# Checkkeu.

오로라
20190999 이지영
20191022 진고은
20190973 신수정
20191025 최주리

청년에게 꼭 필요한

사용자 맞춤 청년정책검책 웹사이트

# Checkkeu

# 시스템개요
## : 문제의식

1

### 청년정책을 찾기가 어려움

인터넷에 청년정책에 대해 검색을 하면 너무 많은 정책들이 나와서 어떤 정책이 자신에게 도움이 되는지 찾기가 어려운 경우가 많아서, 실질적으로 정책을 활용하지 못하는 경우가 많다.

2

### 청년들의 관심 부재

많은 사람들이 관심을 가지지 않아서 어떤 청년정책이 있는 지 모르는 경우가 많다. 하지만 많은 청년정책이 존재한다. 이러한 정책들을 한번에 모아서 보여줄 수 있는 사이트가 있으면 좋겠다고 생각했다.

3

### 정책에 대한 소통 부족

국가에서도 어떤 정책이 청년들에게 필요한지 모르는 경우가 많다. 청년정책을 활용하는 청년들이 이러한 문제에 대해 토론하고 자신이 필요한 정책에 대해 의견을 제시해야 한다.

# 시스템개요
## : 기대효과

1 청년정책에 대한 관심도모

청년정책들만 모아두었기 때문에 청년들이 자신에게 해당되는 정책들을 많이 찾을 수 있다.

2 청년들 간의 소통 강화

정책제안 커뮤니티를 이용하여 원하는 정책 또는 개선되어야 할 정책에 대해 글 을 쓸 수 있으며, 댓글을 통해 소통이 가능하다. 또한 동의하기/비동의하기 버튼을 클릭하여 자신의 의견을 표현할 수 있다.

3 청년정책 검색의 편리성

사용자에게 해당되는 정보를 입력하면 사용자에게 해당되는 정책을 필터링하여 사용자에게 보여준다

4 청년정책 일정 확인

자신이 스크랩한 정책의 일정을 캘린더에 표시하여 확인할 수 있게 해준다.

# What We Do?

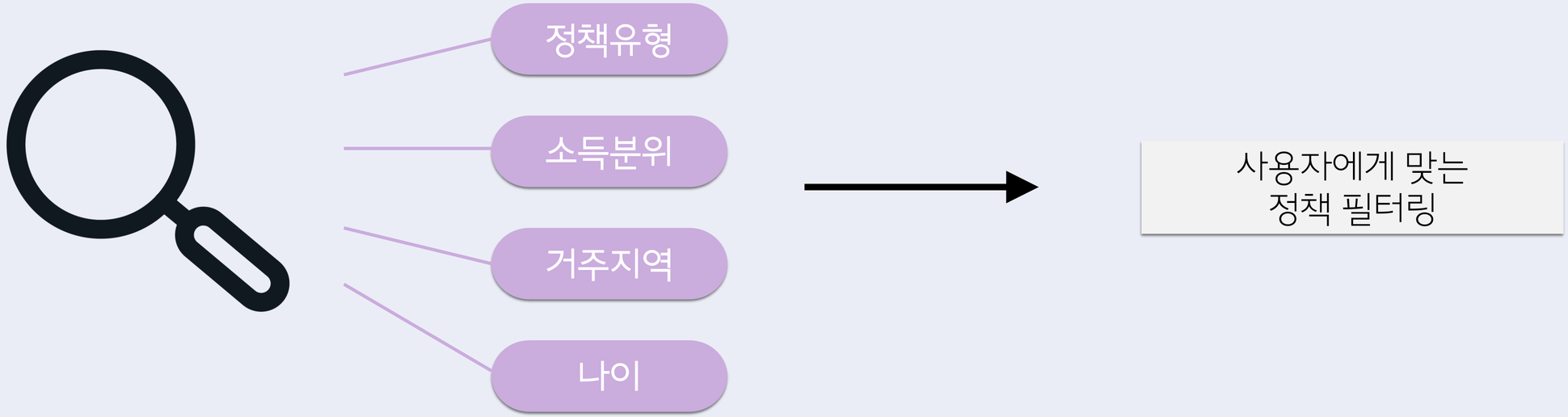
주요기능 및 특징

정책 필터링

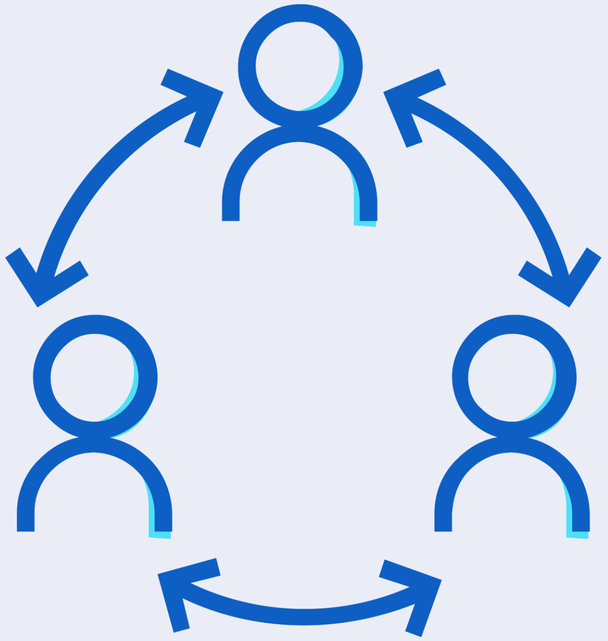
정책제안 커뮤니티

정책일정 확인

# 청년정책 필터링



# 정책제안 커뮤니티



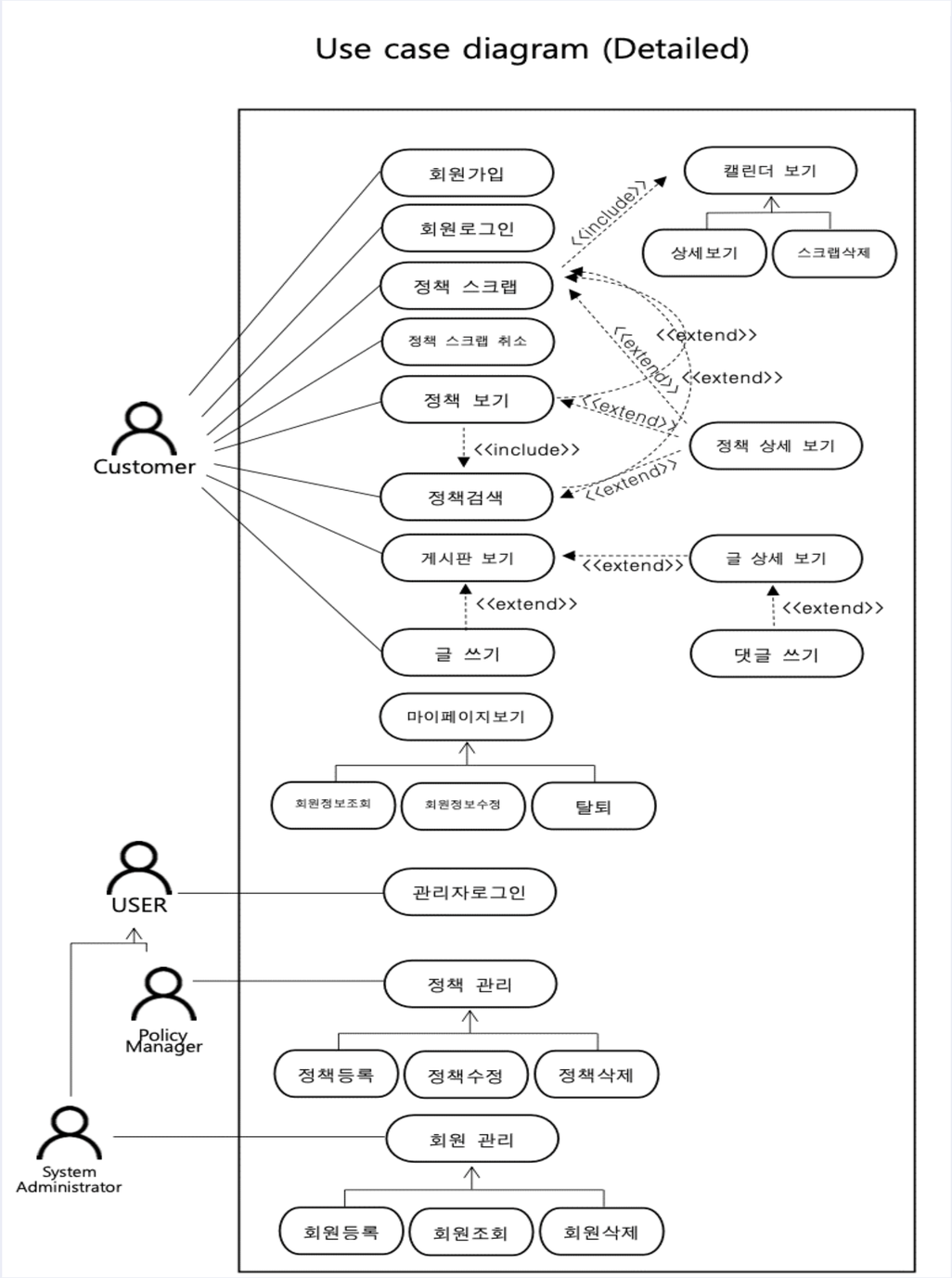
1. 정책제안 글쓰기
2. 댓글로 의견 작성
3. 동의/비동의 기능

# 정책 스크랩/일정확인

관심있는 정책 스크랩 기능

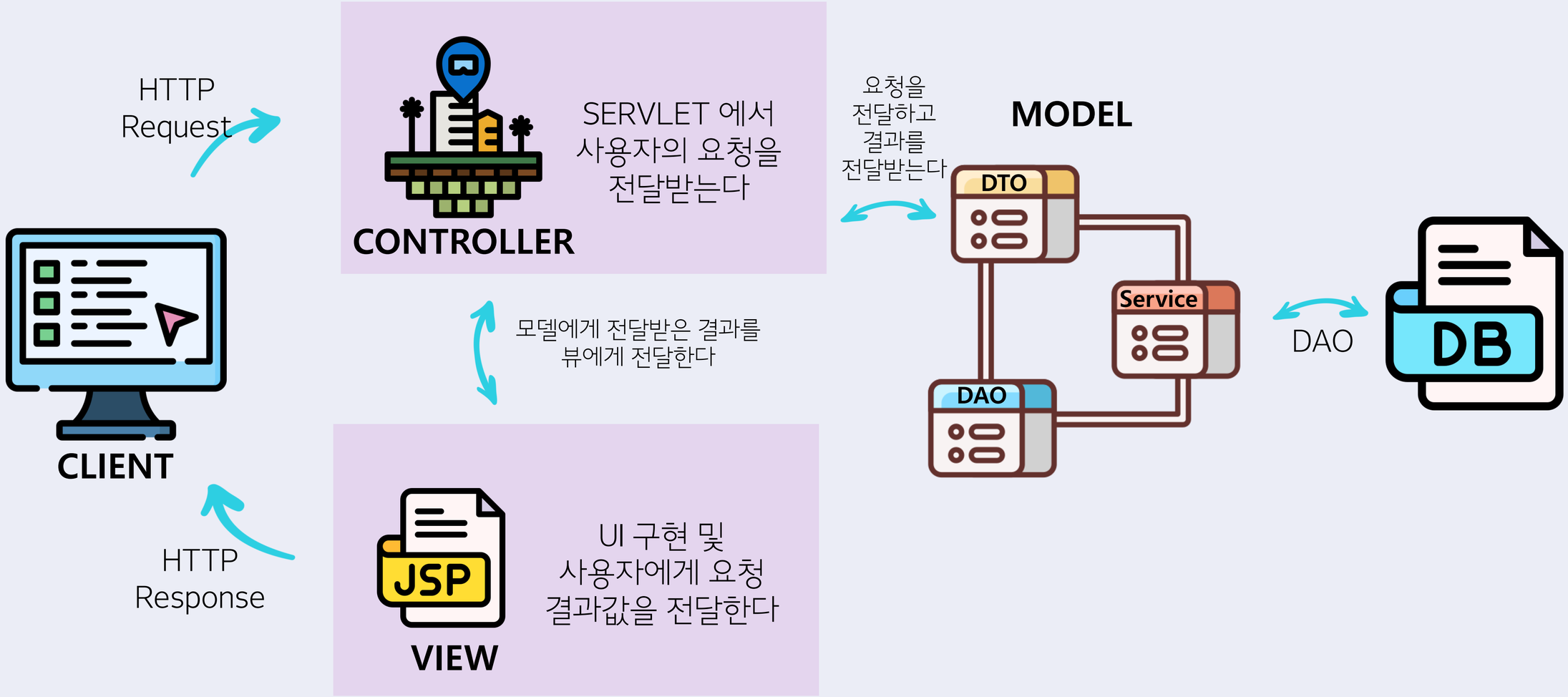
스크랩한 정책의 일정을 캘린더에 표시

# 요구사항 분석

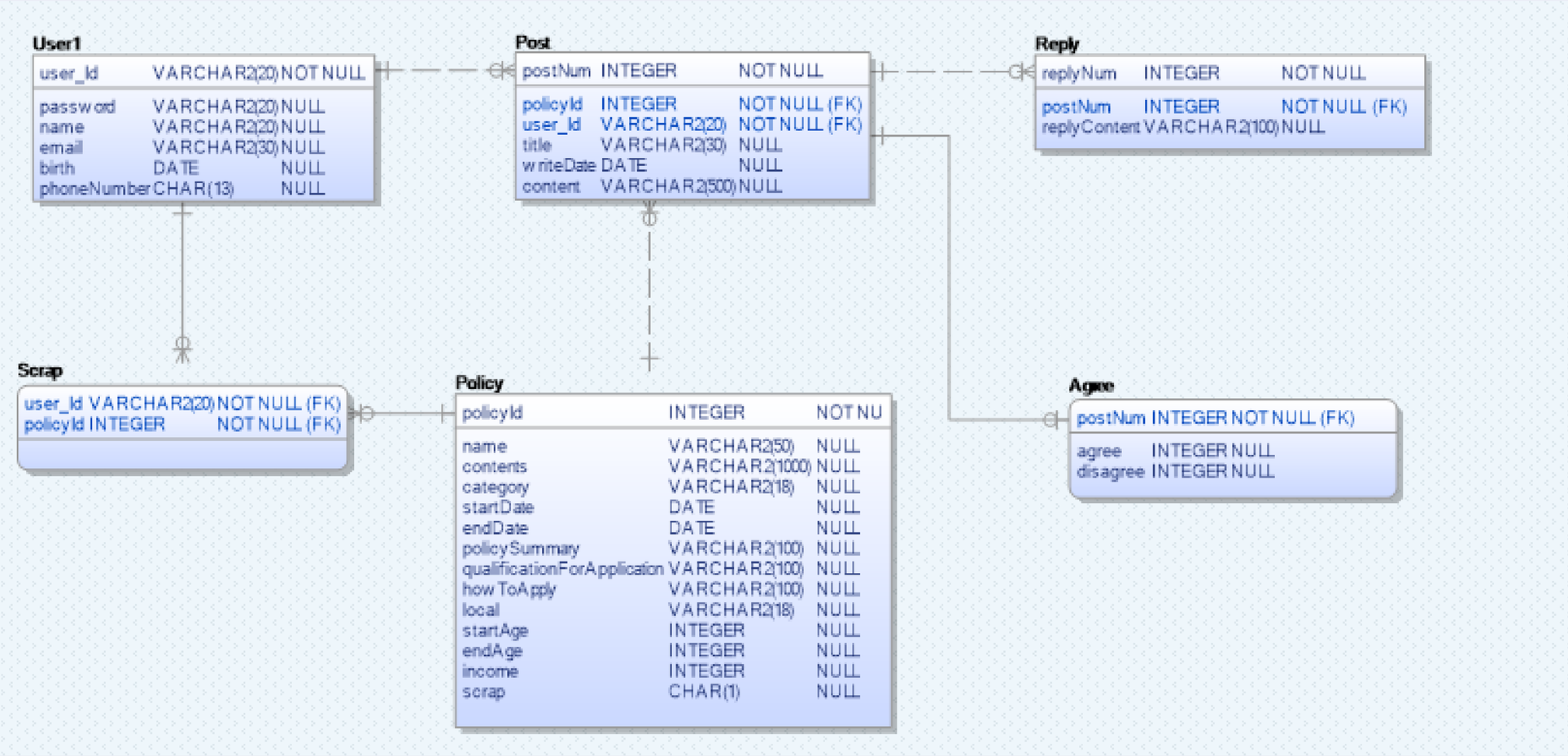

Use case diagram (Detailed)
회원가입
회원로그인
정책 스크랩
정책 스크랩 취소
정책 보기
정책검색
게시판 보기
글 쓰기
마이페이지보기
회원정보조회
회원정보수정
탈퇴
캘린더 보기
상세보기
스크랩삭제
정책 상세 보기
글 상세 보기
댓글 쓰기
관리자로그인
정책 관리
정책등록
정책수정
정책삭제
회원 관리
회원등록
회원조회
회원삭제
Customer
USER
Policy Manager
System Administrator
<<include>>
<<extend>>

# 시스템 설계

# MVC 모델흐름도

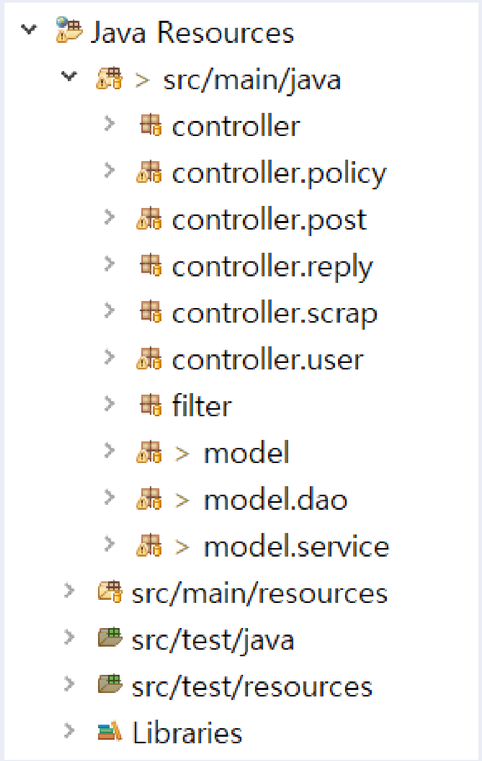

HTTP
Request
CONTROLLER
SERVLET 에서
사용자의 요청을
전달받는다
요청을
전달하고
결과를
전달받는다
MODEL
DTO
Service
DAO
DAO
DB
CLIENT
모델에게 전달받은 결과를
뷰에게 전달한다
HTTP
Response
JSP
VIEW
UI 구현 및
사용자에게 요청
결과값을 전달한다

# 데이터베이스 설계

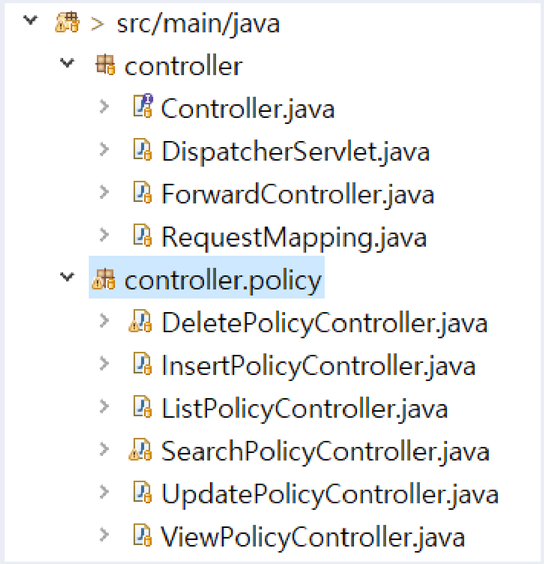

User1
user_Id VARCHAR2(20) NOT NULL
password VARCHAR2(20) NULL
name VARCHAR2(20) NULL
email VARCHAR2(30) NULL
birth DATE NULL
phoneNumber CHAR(13) NULL
Post
postNum INTEGER NOT NULL
policyId INTEGER NOT NULL (FK)
user_Id VARCHAR2(20) NOT NULL (FK)
title VARCHAR2(30) NULL
writeDate DATE NULL
content VARCHAR2(500) NULL
Reply
replyNum INTEGER NOT NULL
postNum INTEGER NOT NULL (FK)
replyContent VARCHAR2(100) NULL
Scrap
user_Id VARCHAR2(20) NOT NULL (FK)
policyId INTEGER NOT NULL (FK)
Policy
policyId INTEGER NOT NU
name VARCHAR2(50) NULL
contents VARCHAR2(1000) NULL
category VARCHAR2(18) NULL
startDate DATE NULL
endDate DATE NULL
policySummary VARCHAR2(100) NULL
qualificationForApplication VARCHAR2(100) NULL
howToApply VARCHAR2(100) NULL
local VARCHAR2(18) NULL
startAge INTEGER NULL
endAge INTEGER NULL
income INTEGER NULL
scrap CHAR(1) NULL
Agree
postNum INTEGER NOT NULL (FK)
agree INTEGER NULL
disagree INTEGER NULL

# 클래스 설계

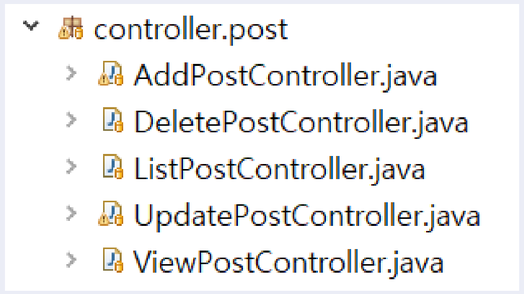
Java Package List



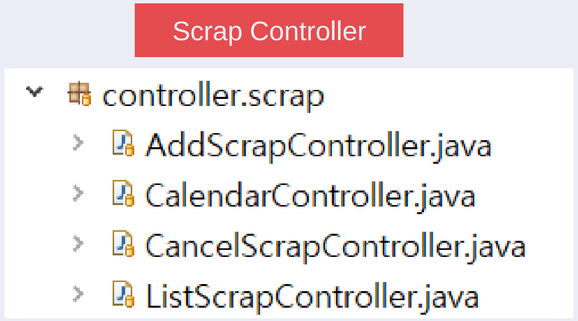

Policy Controller



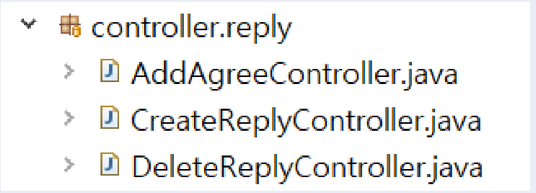

Post Controller



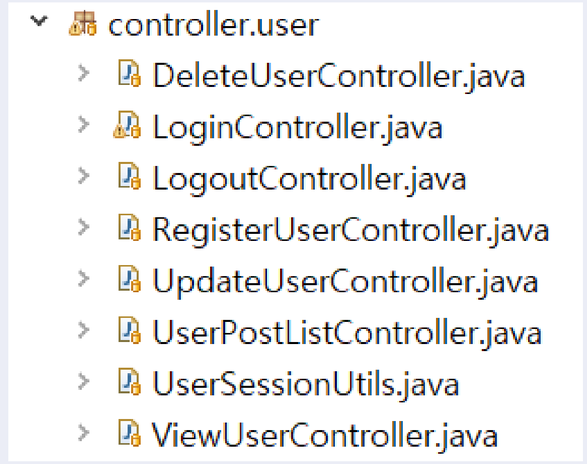

Scrap Controller




Reply Controller

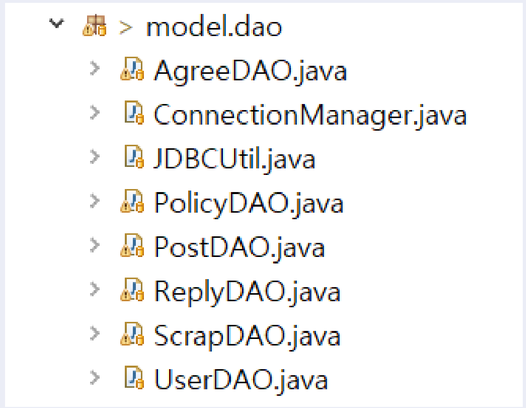



User Controller

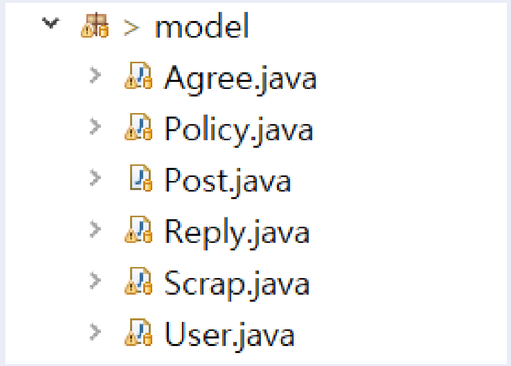


# 클래스 설계

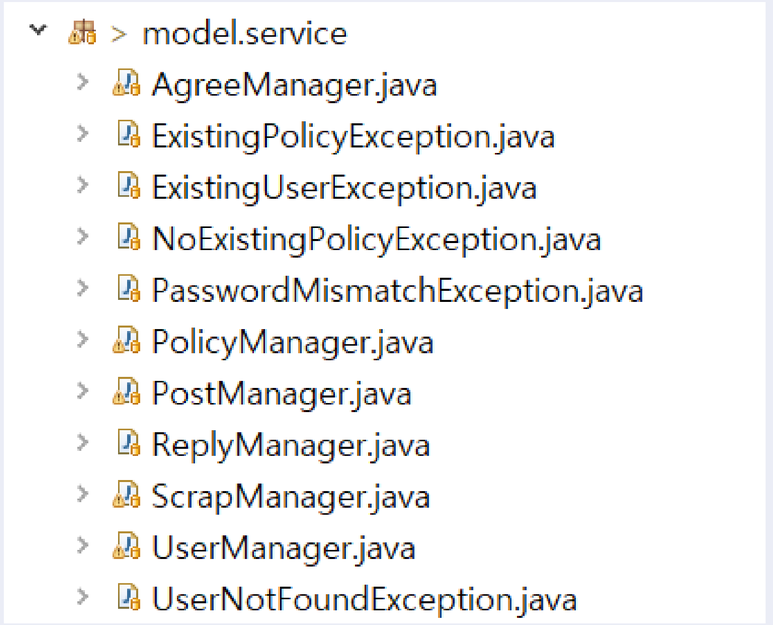
DAO




DTO

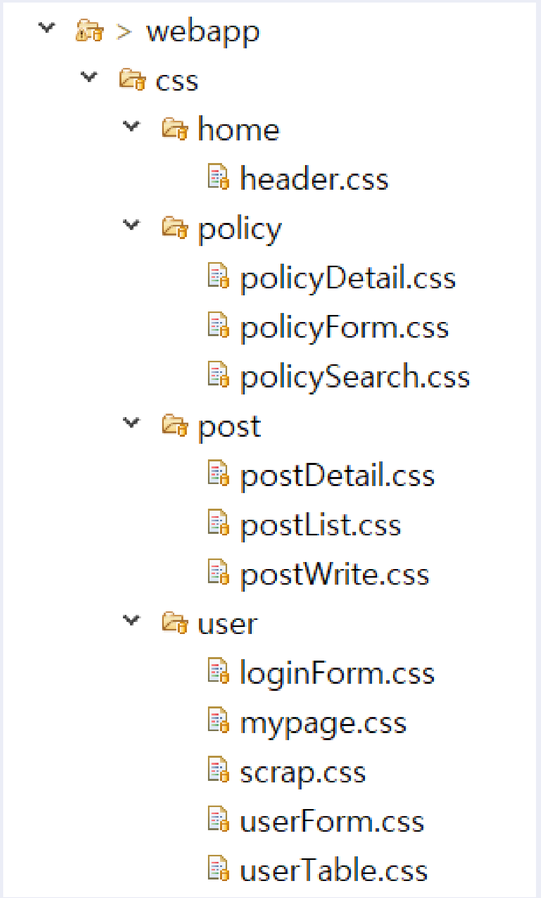



Error처리 Class
Manager Class

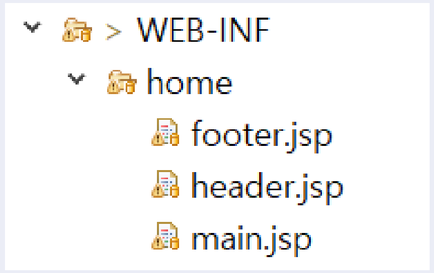


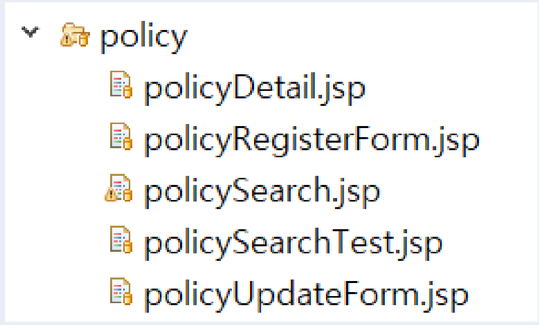
# 클래스 설계

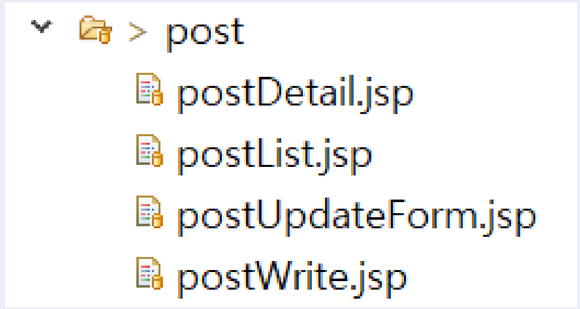
CSS

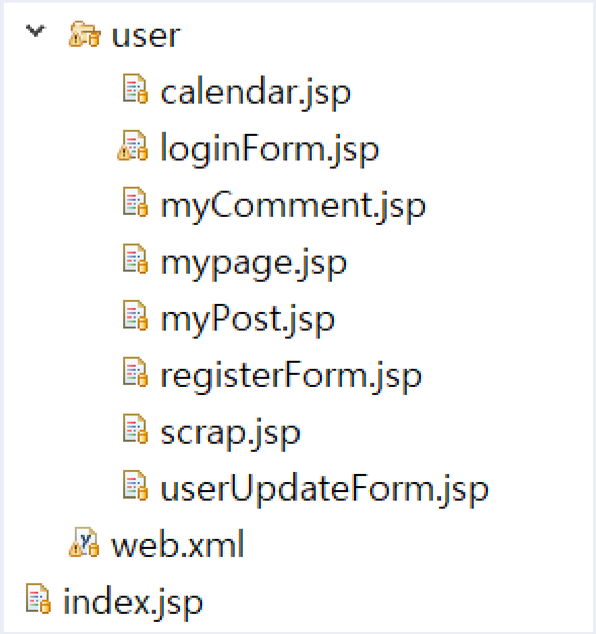



JSP Page



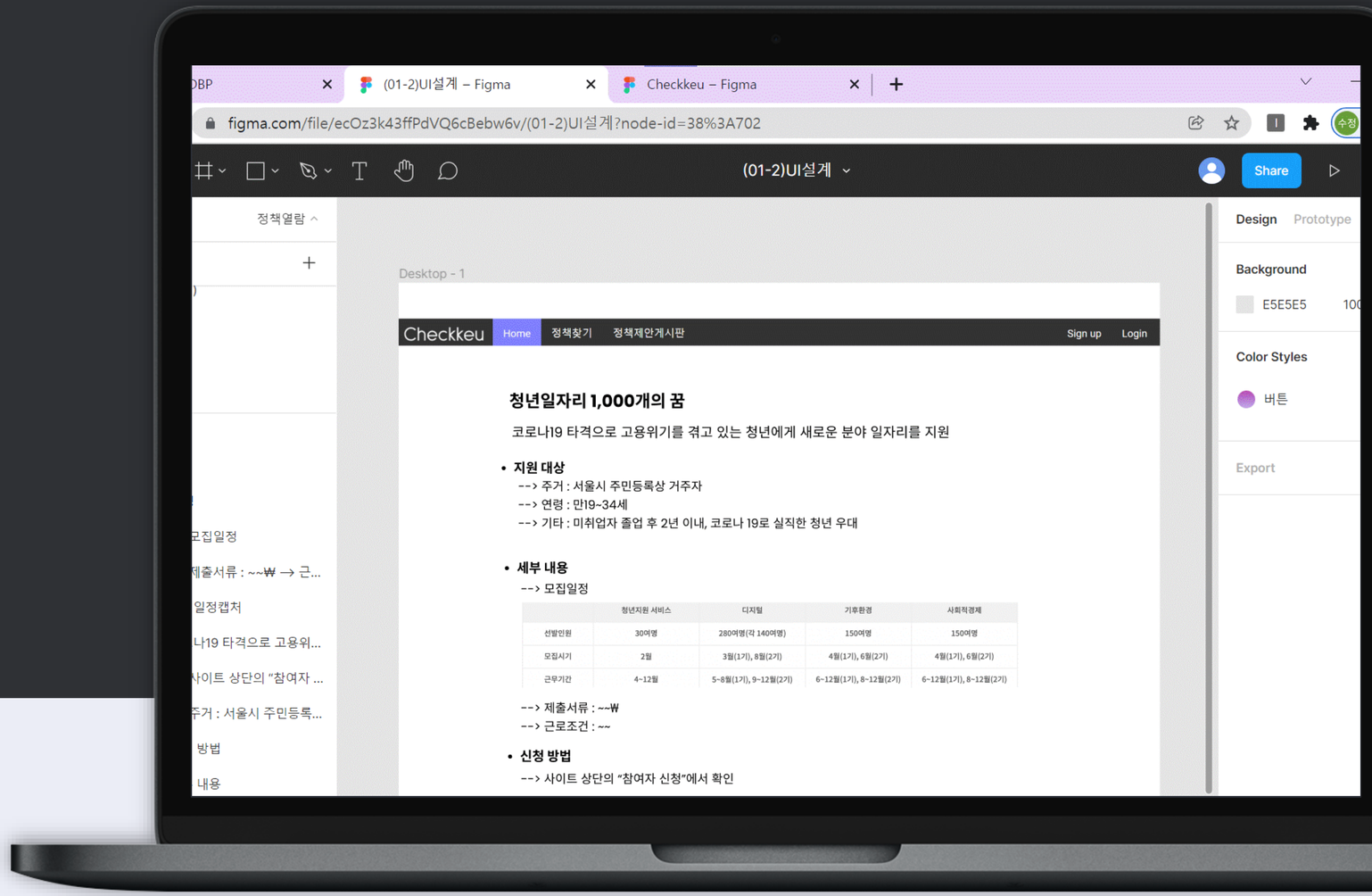



# UI 설계

Figma


(01-2)UI설계 – Figma
Checkkeu – Figma
figma.com/file/ecOz3k43ffPdVQ6cBebw6v/(01-2)UI설계?node-id=38%3A702
(01-2)UI설계
Share
정책열람
Desktop - 1
Checkkeu
Home
정책찾기
정책제안게시판
Sign up
Login
청년일자리 1,000개의 꿈
코로나19 타격으로 고용위기를 겪고 있는 청년에게 새로운 분야 일자리를 지원
지원 대상
--> 주거 : 서울시 주민등록상 거주자
--> 연령 : 만19~34세
--> 기타 : 미취업자 졸업 후 2년 이내, 코로나 19로 실직한 청년 우대
세부 내용
--> 모집일정
--> 제출서류 : ~~₩
--> 근로조건 : ~~
신청 방법
--> 사이트 상단의 "참여자 신청"에서 확인
Design
Prototype
Background
E5E5E5
Color Styles
버튼
Export

Checkkeu | Home | 정책찾기 | 정책제안게시판 | Sign up | Login

## 청년일자리 1,000개의 꿈

코로나19 타격으로 고용위기를 겪고 있는 청년에게 새로운 분야 일자리를 지원

- **지원 대상**
  --> 주거 : 서울시 주민등록상 거주자
  --> 연령 : 만19~34세
  --> 기타 : 미취업자 졸업 후 2년 이내, 코로나 19로 실직한 청년 우대

- **세부 내용**
  --> 모집일정

| | 청년지원 서비스 | 디지털 | 기후환경 | |
|---|---|---|---|---|
| 선발인원 | 30여명 | 280여명(각 140여명) | 150여명 | |
| 모집시기 | 2월 | 3월(1기), 8월(2기) | 4월(1기), 6월(2기) | 4월 |
| 근무기간 | 4~12월 | 5~8월(1기), 9~12월(2기) | 6~12월(1기), 8~12월(2기) | 6~12월 |

  --> 제출서류 : ~~₩
  --> 근로조건 : ~~

- **신청 방법**
  --> 사이트 상단의 "참여자 신청"에서 확인

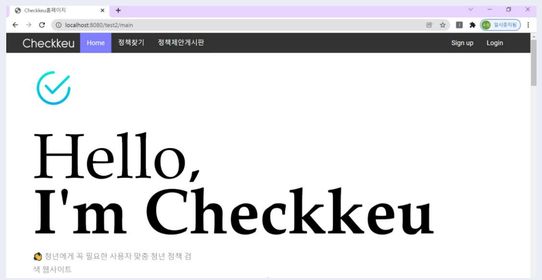
Checkkeu | Home | 정책찾기 | 정책제안게시판 | Sign up | Login

| | |
|---|---|
| 유형 | ☐ 취업지원 ☐ 창업지원 ☐ 주거·금융 ☐ 생활·복지 ☐ 정책참여 |
| 소득분위 | 분위 |
| 거주지역 | ☐ 서울 ☐ 부산 ☐ 대구 ☐ 인천 ☐ 광주 ☐ 대전 ☐ 울산 ☐ 경기 ☐ 강원 ☐ 충북 ☐ 충남 ☐ 전북 ☐ 전남 ☐ 경북 ☐ 경남 ☐ 제주 ☐ 세종 |
| 나이 | 나이 |

| 정책 | 유형 | 설명 |
|---|---|---|
| 국민취업지원제도 | 취업지원 | 한국형 실업부조로 고용안전망 사각지대에 있는 취업취약계층에게 취업지원서비스 및 생활안정을 지원하는 정책 |
| | | |
| | | |
| | | |

# REQUEST 흐름도
## : 회원가입, 로그인

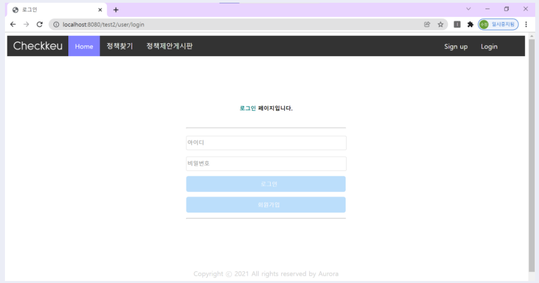
/home/main.jsp




/user/login

Login Controller

/user/loginForm.jsp

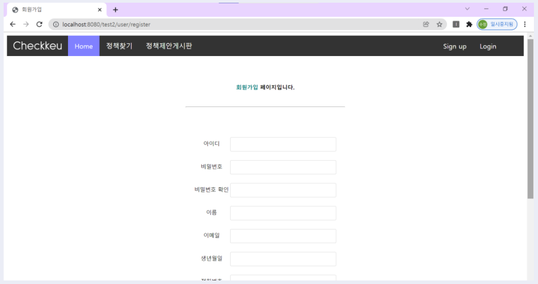



/main

/user/register

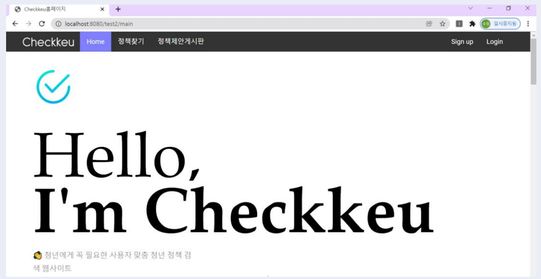
RegisterUser Controller

/user/registerForm.jsp



# REQUEST 흐름도
## : 정책검색, 스크랩

/home/main.jsp




/policy/list

ListPolicy Controller

/policy/policySearch.jsp

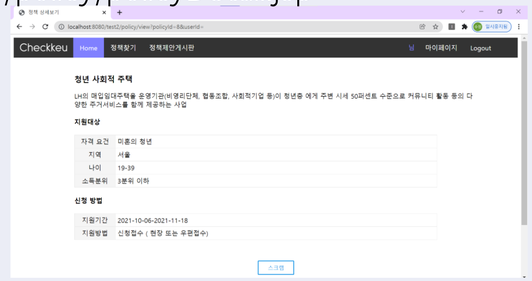

정책검색(필터링)
/policy/search

SearchPolicy Controller

/policy/policySearchTest.jsp


/policy/view

/policy/policyDetail.jsp




/policy/view

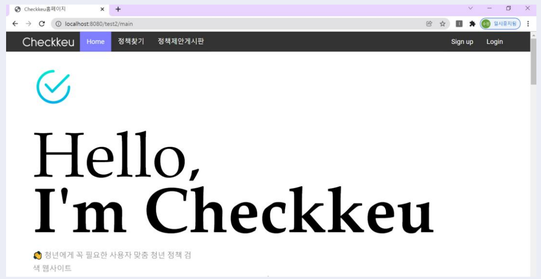
ViewPolicy Controller

/policy/scrap/add

AddScrapController Controller

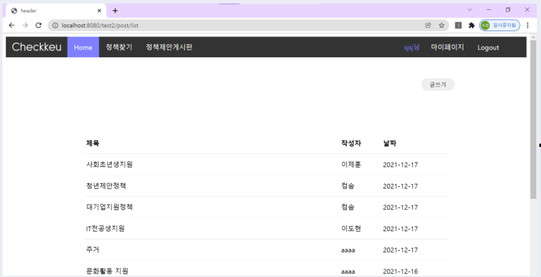
# REQUEST 흐름도
## : 커뮤니티(정책제안게시판)

/home/main.jsp




/post/list → ListPost Controller

/post/postList.jsp




글쓰기 /post/add → AddPost Controller

AddPost Controller → /post/Write.jsp


/post/view

/post/Detail.jsp


댓글쓰기 /post/reply/add → CreateReply Controller

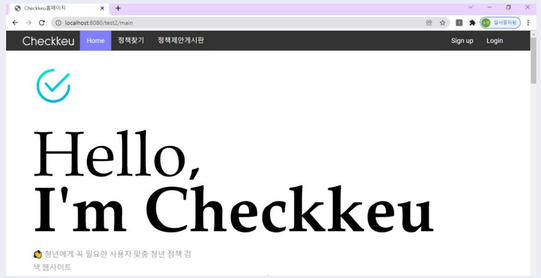
동의/비동의 /post/agree → AddAgree Controller

글 삭제 /post/Delete → DeletePost Controller

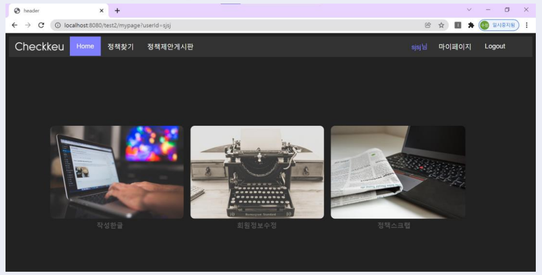
글 수정 /post/update → UpdatePost Controller

# REQUEST 흐름도
## : 마이페이지

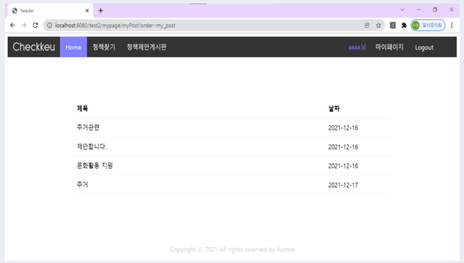
/home/main.jsp



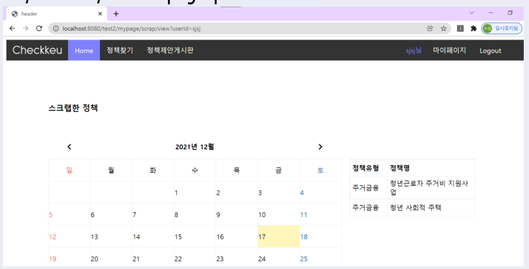

/mypage → Forward Controller → /user/mypage.jsp



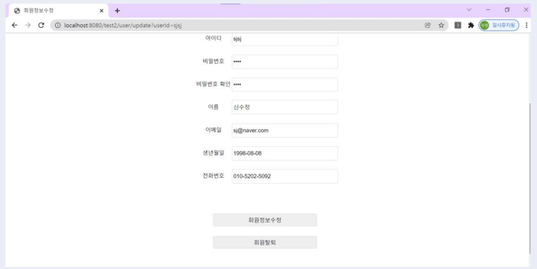

작성한 글 목록
/mypage/myPost → UserPostList Controller → /user/myPost.jsp




스크랩 목록
/mypage/scrap/view → ListScrap Controller → /user/scrap.jsp



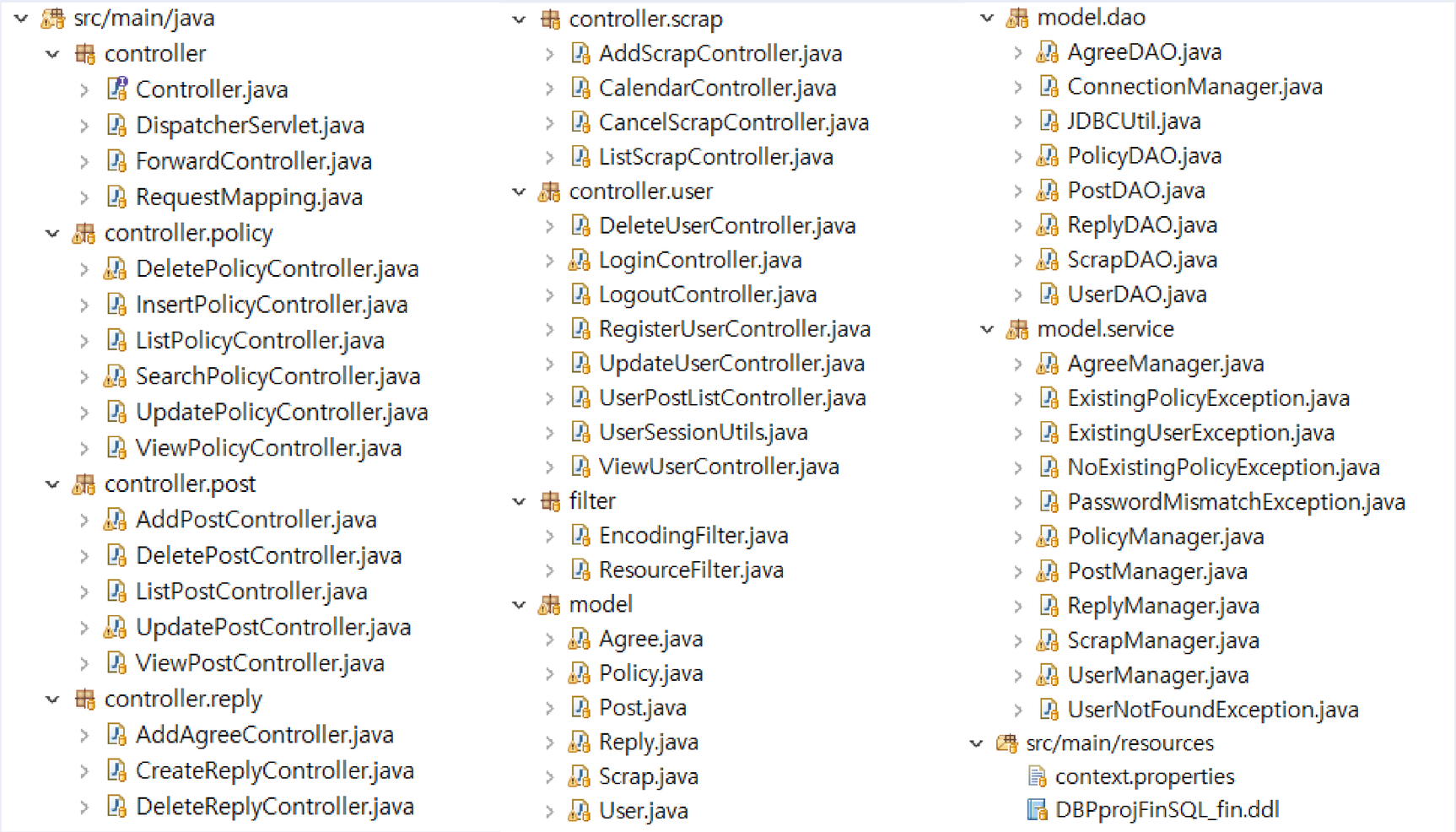

사용자 정보 수정
/user/update → UpdateUser Controller → /user/userUpdate.jsp



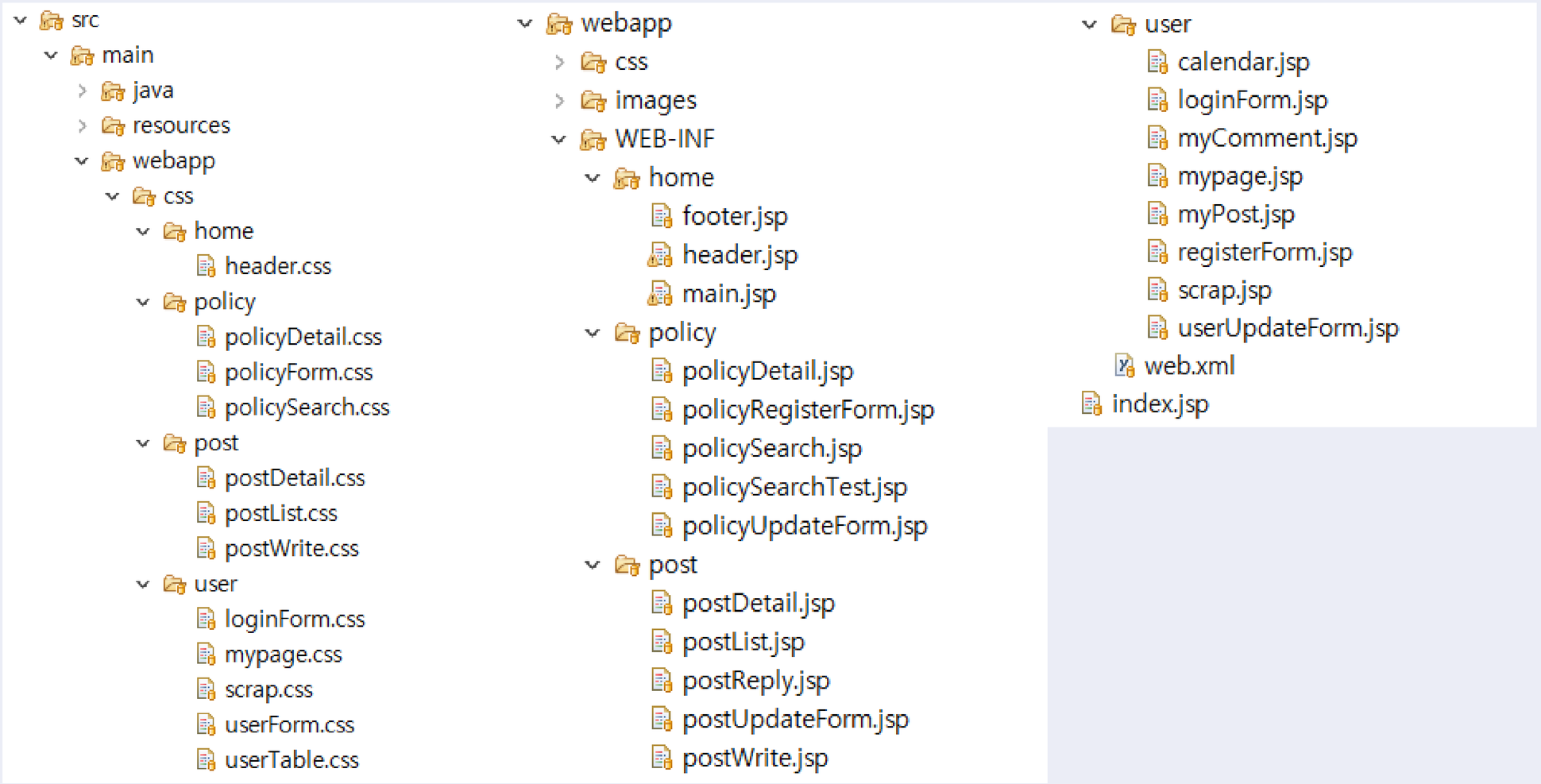

회원탈퇴
user/delete → DeleteUser Controller

# 기능 구현

# Eclipse 프로젝트 구조



# Eclipse 프로젝트 구조



# 구현된 세부 기능 목록

### 회원가입 및
### 로그인/로그아웃

사용자는 회원가입 폼을 작성해 가입
가입한 아이디와 비밀번호를 통해 로그인

### 마이페이지 –
### 작성 게시글 조회

사용자가 정책 제안 게시판에 작성한 정책 제안
게시글 조회 가능

### 마이페이지 –
### 스크랩 조회

사용자가 스크랩한 정책들의 목록과 스크랩한
날짜에 정책을 추가해 캘린더를 통해 조회 가능

### 마이페이지 –
### 회원 정보 수정 및 탈퇴

사용자의 회원 정보 폼을 통해 수정하고 탈퇴
가능

### 관리자 –
### 정책 관리

정책 관리는 사이트 관리자만 가능하게 제한
관리자는 정책 등록 폼을 작성해 등록
정책 상세 보기 페이지에서 수정, 삭제  가능

### 관리자 –
### 댓글 삭제

사용자들이 정책 제안 게시글에 작성한 댓글 삭제
가능

# 구현된 세부 기능 목록

### 정책 제안 게시판 – 게시글 작성

본인이 작성한 글 수정 및 삭제 가능

### 정책 제안 게시판 – 댓글 작성

다른 사용자들의 글에 댓글 작성 가능
동의 / 비동의 버튼을 통해 간단한 의견 제시 가능

### 정책 스크랩

정책 상세보기 페이지에서 원하는 정책 스크랩 등록 및 취소 가능

### 정책 검색

정책 유형,  나이, 지역, 소득분위 조건들을 통한 필터링 검색으로 자신에게 맞는 정책 검색 가능

# 구현된 코드 예
## : 정책 검색 기능

policySearch.jsp

```
<div class="policySearch-div">
    <form method="POST" action="<c:url value='/policy/search' />">

        <table>
        <tr>
            <th><b>정책유형</b></th>
            <td><input type="radio" name="contents" value="취업지원">취업지원
            <input type=radio name="contents" value="창업지원">창업지원
            <input type="radio" name="contents" value="주거금융">주거금융
            <input type="radio" name="contents" value="생활복지">생활복지
            <input type="radio" name="contents" value="정책참여">정책참여</td>
        </tr>
        <tr>
            <th><b>소득분위</b></th>
            <td>
            <select id="income" name="income">
            <option value="">분위</option>
            <c:forEach var="i" begin="1" end="10" step="1">
                <option value="${i}">${i}</option>>
            </c:forEach>
            </select>
            </td>
        </tr>
        <tr>
            <th><b>거주지역</b></th>
            <td>
                <input type="radio" name="local" value="전국">전국
                <input type="radio" name="local" value="서울">서울
                <input type="radio" name="local" value="경기도">경기도
                <input type="radio" name="local" value="인천">인천
                <input type="radio" name="local" value="전라도">전라도
                <input type="radio" name="local" value="경상도">경상도
                <input type="radio" name="local" value="강원도">강원도
                <input type="radio" name="local" value="충청도">충청도
                <input type="radio" name="local" value="제주도">제주도
            </td>
        </tr>

        <tr>
            <th><b>나이</b></th>
            <td>
                <select id="age" name="age">
                <option value="">나이</option>
                <c:forEach var="i" begin="0" end="30">
                    <option value="${i}">${i}</option>
                </c:forEach>
                </select>
            </td>
        </tr>
        </table>

        <div class="btn-div">
            <input class="btnStyle" type="submit" value="검색"> 
            <input class="btnStyle" type="button" value="새로고침" onClick="window.location.reload()">
        </div>

    </form>
```

검색 버튼을 누르면 검색 조건이 POST 형식으로 전달
action 속성에 지정한 url로 request 발생

# 구현된 코드 예
## : 정책 검색 기능

uri에 해당하는 Controller로 mapping하여 SearchPolicyController로 이동

RequestMapping.java

```
//policy
mappings.put("/policy/insert", new InsertPolicyController());
mappings.put("/policy/list", new ListPolicyController());
mappings.put("/policy/view", new ViewPolicyController());
mappings.put("/policy/search", new SearchPolicyController());
mappings.put("/policy/delete", new DeletePolicyController());
mappings.put("/policy/update", new UpdatePolicyController());
```

정책 찾기 페이지로 이동한 경우는 GET 방식으로 전달
policySearch.jsp 페이지를 반환

SearchPolicyController.java

```
@Override
public String execute(HttpServletRequest request, HttpServletResponse response) throws Exception {

    System.out.println("in SearchPolicyController\n");

    if (request.getMethod().equals("GET")) {
        PolicyManager polMan = PolicyManager.getInstance();
        List<Policy> findPolList = polMan.findPolicyList();

        request.setAttribute("polList", findPolList);
        return "/policy/policySearch.jsp";
    }

    PolicyManager polMan = PolicyManager.getInstance();
```

PolicyManager 객체 구함

policySearch.jsp 에서 POST 방식으로 넘겨받은 parameter (검색 조건)

```
    String category = request.getParameter("contents");
    int income = Integer.parseInt(request.getParameter("income"));
    String local = request.getParameter("local");
    int startAge = Integer.parseInt(request.getParameter("age"));
    int endAge = startAge;
```

Manager에게 검색 조건을 통해 조건에 해당하는 정책 목록을 찾는 기능 위임
찾은 정책 목록 request에 저장 후 policySearchTest.jsp 페이지로 이동

```
    List<Policy> searchPolList = polMan.searchPolicyList(category, income, local, startAge, endAge);

    request.setAttribute("searchPolList", searchPolList);
    return "/policy/policySearchTest.jsp";
}
```

PolicyManager.java

```
/* 정책 조건 검색 */
public List<Policy> searchPolicyList(String category, int income, String local, int startAge, int endAge) throws SQLException {

    return polDAO.searchPolicyList(category, income, local, startAge, endAge);
}
```

DAO 클래스의 searchPolicyList를 호출해 데이터를 전달해 처리
처리한 결과 PolicySearchController로 전달

# 구현된 코드 예
## : 정책 검색 기능

local이 전국일때는 모든 지역 조건을 만족하도록 query를 설정하고
특정 지역일 때는 local 조건을 추가하여 설정

PolicyDAO.java

```
public List<Policy> searchPolicyList(String category, int income, String local, int startAge, int endAge) throws SQLException {

    String sql = null;
    Object[] param;
    String q = "SELECT policyId, name, category, policySummary "
             + "FROM Policy ";

    if(local.equals("전국")) {
        sql = q + "WHERE category=? AND income>=? AND startAge<=? AND endAge>=? "
                + "ORDER BY policyId";

        param = new Object[] { category, income, startAge, endAge };
    }
    else {
        sql = q + "WHERE category=? AND income>=? AND local=? AND startAge<=? AND endAge>=? "
                + "ORDER BY policyId";

        param = new Object[] { category, income, local, startAge, endAge };
    }

    jdbcUtil.setSqlAndParameters(sql, param, ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

    try {
        ResultSet rs = jdbcUtil.executeQuery();
        List<Policy> polList = new ArrayList<Policy>();

        while (rs.next()) {
            logger.debug("name: "+ rs.getString("name"));

            Policy pol = new Policy(rs.getInt("policyId"),
                    rs.getString("name"),
                    rs.getString("category"),
                    rs.getString("policySummary"));

            polList.add(pol);
        }
        return polList;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        jdbcUtil.close();
    }
    return null;
```

SQL 문을 실행한 결과들을 policy 객체에  저장해 list에 추가한 후
전체 list 반환

# 구현된 코드 예
## : 정책 검색 기능

policySearchTest.jsp

```
<table>
<thead>
 <tr>
        <td><b>정책명</b></td>
        <td><b>유형</b></td>
        <td style="width: 60%;"><b>요약</b></td>
 </tr>
</thead>
<tbody>
<c:forEach var="policy" items="${searchPolList}">
    <tr>
        <td>
            <a class="aStyle" style="text-decoration: none;" href="<c:url value='/policy/view'>
            <c:param name='policyId' value='${policy.policyId}'/>
            </c:url>">
            ${policy.name}</a>
        </td>
        <td>
         ${policy.category}

        </td>
        <td>${policy.policySummary}</td>
    </tr>
</c:forEach>
</tbody>
</table>
```

SearchPolicyController에서 reqest에 저장한 list를 통해 정책 목록 출력

# 구현된 코드 예
## : 정책제안게시판

postWrite.jsp

```
<%
request.setAttribute("userId", request.getParameter("userId"));
%>

<form name="form" method="POST" action="<c:url value='/post/add' ></c:url>">
<h3 style="margin-top: 100px; margin-bottom: -70px;">정책제안하기 </h3>
    <table style="margin-top: 100px;">

            <tr>
            <td bgcolor=white>

            <table class = "table2">
                    <tr>
                    <td></td>
                    <td><input type="text" name="userId" size=20  value="${userId}" style="display: none;"> </td>
                    </tr>

                    <tr>
                    <td>제목</td>
                    <td> <input type="text" name="title" size=30 style="width: 568px; height: 20px; margin-left: 10px;"
                        <c:if test="${creationFailed}"> value="${post.title}"</c:if>>     </td>
                    </tr>

                    <tr>
                    <td>내용</td>
                    <td><textarea name="content" cols=85 rows=15 style="width: 570px; margin-left: 10px;"></textarea></td>
                    </tr>
                    <!-- 임시/ test용  -->
                    <tr>
                    <td></td>
                    <td> <input type="text" name="writeDate" size=20 style="display: none;" value="<%= sf.format(nowTime) %>" >
                              </td>
                    </tr>
                    <tr>
                    <td></td>
                    <td> <input type="text" name="policyId" size=20 value=1 style="display: none;">
                          </td>
                    </tr>
                    </table>
            </td>
            </tr>
    </table>
```

postWrite.jsp 에서 지정한 url로 POST 방식으로 입력한 parameter 전달

# 구현된 코드 예
## : 정책제안게시판

RequestMapping.java

```
//post
mappings.put("/post/write", new ForwardController("/post/postWrite.jsp"));

mappings.put("/post/add", new AddPostController());
mappings.put("/post/list", new ListPostController());
mappings.put("/post/view", new ViewPostController());
mappings.put("/post/delete", new DeletePostController());
mappings.put("/post/update", new UpdatePostController());
```

Mapping된 Controller 요청

postWrite.jsp 에서 POST 방식으로 넘겨받은 parameter (게시글 정보)

AddPostController.java

```
String userId = request.getParameter("userId");
int policyId = Integer.parseInt(request.getParameter("policyId"));
String uId = request.getParameter("userId");
String title = request.getParameter("title");
String writeDate = request.getParameter("writeDate");
String content = request.getParameter("content");

Post post = new Post(
        0,
        Integer.parseInt(request.getParameter("policyId")),
        request.getParameter("userId"),
        request.getParameter("title"),
        request.getParameter("writeDate"),
        request.getParameter("content"));
try {
    PostManager postMan = PostManager.getInstance();
    postMan.insert(post);

    return "redirect:/post/list";
    return "redirection:/post/list";


} catch (Exception e) { 실패 시 다시 게시글 입력 화면으로 이동
    request.setAttribute("insertFailed", true);
    request.setAttribute("exception", e);
    request.setAttribute("post", post);
    System.out.print(e); //임시
    return "/post/postWrite.jsp";
}
```

전달 받은 parameter post 객체에 저장

PostManager의 insert 메소드에 post 객체를 전달

PostManager.java

```
public Post insert(Post post) throws SQLException {
    return postDAO.insertPost(post);
}
```

# 구현된 코드 예
## : 정책제안게시판

PostDAO.java

```
public Post insertPost(Post po) throws SQLException {

    int generatedKey;

    String sql = "INSERT INTO Post VALUES (postNumSeq.nextval, ?, ?, ?, ?, ?)";

    Object[] param = new Object[] { po.getTitle(), po.getWriteDate(), po.getContent(), po.getUserId(),
            po.getPolicyId() };

    jdbcUtil.setSqlAndParameters(sql, param);

    String key[] = { "postNum" };

    try {
        jdbcUtil.executeUpdate();

        ResultSet rs = jdbcUtil.getGeneratedKeys();
        if (rs.next()) {
            generatedKey = rs.getInt(1);
            po.setPostNum(generatedKey);
        }
        return po;
    } catch (Exception e) {
        jdbcUtil.rollback();
        e.printStackTrace();
    } finally {
        jdbcUtil.commit();
        jdbcUtil.close();
    }

    return null;
}
```

INSERT query 실행

DB에 결과를 저장한 후 post 객체 반환

# 구현된 코드 예
## : 정책제안게시판

ListPostController.java

```java
@Override
public String execute(HttpServletRequest request, HttpServletResponse response) throws Exception {

	PostManager postMan = PostManager.getInstance();
	List<Post> postList = postMan.findPostList();

	request.setAttribute("postList", postList);
	return "/post/postList.jsp";

}
```

등록된 전체 게시글 목록 찾기

postList.jsp

```jsp
<tbody>
  <c:forEach var="post" items="${postList}">
      <tr>
        <td style="width: 70%"><a class="aStyle" style="text-decoration: none;" href="<c:url value='/post/view'>
                                <c:param name='postNum' value='${post.postNum}'/>
                            </c:url>">
            ${post.title} </a>
        </td>
        <td>${post.userId}</td>
        <td>${post.writeDate}</td>
      </tr>
  </c:forEach>
</tbody>
```

request에 저장된 게시글 list 출력

전체 정책 목록에서 정책 클릭 시 지정 url로 이동
이동 시 클릭한 게시글 번호를 parameter로 전달

RequestMapping.java

```java
//post
mappings.put("/post/write", new ForwardController("/post/postWrite.jsp"));

mappings.put("/post/add", new AddPostController());
mappings.put("/post/list", new ListPostController());
mappings.put("/post/view", new ViewPostController());
mappings.put("/post/delete", new DeletePostController());
mappings.put("/post/update", new UpdatePostController());
```

# 구현된 코드 예
## : 정책제안게시판

ViewPostController.java

```
PostManager postMan = PostManager.getInstance();
ReplyManager reMan = ReplyManager.getInstance();
AgreeManager aMan = AgreeManager.getInstance();

int postNum = Integer.parseInt(request.getParameter("postNum"));

Post post = null;
Agree agree = null;
post = postMan.findPost(postNum);
agree = aMan.findAgree(postNum);
List<Reply> replyList = reMan.findReplyList(postNum);

request.setAttribute("replyList", replyList);
request.setAttribute("post", post);
request.setAttribute("agree", agree);

return "/post/postDetail.jsp";
```

전달받은 게시글 번호를 통해
해당 게시글의 상세 정보, 동의 / 비동의 정보,
게시글에 달린 댓글 list를 찾아 request에 저장

postDetail.jsp

```
        <h5><span>글 상세보기</span></h5>
<table>

      <tr>
        <th>제목</th>
        <td>${post.title}</td>
      </tr>
      <tr>
        <th>작성자</th>
        <td>${post.userId}</td>
      </tr>
      <tr>
        <th>날짜</th>
        <td>${post.writeDate}</td>
      </tr>
      <tr>
        <th>내용</th>
        <td>${post.content}</td>
      </tr>

</table>
```

# 구현된 코드 예
## : 정책제안게시판

postDetail.jsp

```
<form name="agreeForm" method="POST" action="<c:url value='/post/agree'></c:url>">
    <div style="text-align:center;">
    <input type="radio" name="agree" value="agree" checked="checked" class="feeling_a">동의
    <input type="radio" name="agree" value="disagree" class="feeling_a">비동의
    <input type="text" name="postNum" size=20  value="${post.postNum}" style="display: none;">
    <button type="submit" >Sure?</button>
    </div>
        </form>
```

지정한 url로 작성한 동의 / 비동의 정보 전달

RequestMapping.java

```
mappings.put("/post/reply/add", new CreateReplyController());
mappings.put("/post/reply/delete", new DeleteReplyController());

mappings.put("/post/agree", new AddAgreeController());
```

지정한 url로 작성한 댓글 정보 전달

```
<form name="form" method="POST" action="<c:url value='/post/reply/add
</c:url>">
    <input type="text" name="postNum" size=20  value="${post.postNum}" style="display: none;">
    <input type="text" name="agree" size=20  value='n' style="display: none;">
    <input type="text" name="disagree" size=20  value='n' style="display: none;">
    <input type="text" name="replyContent" placeholder="댓글을  작성하세요" style="width:85%; height: 50px; border-radius: 8px; border: none; background-color: #F5F5F5;">

    <button type="button" class="w-btn-green"  style="border: none; height: 50px; width: 50px; border-radius: 8px; " onClick="createReply(this.form)">등록</button>  
</form>
</div><br/>
```

# 구현된 코드 예
## : 정책제안게시판

CreateReplyController.java

```java
int postNum = Integer.parseInt(request.getParameter("postNum"));

Reply re = new Reply ();
re = new Reply (
        postNum,
        request.getParameter("replyContent"),
        0);
try {
    ReplyManager reMan = ReplyManager.getInstance();
    reMan.create(re);

    request.setAttribute("postNum", postNum);

    return "redirect:/post/view?postNum=" +postNum;
} catch (Exception e) {
    request.setAttribute("createReplyFailed", true);
    request.setAttribute("exception", e);
    request.setAttribute("reply", re);

    return "redirect:/post/view?postNum=" + postNum;
}
```

댓글 생성 후 request에 저장
url에 postNum과 함께 반환

AddAgreeController.java

```java
int postNum = Integer.parseInt(request.getParameter("postNum"));
String agree = request.getParameter("agree");

Agree a = null;

try {

    if(aMan.findAgree(postNum) == null) {
        a = aMan.create(postNum);
    }
    if (agree.equals("agree")) {
        aMan.addAgree(postNum);

    }
    else {
        aMan.addDisAgree(postNum);
    }

    a = aMan.findAgree(postNum);
    request.setAttribute("agree", request.getParameter("agree"));

    request.setAttribute("agr", a.getAgree());
    request.setAttribute("disagr", a.getDisagree());

    return "redirect:/post/view?postNum=" +postNum;

} catch (Exception e) {
    request.setAttribute("createAgreeFailed", true);
    request.setAttribute("exception", e);
    request.setAttribute("agree", a);

    return "redirect:/post/view?postNum=" + postNum;
```

postNum에 해당하는 게시글에 저장된
동의 / 비동의 객체를 찾아 저장
url에 postNum과 함께 반환

# 구현된 코드 예
## : 정책제안게시판

postNum에 해당하는 게시글의 동의 / 비동의 의견 수 증가

AgreeDAO.java

```java
public int addAgree (int postNum) throws SQLException{
    String sql = "UPDATE AGREE "
            + "SET AGREE = AGREE + 1 "
            + "WHERE POSTNUM=? ";
    Object[] param = new Object[] {postNum};
    jdbcUtil.setSqlAndParameters(sql, param);
    try {
        int result = jdbcUtil.executeUpdate();
        return result;
    } catch (Exception e) {
        jdbcUtil.rollback();
        e.printStackTrace();
    } finally {
        jdbcUtil.commit();
        jdbcUtil.close();
    }
    return 0;
}
```

```java
public int addDisagree (int postNum) throws SQLException{
    String sql = "UPDATE AGREE "
            + "SET DISAGREE = DISAGREE + 1 "
            + "WHERE POSTNUM=? ";
    Object[] param = new Object[] {postNum};
    jdbcUtil.setSqlAndParameters(sql, param);
    try {
        int result = jdbcUtil.executeUpdate();
        return result;
    } catch (Exception e) {
        jdbcUtil.rollback();
        e.printStackTrace();
    } finally {
        jdbcUtil.commit();
        jdbcUtil.close();
    }
    return 0;
}
```

# 구현된 코드 예
## : 정책제안게시판

postDetail.jsp

```
<div class="feeling_div" >
        <div class="button-container like-container">
            <button class="feeling_a" disabled="disabled">
              <i>♥YES</i>
              <div id='result_p'>${agree.agree}</div>
              </button>
          </div>
          <div class="button-container dislike-container">
            <button class="feeling_a" disabled="disabled">
              <i>♡NO</i>  <!--{ $like_sum }-->
              <div id='result_m'>${agree.disagree}</div>
              </button>
        </div>
</div>
<c:forEach var="cm" items="${replyList}" varStatus="status" >
<div style="padding-left: 30px; padding-right: 30px; width:85%; height: 50px; border: solid #F5F5F5 1px; background-color: none;">${cm.replyContent}

    <c:choose>
        <c:when test="${userId=='dbpro0102'}">
            <a style=" float: right;padding-right: 10px; font-size: 10px;" id="btn" href="<c:url value="/post/reply/delete">
                    <c:param name='replyNum' value="${cm.replyNum}"/>
                    <c:param name='postNum' value="${post.postNum}"/>
             </c:url>" onclick="replyDelete()">  삭제</a>
        </c:when>
    </c:choose>
</div>

</c:forEach>
```

agree 객체에 저장된 동의 / 비동의 의견 개수 출력

게시글에 달린 전체 댓글 출력

사용자 ID가 관리자의 ID일 경우 댓글에 '삭제 ' 버튼 출력

# Business logic/algorithm

정책검색

사용자가 원하는 정책의 조건을 저장하고 PolicyManager에서 DAO를 호출

DB 연결

SELECT query를 통해 연결된 DB에 조건에 해당하는 정책이 있는지 검색
만약 local이 “전국”일 경우 local은 조건 고려하지 않음, 특정 지역일 경우 query의 WHERE 부분에 local 조건 추가

검색된 결과를 list에 저장

DB 연결 종료

View 영역에 저장한 데이터 전달

# 문제해결방법

대부분의 SQL 오류 (ORA-01841, ORA-00932 등)
실제 DB에 저장된 column의 순서와 DAO에서 저장한 SQL문의 column 순서 확인으로 해결

HTTP 상태 404 오류 (Origin 서버가 대상 리소스를 위한 현재의 representation을 찾지 못했거나, 그것이 존재하는지를 밝히려 하지 않습니다.)
Servers > tomcat > Modules > Path를 '/'로 재설정하여 해결

GIT 오류 (Checkout conflict with files: (file path))
강제 reset 후 pull 재시도

GIT 오류 (pull / push 충돌)
Git Repositories > project > remotes > origin > 첫번째 configure fetch > Advanced
> Remove 후 Add Spec > finish > save and fetch > 재시도

# 마무리

# 팀원별 업무 분담

이지영

프론트엔드 전반
view(jsp/css)/DB query
post/user 기능

신수정

프론트엔드 전반
view(jsp/css)/DB query
policy/home 기능

진고은

백엔드 전반
service/model
post/user 기능

최주리

백엔드 전반
dao/controller
policy/home 기능

# 소감

## 이지영

JDBC를 활용한 자바와의 연동 방식과 MVC 구조에 대해서 잘 알게 되었다. 프로젝트를 하면서 기술적인 면에서도 많은 걸 배웠다. 필요한 부분이 생겨 erwin을 이용해 DB를 수정하면서 기초가 중요하다는 말에 공감하게 되었다. 이번 경험을 통해 기초가 탄탄한 개발을 할 수 있을 것 같다. 컨트롤러에서 비즈니스 로직을 통해 원하는 값을 얻어와 실행 창에 보일 때 기분이 좋았다. 진행에 어려움도 많았지만 다양한 오류를 경험하고 해결 방법을 알아가는 게 도움이 많이 되었다. 직접 기획하고 기획한 부분을 구현했을 때 뿌듯했다. 청년들을 위한 청년 정책 사이트를 개발하였는데 여러 사람에게 도움이 되는 개발을 하였다는 게 더욱 큰 의미가 있는 것 같다. 깃허브를 통해 프로젝트를 진행하면서 맡은 부분을 확실히 정하면서 하는 게 중요하다고 느꼈고 팀원들이 맡은 부분을 잘해주어 고마웠다. 각자 잘하고 흥미로운 부분을 맡는 게 진행에도 도움이 많이 되었다. 역량이 높은 팀원들과 함께해서 재밌게 했던 개발이었다.

## 신수정

팀원들과 처음 아이디어 기획부터, 디자인, 개발까지 직접 모든 과정에서 참여하는 것이 재미있었다. 지금까지 배웠던 개념들을 프로젝트를 하면서 직접 적용해 볼 수 있었다. 그동안 헷갈렸던 부분들은 프로젝트를 하며 확실하게 깨닫게 되었다. 또한 변수명의 중요성도 알게 되었다. 각자 맡은 페이지가 있다 보니 변수가 서로 다른 경우가 있었는데 나중에 통일하는 과정이 필요했다. 다음번에는 처음부터 상의하여 변수명을 같이 정하고 시작해야겠다. 프로젝트 초반에는 깃,이클립스,erwin 모두에서 에러가 정말 많이 생겨서 에러 고치는데 시간이 많이 걸렸었다. 그때마다 팀원들과 에러별로 해결방법을 정리해서 모아두었다. 이러한 과정을 거치면서 이제는 에러가 떠도 당황하지 않고 어떻게 해결해야 되는지 알게 되었고 덕분에 프로젝트 진행에 속도가 붙었다. 프로젝트 과정에서 힘든 점도 있었지만 하나하나 완성해 나갈때마다 뿌듯함이 더 컸다. 무엇보다 직접 생각한 아이디어를 실전으로 옮겨 웹페이지로 만들 수 있다는 점이 좋았다.

## 진고은

프로젝트 진행을 통해 어려웠던 부분을 더 수월하게 이해할 수 있었고 팀원들과의 지속적인 질문과 답변을 통해 서로 발전할 수 있었다. 협업을 통한 진행을 통해서 후에 있을 여러 개발에도 많은 도움이 될 것 같다. 모델에서의 데이터 관리와 컨트롤러를 주로 맡아서 진행했었는데 데이터 처리 방법 및 모델과 뷰 사이 실행 흐름에 대해서 특히 더 배우는 기회가 되어 좋은 경험이 되었다. 진행하면서 다양한 오류를 반복적으로 해결하면서 역량이 발전했음을 느꼈고 몇몇 오류는 인터넷의 도움없이도 해결할 수 있을 것 같다는 생각이 들었다. 각자의 역할이 중요했던 프로젝트였던 만큼 책임감을 가지고 참여했고 팀원들에게 많은 도움을 받고 함께 진행하면서 전보다 발전된 개발 능력을 확인 할 수 있었고 각자의 역할이 얼마나 중요한지 느낄 수 있었던 활동이었다.

## 최주리

처음 시작할 때는 웹페이지를 과연 내가 구현할 수 있을지 조차 확신하지 못했는데 팀원들과 머리를 맞대고 고민하며 교수님께 모르는 것들을 질문하고 답을 얻어가는 과정을 통해 발전하고 자신감을 가져 프로젝트를 완성할 수 있었습니다. 팀프로젝트에서 맡은 역할에 책임을 다하는 것에 대한 중요성을 느꼈고 프로젝트를 진행하기 전보다 발전된 개발실력을 느끼고 앞으로 진행할 프로젝트를 이 경험을 바탕으로 잘 진행할 수 있을 것 같습니다.

# 시연

# Thank you