OOO 보고서

프로젝트 제목	자유게시판 블로그			작성년월일	2022-12-12	
프로젝트 팀 이름	학번	2019253080	2019253046			
더블조	성명	조상필	조정현			
	역할 기여도(%)	50%	50%			

1. 프로젝트 요약

간단한 자유게시판 웹 서비스를 제공합니다.
회원가입, 로그인, 게시판 게시글 목록, 게시글 확인, 게시글 작성 각각의 페이지를 통해
기본적인 게시판 기능을 구현했습니다.
회원가입 및 로그인 시 유효성 검사를 통해 지정된 형식에 맞는 문자나열만이 DB에 저장되도록 하였습니다. 또한 게시글의 리스트 중 클릭하여 해당 게시글의 내용 게시글view 페이지에서 확인할 수 있고, 사용자가 게시글 작성 페이지에서 게시글을 작성하여 게시글을 게시할 수 있습니다.

2. 프로젝트 내용

가. 프로젝트 제목

자유게시판 블로그

나. 프로젝트 목표

- 웹프로그래밍 수업을 통해 배운 html, css, javascript, jsp를 적용하여 블로그의 게시글 서비스를 제공하는 웹사이트를 간단히 구현해보기

다. 프로젝트 전체 구성도

- 회원가입 -> 로그인 -> 게시판목록확인 -> 게시글 작성 -> 게시글 확인

기본적으로 위 5단계를 거쳐 사용하는 게시판 블로그를 기획했습니다.

회원가입/로그인 페이지에서는 아이디와 비밀번호의 입력양식에 맞게 끔 입력하여 db에 사용자의 정보가 입력되도록 하였으며, 게시판 목록과 게시판 작성 시 각 사용자를 구분하여 작성 정보가 db에 저장되도록하며, 게시글 확인시 해당 게시물 확인이 가능하도록 db에서 그에 맞는 정보를 가져오도록 구현하도록 했습니다.

게시판 목록 페이지의 게시글 목록이 올라오는 table은 bootstrap5를 이용해 디자인을 적용시켰습니다. 또한 각 페이지의 글꼴타입은 css에 구글 폰트사이트의 폰트를 사용하기 위해 css에 import하여 font-family: 'Noto Sans KR', sans-serif; 속성을 적용했습니다.

라. 프로젝트에 사용된 주요 기술 현황

회원가입/로그인 페이지에서의 유효성 검사는 js를 이용하여 preventDefault()를 이용하여 submit을 멈추고 각 유효성 검사를 수행하는 함수를 수행하여 검사하고 이상이 없을 경우 오류가 없음을 보여주고 잠시 뒤 submit를 실행시켜서 회원가입 또는 로그인이 진행되도록 구성했습니다.

```
function isValidPw(input) {
    const pwValidation = /^.*(?=^.{8,16}$)(?=.*\d)(?=.*[a-zA-Z])(?=.*[~,!,@,#,$,*,(,),=,+,_,.,|]).
    *$/;
    if (pwValidation.test(input.value.trim())) {
        showSuccess(input);
    } else {
        showError(input, "영문, 숫자, 특수문자를 포함한 8~16글자의  비밀번호를 입력해주세요.");
    }
}
```

유효성 검사는 정규표현식을 이용하여 조건에 맞게 설정하였고 input의 value값과 비교하여 정상/오류 판별을 하도록 하였습니다.

mariadb database web, user, board 설계입니다.

primary key는 각각 id, boardID 이고 forignkey는 따로 설정하지 않았습니다.

```
MariaDB [web]> describe user;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | varchar(20)  | NO   | PRI | NULL    |       |
| password | varchar(256) | NO   |     | NULL    |       |
| name     | varchar(30)  | NO   |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.084 sec)

MariaDB [web]> describe board;
+----------------+---------------+------+-----+---------+-------+
| Field          | Type          | Null | Key | Default | Extra |
+----------------+---------------+------+-----+---------+-------+
| boardID        | int(11)       | NO   | PRI | NULL    |       |
| boardTitle     | varchar(50)   | YES  |     | NULL    |       |
| userID         | varchar(20)   | YES  |     | NULL    |       |
| boardDate      | datetime      | YES  |     | NULL    |       |
| boardContent   | varchar(2048) | YES  |     | NULL    |       |
| boardAvailable | int(11)       | YES  |     | NULL    |       |
+----------------+---------------+------+-----+---------+-------+
6 rows in set (0.017 sec)
```

write를 실행할 때 database board에 있는 datetime Type에 등록하기 위해
날짜 정보를 받기 위해 Date와 SimpleDateFormat을 import 하였습니다.

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

등록시 boardID에 primary key를 auto_increment를 db에서 자동으로 추가하지 않아
코드상에 prim= prim+1을 넣어주어 증가시켰습니다.

```
28
29      String title = request.getParameter("title");
30      String context = request.getParameter("content");
31
32      String user_id = (String) session.getAttribute("user_id");
33      long time = System.currentTimeMillis();
34      SimpleDateFormat datess = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
35      String date = datess.format(new Date(time));
36
37      int prim =(int) session.getAttribute("check");
38      prim= prim +1;
39
40      String insert_value="INSERT INTO board VALUES ('"+ prim /* auto primarykey*/+"', '"+ title +"', '"+ user_id +"','"+ date +"', '"+ context +"','1');";
41
42      stmt.executeUpdate(insert_value);
43      String redirect = "board.jsp";
44      response.sendRedirect(redirect);
45
```

반복적으로while문을 통해 table 목록들을 추가

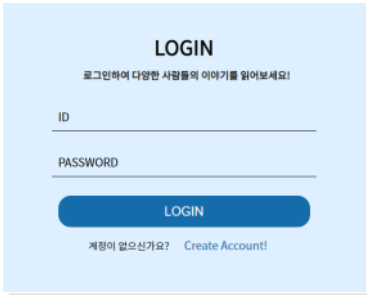
```
67        <tr>
68            <td>Title</td>
69            <td>Writer</td>
70            <td>Date</td>
71            <td>Views</td>
72        </tr>
73    </thead>
74    <%
75    while(result.next()) {
76        check++;
77        session.setAttribute("check",check);
78    %>
79    <tbody id="tablebody" class="table-light">
80        <tr onclick="trClick(this)">
81            <td><%=result.getString("boardTitle")%></td>
82            <td><%=result.getString("user.name")%></td>
83            <td><%=result.getString("boardDate")%></td>
84            <td><%=result.getString("boardAvailable") %></td>
85        </tr>
86
87    </tbody>
88    <%} %>
89  </table>
```

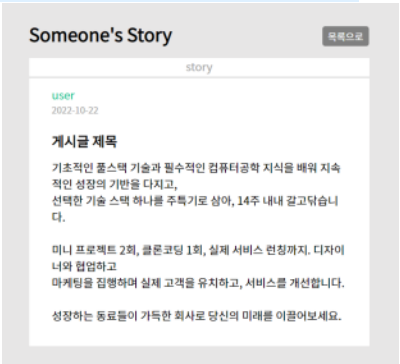
3. 프로젝트 수행결과물(개별파트)

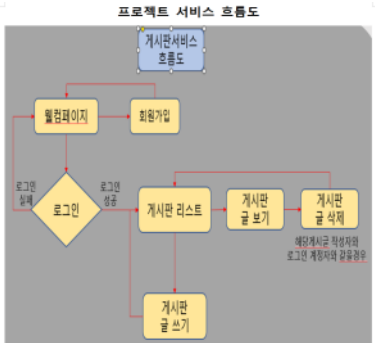
개인 담당 : 조상필 - 프론트엔드

가. 프로젝트 **개별** 결과물

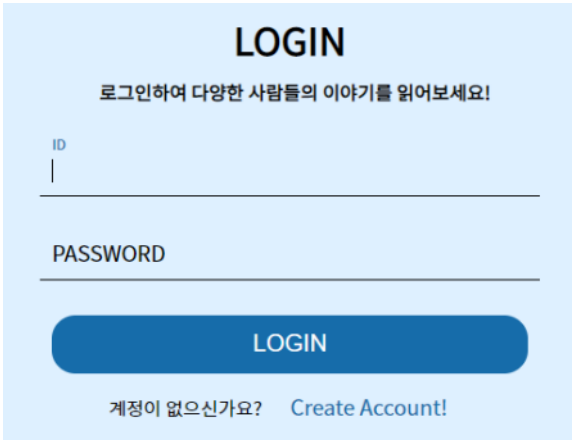
- 게시판 블로그 웹 프로젝트에서 프론트엔드를 담당하였습니다. 블로그 게시판 서비스의 기본적인 서비스 흐름도를 구상한 뒤, 로그인/회원가입/게시판목록/게시글작성/게시글확인 페이지를 퍼블리싱 하였으며 CSS, Javascript를 활용하여 각 페이지에서 사용자가 입력하거나 마우스 오버, 클릭 등 외부로 나타나는 효과를 적용시켰습니다. 또한 회원정보 등 원하는 양식에 맞게 입력할 수 있도록 필수 입력값에 대한 유효성 검사를 적용 및 페이지 이동 간 필요값들을 넘길 수 있도록 하였습니다.







나. 개발 결과물의 주요 부분 기술개발 내용



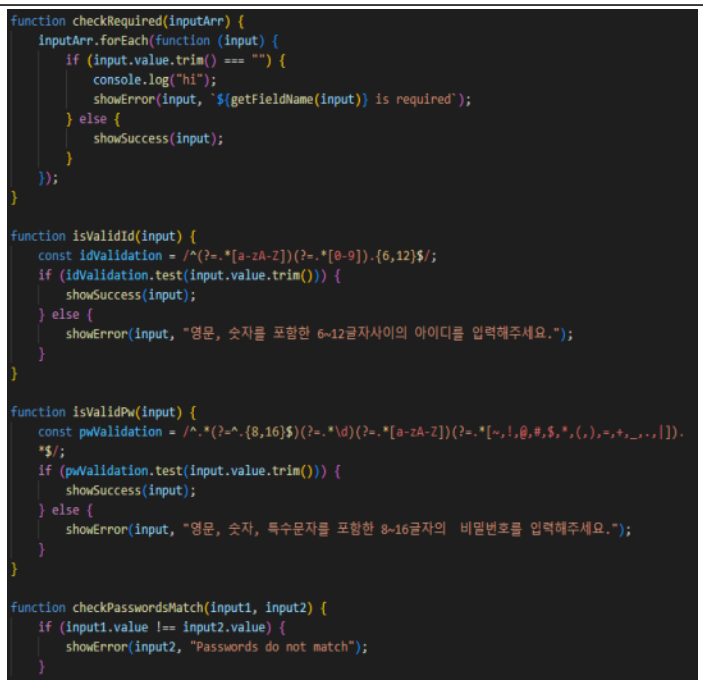
로그인 및 회원가입 시 영문,숫자,특수문자 등 입력항목에 맞게 유효성검사를 시행하고 그에 따른 결과에 맞게 사용자가 확인 후 입력할 수 있도록 적용시켰습니다.
회원가입에서 아이디는 영문과 숫자가 포함되도록, 비밀번호는 영문, 숫자, 특수문자가 포함되도록 하였습니다.

```html
<form id="signForm" action="">
    <div id="nameForm" class="formControl">
        <input id="name" type="text" name="name"
        placeholder="Name" maxlength="8">
        <small>Error message</small>
    </div>
    <div id="idForm" class="formControl">
        <input id="id" type="text" name="id" placeholder="User
        Id">
        <small>Error message</small>
    </div>
    <div id="pwForm" class="formControl">
        <input id="pw" type="password" name="pw"
        placeholder="Password">
        <small>Error message</small>
    </div>
    <div id="repwForm" class="formControl">
        <input id="pwCheck" type="password"
        placeholder="RePassword">
        <small>Error message</small>
    </div>
    <div class="signButtonBox">
        <button id="signButton" type="submit">가입하기</button>
    </div>
</form>
```

```css
.formControl {
    width: 480px;
    position: relative;
    margin-top: 20px;
}

.formControl.success input {
    border-color: #2ecc71;
}

.formControl.error input {
    border-color: #e74c3c;
}

.formControl small {
    color: #e74c3c;
    position: absolute;
    top: 1;
    left: 0;
    visibility: hidden;
}

.formControl.error small {
    visibility: visible;
}
```

회원가입 페이지의 입력파트인 각 input태그에 id를 주고, 오류메시지는 small태그를 이용하였으며 css로 visibility 속성을 이용해 입력양식에 맞지않는다면 input에 error클래스를 추가시켜 error클래스가 존재할 시 small태그(오류메시지)가 화면에 표출될 수 있도록 하였습니다.

```
function showError(input, message) {
    const formControl = input.parentElement;
    formControl.className = "formControl error";
    const small = formControl.querySelector("small");
    small.innerText = message;
}

function showSuccess(input) {
    const formControl = input.parentElement;
    formControl.className = "formControl success";
}
```



```
function checkRequired(inputArr) {
    inputArr.forEach(function (input) {
        if (input.value.trim() === "") {
            console.log("hi");
            showError(input, `${getFieldName(input)} is required`);
        } else {
            showSuccess(input);
        }
    });
}

function isValidId(input) {
    const idValidation = /^(?=.*[a-zA-Z])(?=.*[0-9]).{6,12}$/;
    if (idValidation.test(input.value.trim())) {
        showSuccess(input);
    } else {
        showError(input, "영문, 숫자를 포함한 6~12글자사이의 아이디를 입력해주세요.");
    }
}

function isValidPw(input) {
    const pwValidation = /^.*(?=^.{8,16}$)(?=.*\d)(?=.*[a-zA-Z])(?=.*[~,!,@,#,$,*,(,),=,+,_,.,|]).
    *$/;
    if (pwValidation.test(input.value.trim())) {
        showSuccess(input);
    } else {
        showError(input, "영문, 숫자, 특수문자를 포함한 8~16글자의  비밀번호를 입력해주세요.");
    }
}

function checkPasswordsMatch(input1, input2) {
    if (input1.value !== input2.value) {
        showError(input2, "Passwords do not match");
    }
```

각 input의 입력된 값의 유효성 검사와 오류 또는 정상일시 클래스를 추가시켜 오류메시지를 출력할지 form전송을 수행할지 결정짓도록 하는 것은 js를 이용하여 완성시켰습니다.
form의 submit을 preventDefault()를 이용하여 막은 뒤, checkRequired() 함수에서 input의 입력값이 비어있는지 체크하고, isValidId()함수와 isValidPw()함수, checkPasswordsMatch()함수로 id와 패스워드의 유효성 검사를 수행하고 비밀번호 확인을 합니다.

오류가 있을 경우에는 error클래스를 추가하고 그에 맞는 에러메시지를 넣어줍니다. 오류가 없다면 success클래스를 추가합니다. 모든 input의 클래스가 success일 경우에는 모든 입력항목이 정상임을 보여주고 잠시 후 입력한 내용대로 회원가입을 진행할지 묻는 질문창이 나오도록 하였습니다.

개인 담당 : 조정현 - 백엔드

가. 프로젝트 **개별** 결과물

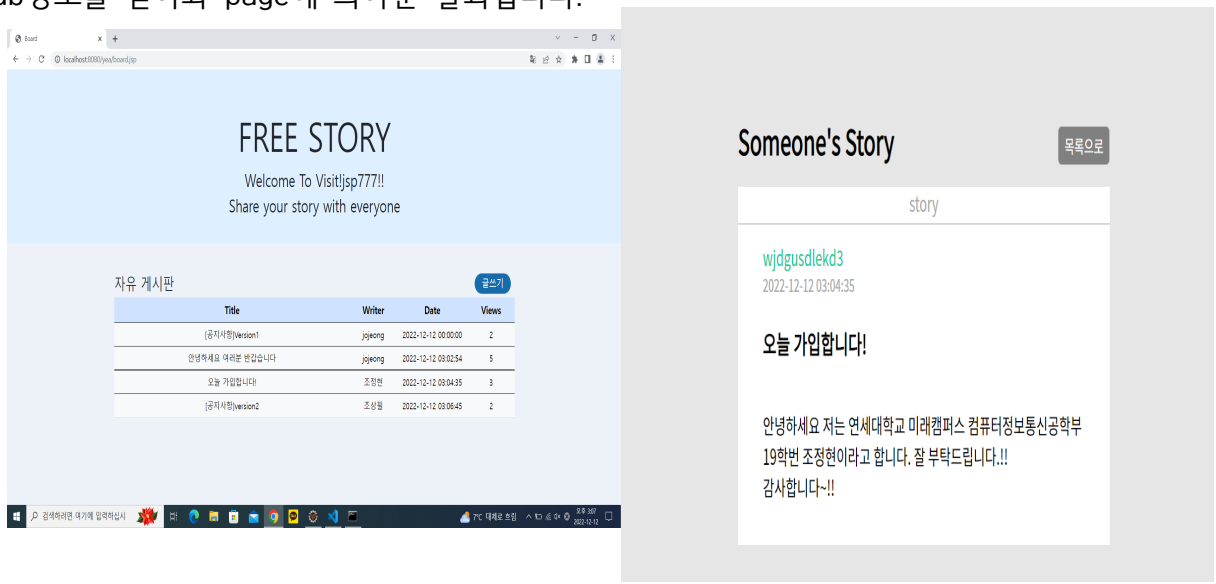
- 게시판 블로그 웹 프로젝트에서 백엔드를 담당하였습니다. 블로그 게시판 서비스의 기본적인 서비스 흐름도에 맞추어, login_result.jsp/signup_result.jsp/board.jsp/write_result.jsp/view.jsp에서 서버와 연동하여 session관리와 db에 연동하는 작업을 진행하였습니다.

jsp를 활용하여 각 페이지에서 사용자가 입력하거나 마우스 오버, 클릭 등 외부로 나타나는 값을 브라우저에서 받아오거나 session으로 넘겨주어 적용시켰습니다. database table을 설계하고 그 값을 insert하거나 update를 해주었습니다.
database user table과 board table을 설계하였으며 두 테이블의 join을 통해 board 화면에 정보들을 띄어주었습니다.

두 테이블을 join 한 결과입니다. (content 내용은 join시 table형태가 깨져서 따로 보여주진 않았습니다. boar table에는 등록되어있습니다.)

```
MariaDB [web]> select S.id, S.name, T.boardID, T.boardTitle, T.boardDate, T.boardAvailable from user S, board T where S.id=T.userID;
+------------+--------+---------+-----------------------------------+---------------------+----------------+
| id         | name   | boardID | boardTitle                        | boardDate           | boardAvailable |
+------------+--------+---------+-----------------------------------+---------------------+----------------+
| cisxo22    | jojeong |       1 | [공지사항]Version1                | 2022-12-12 00:00:00 |              2 |
| cisxo22    | jojeong |       2 | 안녕하세요 여러분 반갑습니다     | 2022-12-12 03:02:54 |              5 |
| wjdgusdlekd3 | 조정현 |       3 | 오늘 가입합니다!                  | 2022-12-12 03:04:35 |              3 |
| jsp777     | 조상필 |       4 | [공지사항]version2                | 2022-12-12 03:06:45 |              2 |
+------------+--------+---------+-----------------------------------+---------------------+----------------+
4 rows in set (0.000 sec)
```

db정보를 받아와 page에 띄어준 결과입니다.



views는 게시판에 들어갈 때마다 1씩 증가합니다.

나. 개발 결과물의 주요 부분 기술개발 내용

*db 설계 내용 user table과 board table입니다.

```
MariaDB [web]> describe user;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | varchar(20)  | NO   | PRI | NULL    |       |
| password | varchar(256) | NO   |     | NULL    |       |
| name     | varchar(30)  | NO   |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.084 sec)

MariaDB [web]> describe board;
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| boardID       | int(11)       | NO   | PRI | NULL    |       |
| boardTitle    | varchar(50)   | YES  |     | NULL    |       |
| userID        | varchar(20)   | YES  |     | NULL    |       |
| boardDate     | datetime      | YES  |     | NULL    |       |
| boardContent  | varchar(2048) | YES  |     | NULL    |       |
| boardAvailable | int(11)      | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
6 rows in set (0.017 sec)
```

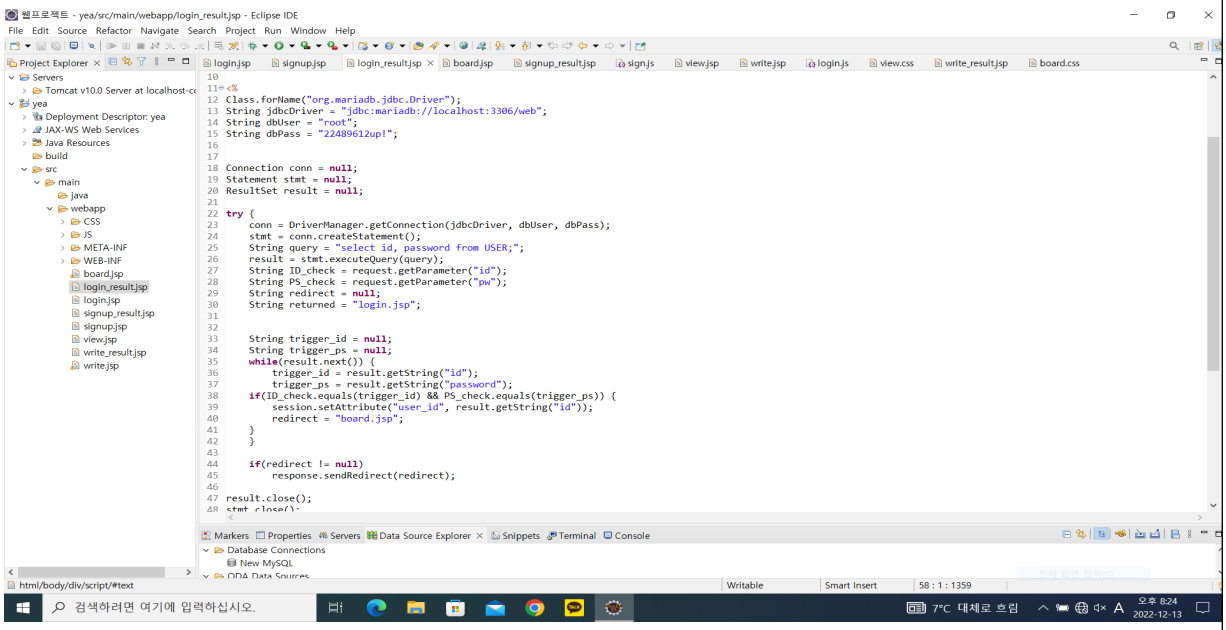
*회원가입시 받아온 정보들을 insert(signup_result.jsp)

```
    result = stmt.executeQuery(query);

    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String name = request.getParameter("name");

    String insert_value="INSERT INTO user(id, password, name) VALUES ('"+ id +"', '"+ pw +"', '"+ name +"');";

    stmt.executeUpdate(insert_value);
    String redirect = "login.jsp";
    response.sendRedirect(redirect);
```

page에서 받아온 id pw name을 database에 insert 해 준 후 response를 통해서 login.jsp page로 이동합니다.

*로그인시 db정보와 확인 (ID_check.equals(trigger_id) && PS_check.equals(trigger_ps))

login_result.jsp



```
try {
    conn = DriverManager.getConnection(jdbcDriver, dbUser, dbPass);
    stmt = conn.createStatement();
    String query = "select id, password from USER;";
    result = stmt.executeQuery(query);
    String ID_check = request.getParameter("id");
    String PS_check = request.getParameter("pw");
    String redirect = null;
    String returned = "login.jsp";


    String trigger_id = null;
    String trigger_ps = null;
    while(result.next()) {
        trigger_id = result.getString("id");
        trigger_ps = result.getString("password");
    if(ID_check.equals(trigger_id) && PS_check.equals(trigger_ps)) {
        session.setAttribute("user_id", result.getString("id"));
        redirect = "board.jsp";
    }
    }

    if(redirect != null)
        response.sendRedirect(redirect);

result.close();
```

if(ID_check.equals(trigger_id) && PS_check.equals(trigger_ps))을 통해 login.jsp page에서 입력된 값들과 signup.jsp를 통해 insert된 db정보와 비교합니다. equals를 쓰는 이유는 db에 insert된 값이 다른 type의 형태이기 때문에 그냥 String끼리 비교할수 없습니다.

만약 db목록과 같은값을 찾게될 경우login에 성공하였기 때문에 session에 성공한 user_id를 세션에 남겨주고 board.jsp page로 response 시켜 줍니다.

*게시판에 db에 저장된 board table 띄어주기(board.jsp)

```
try {
    conn = DriverManager.getConnection(jdbcDriver, dbUser, dbPass);
    stmt = conn.createStatement();
    String search = request.getParameter("search");
    String query = "SELECT * FROM user,board where user.id = board.userID";
    result = stmt.executeQuery(query);

    String user_id = (String) session.getAttribute("user_id");
    int check=0;
%>

<body style="background-color: #edf2f9 !important;">
    <div class="headerColor"></div>
    <div class="boardSection">
        <h1 class="title">FREE STORY</h1>
        <h4 class="titleScript">
            Welcome To Visit!<%=user_id%>!!
        </h4>
        <h4 class="titleScript">
            Share your story with everyone
        </h4>
```

select * from user,board where user.id= board.userID join한 table을 불러옵니다.

```
                    <tr>
                        <td>Title</td>
                        <td>Writer</td>
                        <td>Date</td>
                        <td>Views</td>
                    </tr>
                </thead>
                <%
                while(result.next()) {
                    check++;
                    session.setAttribute("check",check);
                %>
                <tbody id="tablebody" class="table-light">
                    <tr onclick="trClick(this)">
                        <td><%=result.getString("boardTitle")%></td>
                        <td><%=result.getString("user.name")%></td>
                        <td><%=result.getString("boardDate")%></td>
                        <td><%=result.getString("boardAvailable") %></td>
                    </tr>

                </tbody>
                <%} %>
```

table을 순서대로 받아오며 값을 넣어줍니다. session에 db에서 읽어온 tuple 개수 만큼 check를 세션에 납겨줍니다.

*게시판 write 시 insert (write_result.jsp)

```
    String title = request.getParameter("title");
    String context = request.getParameter("content");

    String user_id = (String) session.getAttribute("user_id");
    long time = System.currentTimeMillis();
    SimpleDateFormat datess = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
    String date = datess.format(new Date(time));

    int prim =(int) session.getAttribute("check");
    prim= prim +1;

    String insert_value="INSERT INTO board VALUES ('"+ prim /* auto primarykey*/+"', '"+ title +"', '"+ user_id +"','"+ date +"', '"+ context +"','1');";

    stmt.executeUpdate(insert_value);
    String redirect = "board.jsp";
    response.sendRedirect(redirect);
```

session에 남긴 user_id를 string user_id로 저장해 줍니다.
session에 있는 check 값을 가져와 prim에 넣어주고 db에 session에서 받아온 정보을 page에서 입력받은 값들과 함께 insert 시켜준 후 board.jsp로 response 시킵니다.

*게시판 확인시 조회수 update(view.jsp)

```
    result = stmt.executeQuery(query);
    int check = Integer.parseInt(request.getParameter("indexNum"));
%>

    <div class="viewSection">
        <div class="titleBox">
            <h1 class="pageTitle">Someone's Story
                <a href="board.jsp" class="linkButton">목록으로</a>
            </h1>
        </div>
        <%
    while(result.next()) {
        if(check == result.getInt("boardID")){
        String update_Available = "UPDATE board SET boardAvailable = boardAvailable + 1 WHERE BoardID = '"+ check +"'";
            //UPDATE board SET boardAvailable = boardavailable+1 WHERE BoardID=check

        stmt.executeUpdate(update_Available);
    %>
        <div class="pageBody">
            <div class="setLine">story</div>
            <div class="setBox">
                <div class="storyHeader">
                    <div class="header">
                        <span class="user"><%=result.getString("userID")%></span>
                        <div class="date"><%=result.getString("boardDate")%></div>
                    </div>
                </div>
                <div class="storyBody">
                    <h2 class="storyTitle"><%=result.getString("boardTitle")%></h2>
                    <div class="story">
                    <%=result.getString("boardContent")%>
```

위의 check는 session 에 있는 check가 아닙니다. 클릭이벤트로 받아온 순서대로 boardID이고 boardAvailable은 조회수 테이블로 사용하였습니다.
boardAcailable 테이블에 있는 조회수에 들어올 때 마다 1씩 증가시켜 update 시켜줍니다.

4. 문제점 검토 및 향후 개선방향

- 현 게시판 프로젝트의 결과물은 단순히 회원가입, 로그인, 게시글 확인, 게시글 작성 기능만 있습니다. 로그인 화면에서 사용자가 아이디나 비밀번호를 잊어버렸을 경우 찾을 수 있는 기능이 없기 때문에 사용자는 이전에 만들었던 계정을 찾을 수 있는 방법이 없습니다. id/pw 찾기 기능을 추가시켜 이를 개선할 수 있을 듯 보입니다.
또한 현 게시글은 작성만 가능할 뿐 수정or삭제는 불가능합니다. 게시글 작성 뿐 아니라 작성한 게시글에 대한 수정/삭제 기능을 추가시켜 유동적으로 게시글을 운용할 수 있도록 보완이 필요해 보입니다. 또한 게시글 목록을 확인하는 페이지에서 게시글을 모두 표출하기 때문에 게시글의 양이 많아진다면 기능적으로도, UI적으로도 좋지않습니다. 페이지네이션 기능을 이용하여 게시글을 가져오는 수를 조정하여 이를 개선해야 할 듯 보입니다.
주제별로 게시판이 나뉘어 질수 있도록 조정하는 것이 좋아보입니다.
회원 탈퇴시 casecade를 통해 유저 정보 삭제시 board 게시판 테이블에 user 정보 모두 연쇄삭제 되도록 db설계를 진행해야 합니다.