 * Math.PI, true); // 원을 그림
        context.fillStyle = "white"; // 색깔 채움
        context.strokeStyle = "black"; // 선 색 지정
        context.lineWidth = "2"; // 선 두께 지정
        context.closePath(); // 닫힌 도형 생성
        context.fill(); // 색칠하기 (먼저 closePath해야 함)
        context.stroke(); // 선 그리기
    }
}
```

저는 HTML5의 canvas를 record.jsp에 적용하여 위젯과 유사한 효과를 낼 수 있도록 설계하였습니다. 이 기능은 기록을 작성하면서 현재 시간을 확인하고 싶을 때 사용할 수 있도록 설계하였습니다.

HTML5 시맨틱 태그의 적용 #1

```
<!-- NAVIGATION -->
<a class="menu-toggle rounded" href="#"> <i class="fas fa-bars"></i></a>
<nav id="sidebar-wrapper">
<!--/ NAVIGATION -->

<!-- HEADER -->
<header class="masthead d-flex">
<!--/ HEADER -->

<!-- SECTION -->
<section class="content-section bg-light" id="about" style="height:800px;">
  <div class="container text-center">
    <!-- ARTICLE #2 -->
    <article id="log">
    <!--/ ARTICLE #2 -->
  </div>
</section>
<!-- SECTION -->
```

nav, header, section, article의 시맨틱 구조화 태그를 적용하여 페이지의 구조를 잡는데 반영하였습니다.

HTML5 시맨틱 태그의 적용 #2

```
<!-- DIGITAL CLOCK -->  
<time id="timer"></time>
```

time 시맨틱 태그를 사용하여 시간의 의미가 명확히 드러나게 하였습니다.

```
<!-- Background Video -->  
<div class="overlay"></div>  
<video autoplay="autoplay" muted="muted"  
  loop="loop">  
  <source src="bootstrap/main/mp4/bg.mp4" type="video/mp4">  
</video>
```

video 시맨틱 태그를 사용하여 비디오를 삽입하였습니다.

```
<meta name="description" content="study log maker">  
<meta name="author" content="YOOK DONGHYUN">
```

meta 태그를 사용하여 페이지에 대한 정보를 검색엔진에 제공하도록 하였습니다.

구현 테크닉 적용 #1

```
if (flag)
{
    flag = !flag; // flip
    document.getElementById("btn").value = "기록완료";
    insertBegin();
}
else
{
    if (document.getElementById("task").value) // 입력이 NULL인지 확인
    {
        flag = !flag; // flip
        document.getElementById("btn").value = "기록시작";
        insertEnd();
        document.getElementById("task").value = ""; // 입력박스 초기화
    } else
        alert("내용이 입력되지 않았습니다!");
}
```

학습로그

학습로그

13:03:15~

값에 따라 표시되는 형식이 달라지게 하는 테크닉을 적용하여 사용자로 하여금 버튼을 누르면 기록이 시작되는 것인지 기록이 끝나는 것인지 명확히 알 수 있도록 디자인에 반영하였습니다.

구현 테크닉 적용 #2

```
var edit = document.createElement("input"); // 수정버튼
edit.setAttribute("type", "button");
edit.setAttribute("value", "수정");
edit.setAttribute("class", "btn btn-secondary");
edit.setAttribute("id", "edit" + endCnt);
edit.addEventListener("click", editing);
// 로그 한줄에 삽입
document.getElementById("log" + endCnt).appendChild(end);
document.getElementById("log" + endCnt).appendChild(cmt);
```

자바스크립트를 이용하여 동적으로 요소를 삽입하고 이벤트리스너를 등록하거나 속성을 설정하였습니다.

```
var c = this.parentNode;
var p = c.parentNode;
p.removeChild(c);
```



또한 자바스크립트를 이용하여 동적으로 요소를 삭제하는 기능도 적용하였습니다.

구현 테크닉 적용 #3

```
var ret = confirm("정말 삭제하시겠습니까?");  
if (ret)  
{  
    var c = this.parentNode;  
    var p = c.parentNode;  
    p.removeChild(c);  
  
    var rmvId = this.id; // 삭제할 대상 노드의 id를 받아옴  
    var parse = rmvId.split("rmv"); // 파싱함  
  
    var id = getParameterByName('user_id');  
    var user_log = firebase.database().ref('user_log/' + id + '/' + getTimestamp() + '/log' + parse[1]);  
    user_log.remove();  
}  
else  
    alert("삭제되지 않았습니다");
```



확인 다이얼로그를 통해 사용자의 실수로 인한 삭제를 방지하였습니다.

구현 테크닉 적용 #4

```
/* 로그수정 */
function editing()
{
    var target = this.parentNode.getElementsByTagName("span")[2];
    task_name = prompt("변경할 업무이름을 입력해주세요");
    target.innerHTML = task_name + "&nbsp;&nbsp;&nbsp;";

    var editId = this.id;
    var parse = editId.split("edit");

    var id = getParameterByName('user_id');
    var user_log = firebase.database().ref('user_log/' + id + '/' + getTimestamp() + '/log' + parse[1]);

    user_log.update
    ({
        task_name : task_name
    });
}
```

localhost:8080 says

변경할 업무이름을 입력해주세요

OK Cancel

프롬프트 다이얼로그를 통해 사용자로부터 입력 값을 받고 해당 입력 값을 통해 데이터베이스를 업데이트하는 처리를 수행하도록 적용하였습니다.

구현 테크닉 적용 #5

```
// passing을 함
var beginParse = begin_time.split(":");
var endParse = end_time.split(":");

// Date 객체 생성하여 시간차이를 구함
var b = new Date("0", "0", "0", beginParse[0], beginParse[1], beginParse[2], "0"); // 시작시간
var e = new Date("0", "0", "0", endParse[0], endParse[1], endParse[2], "0"); // 끝 시간
var diff = (e.getTime() - b.getTime()) / 1000; //sec 단위로 시간차이 나타냄

result += diff;
var temp = result;

var sec = result % 60;
result -= sec;
result /= 60;
var min = result % 60;
var hr = result / 60;

result = temp;
```

두 시간의 차이를 계산하는 테크닉을 공부를 시작한 시간과 공부가 끝난 시간을 파악하여 이 둘의 차이(학습시간)를 구하는 처리를 구현하는데 적용하였습니다.

JSP에서 자바코드 이용 #1

12131819님 반갑습니다!

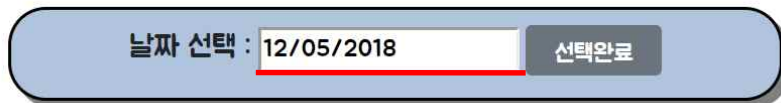
```
<% String id = (String)session.getAttribute("id"); %>  
<h3 class="juaFont"><%= id %>님 반갑습니다!</h3><hr>
```

```
var rmv_data = firebase.database().ref('user_data/' + '<%= id %>');  
var rmv_profile = firebase.database().ref('user_profile/' + '<%= id %>');  
var rmv_post = firebase.database().ref('user_post/' + '<%= id %>');
```

<% %> 문법을 이용하여 자바코드를 JSP(또는 HTML)에 사용하여
로그인 된 계정에 맞게 환영 메시지를 출력하는데 사용하였습니다.

JSP에서 자바코드 이용 #2

로그조회



날짜 선택 : 12/05/2018 선택완료

```
<%@ page import="java.util.Date" %>
<%@ page import="java.text.SimpleDateFormat" %>

<%
    Date date = new Date();
    String modifiedDate= new SimpleDateFormat("yyyy-MM-dd").format(date);
%>
<input type="date" id="date" value="<%= modifiedDate %>" class="jua">
```

<% %> 문법을 이용하여 자바코드를 JSP(또는 HTML)에 사용하여
오늘 날짜를 input의 초기 설정값으로 입력되게 하여 사용자로 하여금 조회가
편리하도록 디자인하였습니다.

페이지 간 인자전달 방법 적용

```
var form = $("#login");  
form.attr("action", "./Login?id=" + id + "&&pw=" + pw + "");  
form.submit(); // form 제출 -> Login.java로 이동 -> Servlet에 의해 처리
```

```
function writePost()  
{  
    var id = getParameterByName('user_id');  
    location.href = "writePost.jsp?user_id=" + id;  
}
```

```
<% String id = (String)session.getAttribute("id"); %>  
function goBoard()  
{  
    location.href = "board.jsp?user_id=" + '<%= id %>';  
}
```

```
String login = "profile.jsp?user_id=" + id;  
request.getRequestDispatcher(login).forward(request, response);
```

저는 위의 4가지 방법을 적용하여 페이지 간 인자 전달을 구현하였습니다.

JSON의 적용

```
edit.update  
(  
  {  
    name: _name,  
    gender: _gender,  
    major: _major,  
    grade: _grade,  
    interest: _interest  
  }  
);
```

```
var config =  
{  
  apiKey: "AIzaSyA6F49hsvuStEL_ukhsLiKUEpIxsf2iz40",  
  authDomain: "mypage-c40c6.firebaseio.com",  
  databaseURL: "https://mypage-c40c6.firebaseio.com",  
  projectId: "mypage-c40c6",  
  storageBucket: "mypage-c40c6.appspot.com",  
  messagingSenderId: "631656443065"  
};  
firebase.initializeApp(config);
```

JSON 문법에 기반한 firebase를 통해 데이터 전송을 목적으로 사용하였습니다

Thank You!