



# 스마트 마이팜

# Smart My Farm

ver. 1.0



Spring, MySQL, Python Project

한이진

# 목차 CONTENTS



기획 의도

개발 환경

요구사항 정의서

개선 및 발전

기술 상세

DB 설계

# 기획 의도



## 미래의 생동력 스마트팜!

인공지능과 IoT, 빅데이터 분석과 클라우드 서비스 기술과 자동화 기술을 융합해 접목한 미래 대응형 첨단 농업시스템.

## 차세대 농업을 위한 정보공개!

농업 기술, 종자 정보, 다양한 양질의 데이터를 사이트로 원활한 데이터 수집과 관리하여 스마트팜 발전에 이바지 함으로써 차세대 먹거리 문제 해결 .

## Smart My Farm (ver. 1.0)

## 미래 먹거리 책임자는 우리!

농업인들의 기술 정보 공유  
품질을 높이는 유통과 소비를 위해 소비자에게 정확한 정보 제공

# 개발 환경

---



## spring<sup>®</sup>

자바 플랫폼을 위한 오픈 소스  
애플리케이션 프레임워크

공공기관의 웹 서비스 개발 시 사용  
권장하는 전자정부 표준프레임워크



## eclipse

자바를 비롯한 다양한 언어를 지원하는  
통합 개발 환경 (IDE)

범용 응용 소프트웨어 플랫폼



## MySQL<sup>®</sup>

오픈 소스의  
관계형 데이터베이스 관리 시스템  
(RDBMS)  
다중 스레드, 다중 사용자  
구조질의어 형식의  
데이터베이스 관리 시스템

# 개발 환경

## Maven 기본 폴더 구조

src/main/java\_ 자바 코드

src/main/resources\_ XML, 프로퍼티 파일

```
└── src
    └── main
        ├── java
        │   ├── kr.co.util
        │   ├── kr.co.yj.controller
        │   ├── kr.co.yj.dao
        │   ├── kr.co.yj.service
        │   └── kr.co.yj.vo
        └── resources
            ├── mappers
            │   ├── boardMapper.xml
            │   ├── MemberMapper.xml
            │   ├── replyMapper.xml
            │   └── tomatoMapper.xml
            └── META-INF
                ├── log4j.xml
                └── log4jdbc.log4j2.properties

```

src/main/webapp\_ 웹 어플리케이션 개발 기준 폴더  
JSP 소스 코드, WEB\_INF/web.xml 파일 등이 위치.

```
└── src
    └── main
        ├── java
        ├── resources
        └── webapp
            ├── resources
            └── WEB-INF
                ├── classes
                ├── lib
                └── spring
                    ├── appServlet
                    │   ├── context-common.xml
                    │   ├── root-context.xml
                    │   └── spring-security.xml
                    └── views
                        └── web.xml
            └── test
            └── target
                └── pom.xml
```



# 개발 환경

## Maven 의존 설정\_pom.xml



자바용 프로젝트 관리 도구

아파치 라이선스로 배포되는 오픈 소스 소프트웨어

pom.xml에서 메이븐 프로젝트에 대한 설정 정보를 관리

프로젝트에서 필요로 하는 의존 모듈이나 플러그인 등에 대한 설정함.

<dependency> 설정을 알맞게 추가하면 메이븐 중앙 리포지토리에 아티팩트 파일을 등록하여 필요한 jar파일을 손쉽게 추가함.

MySql, spring-jdbc, junit, spring-test, log4jdbc, mybatis, mybatis-spring 추가

```
<!-- mysql -->
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.49</version>
</dependency>
```

```
<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>

<!-- spring-test -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${org.springframework-version}</version>
    <scope>test</scope>
</dependency>

<!-- log4jdbc-log4j2-jdbc4 -->
<dependency>
    <groupId>org.bgee.log4jdbc-log4j2</groupId>
    <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
    <version>1.16</version>
</dependency>

<!-- org.mybatis/mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.1</version>
</dependency>

<!-- mybatis-spring -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.0</version>
</dependency>
```

# 개발환경

## WEB-INF/web.xml

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>
```

## WEB-INF/spring/root-context.xml

## Mysql + mybatis 연동

```
<!-- mysql 접속 -->
<beans:bean id="ds"
  class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
  <beans:property name="driverClassName"
    value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></beans:property>
  <beans:property name="url"
    value="jdbc:log4jdbc:mysql://localhost:3305/mybatis_db?allowPublicKeyRetrieval=true&useSSL=false&useTimezone=true" />
  <beans:property name="username" value="mybatis"></beans:property>
  <beans:property name="password" value="1234"></beans:property>
</beans:bean>
<beans:bean id="ssf" class="org.mybatis.spring.SqlSessionFactoryBean">
  <beans:property name="dataSource" ref="ds"/>
  <beans:property name="configLocation" value="classpath:/mybatis-config.xml"/>
  <beans:property name="mapperLocations" value="classpath:mappers/**/*Mapper.xml"/>
</beans:bean>
<beans:bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate" destroy-method="clearCache">
  <beans:constructor-arg name="sqlSessionFactory" ref="ssf"/>
</beans:bean>
<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <beans:property name="prefix" value="/WEB-INF/views/" />
  <beans:property name="suffix" value=".jsp" />
</beans:bean>
```

spring MVC가 웹 요청을 처리하기 위한 web.xml 파일에  
DispatcherServlet을 등록

# Mysql + mybatis 연동 설정

# 개발 환경

## Dependency Injection 의존주입

WEB-INF/web.xml

```
<init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/spring/appServlet/servlet-context.xml
        /WEB-INF/spring/spring-security.xml
        /WEB-INF/spring/context-common.xml
    </param-value>
</init-param>
```

WEB-INF/spring/servlet-context.xml

```
<context:component-scan base-package="kr.co.yj.*" />
<context:component-scan base-package="kr.co.util"></cont
```

base package 기준으로 클래스들을 스캔

base package 하위의 @Controller, @Service, @Repository 클래스가 모두 빈으로 등록

spring은 객체 컨테이너

객체를 생성하고 의존성 (DI)을 설정하는 것이 중요

AnnotationConfigApplicationContext 자바 클래스에서 설정하거나  
XML파일을 이용하여 객체를 생성하고 객체간의 의존성을 관리.

component-scan

bean으로 등록 된 클래스들을 스캔하여 bean으로 등록

@Controller, @Service, @Component, @Repository 어노테이션을  
붙인 클래스들이 bean으로 등록된 것.

```
@Controller
@RequestMapping("/board/*")
public class BoardController {
    @Inject
    BoardService bservice;
    @Inject
    ReplyService rservice;
```

```
@Service
public class BoardService {
    @Inject
    private BoardDAO bDao;
```

```
@Repository
public class BoardDAO {
    @Inject
    private SqlSession sqlSession;
```

component-scan은 기본적으로 @Component 어노테이션을 bean등록 대상으로 포함.  
@Component는 @Controller나 @Service, @Repository 를 포함

# MVC 패턴

## MODULE

- > Board.java
- > Criteria.java
- > Member.java
- > PageMaker.java
- > Reply.java
- > SearchCriteria.java
- > Tomato.java
- \*

## VIEW

- > > board
  - > list.jsp
  - > navbar.jsp
  - > readView.jsp
  - > replyDeleteView.jsp
  - > replyUpdateView.jsp
  - > updateView.jsp
  - > writeView.jsp
- > > Member
  - > loginView.jsp
  - > memberDeleteView.jsp
  - > memberUpdateView.jsp
  - > signup.jsp
- > > tomato
  - > home.jsp

## CONTROLLER

- > > kr.co.yj.controller
  - > > BoardController.java
  - > HomeController.java
  - > MemberController.java
  - > ReplyController.java
  - > TomatoController.java
- > > kr.co.yj.dao
- > > kr.co.yj.service

# 요구사항 정의서

Main page

ID	이 름	내 용	유형	속성	우선순위	전제조건
SF_001	메인화면 로그인 배치 로그인 페이지	우측 상단바에 "회원관리-로그인" 을 배치, 클릭을 하면 로그인 페이지로 이동 "아이디", "비밀번호"칸 기입후 완료를 위한 "로그인"버튼을 배치. 그아래 회원가입을 위한 "회원가입" 아이콘 배치	비기능	화면	필수	-
SF_002	로그인 기능	SF_001에서 기술한 로그인 페이지에서 "로그인"을 누르면 DB의 회원정보를 근거하여 로그인이 진행.	기능	N/A	필수	-
SF_003	회원가입(화면구성)	우측 상단바에 "회원관리-회원가입" 배치 클릭시 페이지로 이동. "아이디", "비밀번호", "비밀번호 확인", "전화번호", "주소" 기입란을 배치, 아이디 확인 버튼 배치	비기능	화면	필수	-
SF_004	회원가입(DB 저장)	회원가입 페이지(SF_003)의 기입란을 작성후 "확인"을 누르면 회원가입이 진행 기입한 정보를 기반으로 가입자 정보가 DB에 저장.	기능	-	필수	SF_003
SF_005	회원 정보 수정	SF_002를 통해 로그인 완료시 회원 정보란 아래 "회원정보수정" 배치	비기능	화면	필수	SF_002
SF_006	회원정보 수정	SF_005를 통해 회원정보 수정 페이지로 이동 후 기입란에 정보수정 후 "완료" 버튼 클릭시 동일한 "아이디"조건으로 DB에 저장된 정보 수정	기능	-	필수	SF_002
SF_007	로그아웃	SF_002를 통해 로그인 시 우측 상단 "회원관리-로그아웃" 생성 클릭시 Session 삭제 후 메인페이지로 이동	인터페이스	-	필수	SF_002
SF_008	공지사항 페이지	운영자가 공지할 수 있는 게시판 메인화면에 위치	비기능	화면	선택	
SF_009	게시판 페이지	상단바에 "게시판" 클릭시 게시판 페이지 이동 번호, 작성자, 작성일자로 구성, 제목, 작성자, 내용 기반으로 게시글 검색 선택란과 키워드 기입란 배치, 하단 "글쓰기" 페이지 이동 버튼 배치 최신순으로 게시글 배열	비기능	화면	필수	-

# 요구사항 정의서

Free Board

ID	이 름	내 용	유형	속성	우선순위	전제 조건
SF_010	게시판 페이지 이동	DB저장된 게시글 기준으로 게시글 10개 씩 페이지 분할	인터페이스	화면	필수	-
SF_011	게시글 작성하기	SF_009을 통해 페이지 이동, SF_002를 완료시 작성란 배치 “제목”, “작성자”, “내용” 기입란 배치, 하단 “완료” 버튼 배치	비기능	화면	필수	SF_002 SF_009
SF_012	게시글 작성하기(DB저장)	SF_012를 통해 기입된 정보 DB의 게시글에 저장	기능	-	필수	SF_002 SF_011
SF_013	게시글 검색하기	<b>제목, 작성자, 내용, 내용+제목 기준 선택, 키워드 기반으로 DB에 저장된 해당 게시글 제공(최신순)</b>	인터페이스	-	필수	SF_009
SF_014	게시글 읽기	SF_009에 제공되는 게시글 선택시 해당 게시글 상세보기 이동, 로그인 정보 일치시 수정, 삭제 버튼 배치	기능/화면	-	필수	SF_002
SF_015	게시글 수정하기	SF_002 조건 성립 시 SF_014의 “수정” 버튼을 통해 수정 페이지 이동 해당 게시글 수정 기입란 작성 후 “수정” 버튼 클릭시 DB저장 정보 수정	기능	-	필수	SF_002 SF_014
SF_016	게시글 삭제하기	SF_002 조건 성립 시 SF_014의 “삭제” 버튼을 통해 해당 게시글의 DB 정보 삭제	기능	-	필수	SF_002 SF_014
SF_017	댓글 작성(화면)	SF_014의 해당 게시글 아래 댓글, 댓글 작성란 배치	비기능	화면	선택	SF_014
SF_018	댓글 작성하기(DB저장)	SF_017의 댓글 작성란 작성된 정보 “작성” 버튼 클릭시 DB에 저장	기능	-	선택	SF_017
SF_019	댓글 수정 및 삭제	SF_002 조건 성립시 해당 회원 작성한 댓글 수정 및 삭제	기능	-	선택	SF_002 SF_017

# 요구사항 정의서

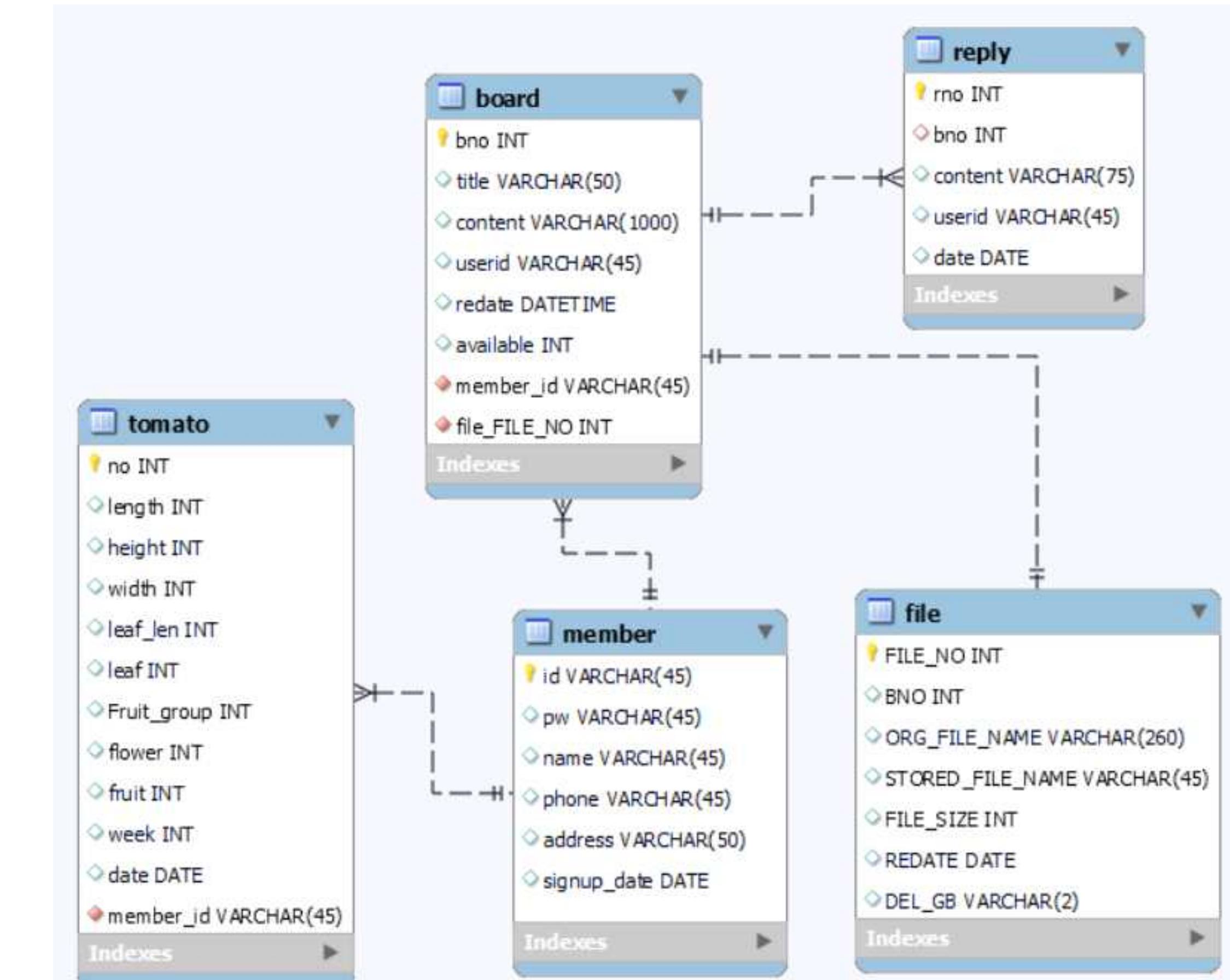
생장 일지

ID	이 름	내 용	유형	속성	우선순위	전제 조건
SF_020	생장일지 (마이)페이지	SF_002를 통해 로그인시 상단바에 "생장일지" 을 배치, 클릭시 회원정보 기준의 해당 페이지로 이동 DB에 저장된 생장일지의 "주차", "작성일자", "생장길이" 정보 제공 하단에 "일지 작성하기" 버튼 배치	비기능	화면	필수	SF_002
SF_021	생장일지 작성하기(화면)	SF_021에서 "일지작성" 클릭하여 작성하기 페이지 이동 "주차", "작성일자", "생장길이"등 작물 생육정보 기입란 배치	비기능	화면	필수	SF_020
SF_022	생장일지 작성하기	생육정보 기입하여 "작성완료" 버튼 클릭시 해당 DB에저장	기능	-	필수	SF_021
SF_023	생장일지 읽기	SF_020에 제공되는 글 선택시 해당 작성된 글 페이지 이동, DB에 저장된 생육정보 제공 하단에 "수정", "삭제" 버튼 배치(화면)	기능	-	필수	SF_020
SF_024	생장 일지 수정하기	SF_023을 "수정"버튼 클릭 수정 페이지 이동, 기입란에 작성한 정보, DB에 해당 글번호 기준으로 정보 수정	기능	-	필수	SF_023
SF_025	생장일지 삭제하기	SF_023에서 "삭제" 버튼 클릭 시 해당 글 번호 기준으로 DB에 저장된 정보 삭제	기능	-	필수	SF_023
SF_026	회원 탈퇴	해당 회원 비밀번호 일치 시 DB에 저장된 회원정보 삭제로 기능 수행	기능	-	필수	SF_002

# DB 설계 ERD

SMART\_MY\_FARM\_DB

Member\_ 회원 테이블  
Board\_ 게시판 테이블  
Reply\_ 댓글 테이블  
Tomato\_ 생장일지 테이블  
(추후 농작물별 테이블 추가)  
File\_ 파일 테이블



HOME My Farm Stroy 생장일지 회원관리 ▾



로그인

회원가입

# 똑똑한 마이팜

## Smart My Farm

아이디 han 비밀번호 \*\*\*\*

로그인 회원가입

### 새소식

'토마토' 항산화 물질 높이는 재배 기술 주목

농촌진흥청(청장 허태웅)은 고추의 항산화 물질을 높리기 위해 개발한 액비(물비료) 재배 기술을 토마토에 적용했을 때도 우수한 효과가 있음을 확인했다고 밝혔다.

액비 제조      토마토 포장      과실채취

항산화 물질분석

폭염 등 기상재해 예방·복구 위해 현장 영농기술 지원 강화



# 기술 상세

## 회원가입\_아이디 중복 검사

### views/Member/signup.jsp

아이디 입력 후 '중복확인' 버튼 클릭

### co.kr.yj.controller.MemberController

```
int result = mservice . idChk(member);
try {
    if (result == 1) {
        return "Member/signup";
    } else if (result == 0) {
        mservice . memberInsert(member);
    }
}
```

### co.kr.yj.service.MemberService

```
@Inject MemberDAO mdao;
public void memberInsert(Member meber) ...
    { mdao . memberInsert(member); }
```

### co.kr.yj.dao.MemberDAO

```
@Inject SqlSession sqlSession;
sqlSession.insert("memberMapper.insert", member);
```

localhost:8000 내용:  
중복된 아이디입니다.

확인

회원가입

회원가입

### resources/mepper/MemberMapper.xml

```
<insert id="insert" parameterType="kr.co.yj.vo.Member">
    Insert into member
        (id,pw ,name,phone,address,signup_date)
    values (#{id},#{pw },#{name},#{phone},#{address},now())
</insert>
```

### views/Member/loginView.jsp

아이디, 비밀번호 입력 후 '로그인' 버튼 클릭

### co.kr.yj.controller.MemberController

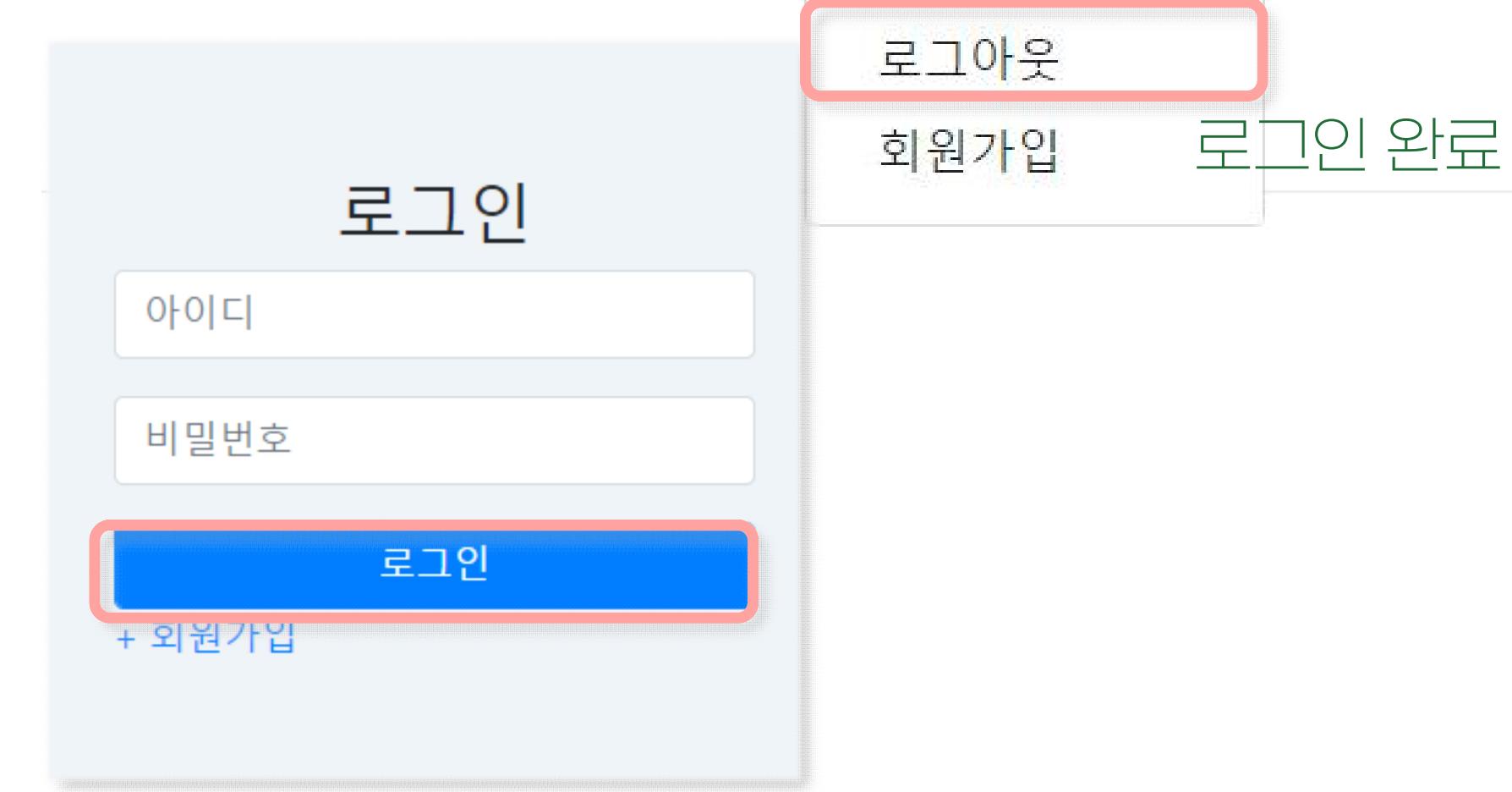
```
HttpSession session = req.getSession();
Member login = mservice.login(member);
if (login == null) {
    session.setAttribute("member", null);
} else {
    session.setAttribute("member", login);
    req.setAttribute("member", login);
```

### co.kr.yj.service.MemberService

```
@Inject MemberDAO mdao;
return mdao.login(member);
```

### co.kr.yj.dao.MemberDAO

```
@Inject SqlSession sqlSession;
return sqlSession.selectOne("memberMapper.login", member);
```



### resources/mepper/MemberMapper.xml

```
<select id="login" resultType="kr.co.yj.vo.Member">
    Select * From member
    Where id = #{id} And pw = #{pw}
</select>
```

# 기술 상세

## 회원정보 수정

### views/Member/Home.jsp

로그인 시 회원정보란, “회원정보 수정” 버튼 클릭 수정 페이지 이동  
입력란에 수정 정보 입력, 비밀번호 일치 확인 후 “수정” 버튼 클릭

### co.kr.yj.controller.MemberController

```
@RequestMapping(value = "/memberUpdateView", method =  
RequestMethod.GET)  
HttpSession session = req.getSession();  
Member member = (Member)session.getAttribute("member");  
model.addAttribute("member", member);  
  
@RequestMapping(value = "/memberUpdate", method =  
RequestMethod.POST)  
mservice.memberUpdate(member);  
session.invalidate();
```

### co.kr.yj.service.MemberService

```
@Inject MemberDAO mdao;  
mdao.memberUpdate(member);
```



## 회원정보 수정

아이디:  
han

성명:  
한이진

비밀번호:  
....

비밀번호 확인:  
비밀번호를 재입력하세요.

전화번호:  
01099827993

주소:  
울산광역시 울주군 서생면 해맞이로 1375-24

취소

회원정보 수정

### co.kr.yj.dao.MemberDAO

```
@Inject SqlSession sqlSession;  
sqlSession.update("memberMapper.memberUpdate",  
member);
```

### resources/mepper/MemberMapper.xml

```
<update id="memberUpdate" >  
    Update member Set  
        pw = #{pw}, name=#{name},  
        phone=#{phone}, address=#{address}  
    Where id=#{id}  
</update>
```

### 게시판

제목/내용/작성자/제목+내용 분류  
키워드로 게시글 검색 가능

----- 제목 ▼

번호 제목 작성자 작성일

localhost:8000/board/list?&searchType=t&keyword=수정

293	dd	ddd	2021-07-26 13:35:04.0
292			2021-07-25 11:55:01.0
289	○○○	○○○○	2021-07-25 11:50:31.0

localhost:8000/board/readView?bno=254&page=1&perPageNum=10&searchType=t&keyword=수정

255	내용수정되랏!	한이라	2021-07-24 15:17:33.0
254	수정수정	gksdlwls123	2021-07-24 15:17:33.0
253	ssssss	gksdlwls123	2021-07-24 15:17:33.0
252	내용수정되랏!	한이라	2021-07-24 15:17:33.0
251	수정수정	gksdlwls123	2021-07-24 15:17:33.0

글쓰기

게시글 작성 페이지로 이동

1 2 3 4 5 6 7 8 9 10 다음

게시글 페이지 처리

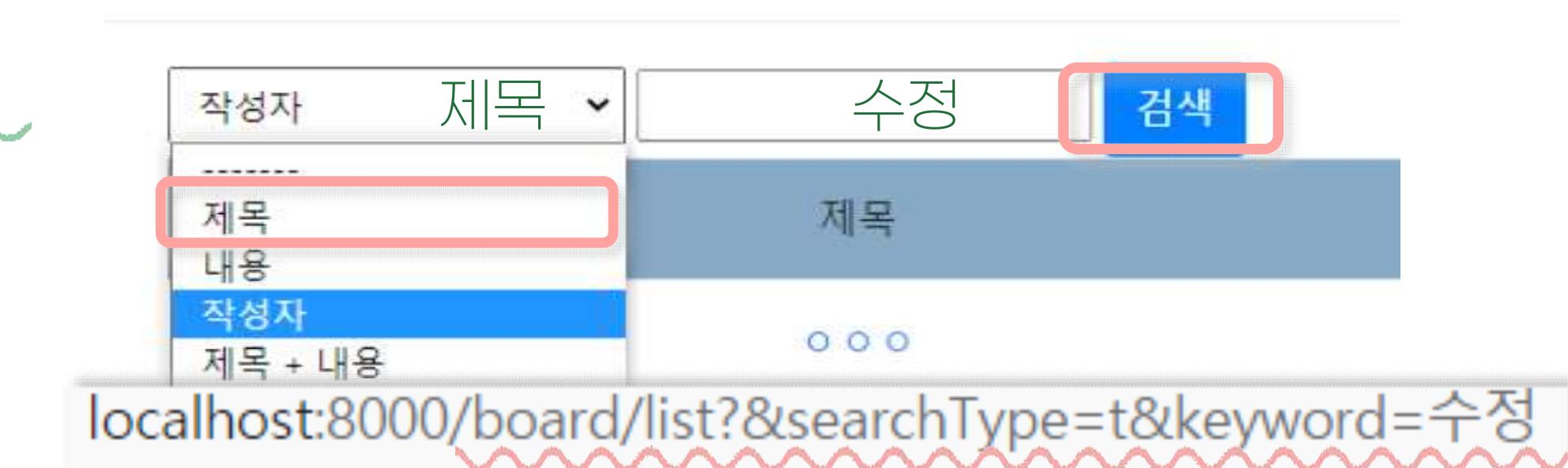
# 기술 상세

## 게시판 검색

### views/Board/list.jsp

```
<select name="searchType" class="custom-select-sm col-sm-2">
    <option selected>-----</option>
    <option value="t" <c:out value="${scri.searchType eq 't' ? 'selected' : ''}" />>제목
    <option value="c" <c:out value="${scri.searchType eq 'c' ? 'selected' : ''}" />>내용
    <option value="w" <c:out value="${scri.searchType eq 'w' ? 'selected' : ''}" />>작성자
    <option value="tc" <c:out value="${scri.searchType eq 'tc' ? 'selected' : ''}" />>제목 + 내용
</select>
<input type="text" name="keyword" id="keywordInput" value="${scri.keyword}"/>
    <button id="searchBtn" type="button" class="btn btn-primary">검색</button>
```

```
<script type="text/javascript">
$(function(){
    $('#searchBtn').click(function(){
        self.location = "list?" + ${pageMaker.makeQuery(1)}
        + "&searchType=" + $("select option:selected").val()
        + "&keyword=" + encodeURIComponent($('#keywordInput').val());
    });
}</script>
```



### resources/mepper/BoardMapper.xml

```
<sql id="search"><if test="searchType != null">
    <if test="searchType == 't'.toString()">AND TITLE LIKE
        CONCAT('%',#{keyword},'%')</if>
    <if test="searchType == 'c'.toString()">AND CONTENT LIKE
        CONCAT('%',#{keyword},'%')</if>
    <if test="searchType == 'w'.toString()">AND USERID LIKE
        CONCAT('%',#{keyword},'%')</if>
    <if test="searchType == 'tc'.toString()">AND (TITLE LIKE CONCAT
        ('%',#{keyword},'%')) or (CONTENT LIKE CONCAT('%',#{keyword},'%'))</if>
</if></sql>
```

# 기술 상세

## 게시판 목록

### views/Board/list.jsp

상단바에서 “MyFarmStory” 클릭 시 페이지 이동

### co.kr.yj.controller.BoardController

```
@RequestMapping(value = "/list", method = RequestMethod.GET)
model.addAttribute("list", bservice.selectList(scri));
//페이지 기능 생성
PageMaker page = new PageMaker();
page.setCri(scri);
page.setTotalCount(bservice.listCount(scri));
model.addAttribute("page", page);

return "board/list";
```

### co.kr.yj.service.BoardService

```
@Inject BoardDAO bDao;
List<Board> selectList(SearchCriteria scri){
    return bDao.selectList(scri); }
```

번호	제목	작성자	작성일
254	수정수정	gksdlwls123	2021-07-24 15:17:33.0
252	내용수정되랏!	한이라	2021-07-24 15:17:33.0
251	수정수정	gksdlwls123	2021-07-24 15:17:33.0
249	내용수정되랏!	한이라	2021-07-24 15:17:33.0
248	수정수정	gksdlwls123	2021-07-24 15:17:33.0
246	내용수정되랏!	한이라	2021-07-24 15:17:33.0
245	수정수정	gksdlwls123	2021-07-24 15:17:33.0
243	내용수정되랏!	한이라	2021-07-24 15:17:33.0
242	수정수정	gksdlwls123	2021-07-24 15:17:33.0
240	내용수정되랏!	한이라	2021-07-24 15:17:33.0

글쓰기

1 2 3 4 5 6 7 8 9 10 다음

### co.kr.yj.dao.BoardDAO

```
@Inject SqlSession sqlSession;
return sqlSession.selectList("boardMapper.select_list",scri);
```

### resources/mepper/BoardMapper.xml

```
<select id="select_list" resultType="kr.co.yj.vo.Board"
        parameterType="kr.co.yj.vo.SearchCriteria">
    SELECT * FROM BOARD
    <where>
        <include refid="search"></include> 중복쿼리 (키워드검색)대체
    </where>
        order by bno desc limit #{rowStart},#{rowEnd}; 최신순
</select>
```

# 기술 상세

게시글 작성

## views/Board/writeView.jsp

게시판 목록 하단 “글쓰기” 버튼 클릭 시 페이지 이동,  
정보 입력 후 “작성완료” 버튼 클릭

## co.kr.yj.controller.BoardController

```
@RequestMapping(value= "/board/write")
public String write(Board board) throws Exception{
    bservice.write(board);
    return "redirect:/board/list";
```

## co.kr.yj.service.BoardService

```
@Inject BoardDAO bDao;
bDao.write(board);
```

## co.kr.yj.dao.BoardDAO

```
@Inject SqlSession sqlSession;
sqlSession.insert("boardMapper.insert",board);
```

HOME My Farm Stroy 생장일지 회원관리 ▾

## 게시판

로그인 완료 시 게시글 작성 가능

로그인 후에 작성하실 수 있습니다.

게시판 글 작성

제목 #{{title}}

작성자 #{{userid}}

내용 #{{content}}

파일 선택 선택된 파일 없음

작성완료

## resources/mepper/BoardMapper.xml

```
<insert id="insert" parameterType="kr.co.yj.vo.Board">
```

```
Insert into board(title, content, userid, redate, available)
values (#{title},#{content},#{userid},now(),1)
</insert>
```

# 기술 상세

## 게시글 읽기

### views/Board/list.jsp

해당 게시글 클릭 시 “board/readView?bno=” url로 이동

### co.kr.yj.controller.BoardController

```
@RequestMapping(value="/readView", method =  
                RequestMethod.GET)  
model.addAttribute("read", bservice.read(board.getBno()));  
//해당 게시글에 작성된 댓글  
model.addAttribute("replyList",  
                rservice.selectReplyList(board.getBno()));  
model.addAttribute("scri", scri);  
return "board/readView";
```

### co.kr.yj.service.BoardService

```
@Inject BoardDAO bDao;  
return bDao . read(bno);
```

### co.kr.yj.dao.BoardDAO

```
@Inject SqlSession sqlSession;  
return sqlSession.selectOne("boardMapper.read", bno);
```

The screenshot shows a web application interface for reading a post. At the top, there's a header bar with the title '게시글' (Post). Below it is a detailed view of a post:

글 제목	안녕하세요
작성자	gksdlwls123
작성일자	2021-07-25 11:50:17.0
내용	spring 게시판 구현!!

Below the post details are three buttons: '수정' (Edit), '삭제' (Delete), and '목록' (List). To the right of the post, there's a section for comments:

댓글	작성자	내용	수정	삭제
gksdlwls123	댓글 수정해볼까요?!		수정	삭제
gksdlwls123	댓글 삭제해볼까요?		수정	삭제

At the bottom right, there's a blue button labeled '작성완료' (Completed).

### resources/mepper/BoardMapper.xml

```
<select id="read" parameterType="int"  
        resultType="kr.co.yj.vo.Board">  
    select bno, title, content, userid, redate  
    from board  
    where bno = #{bno}  
</select>
```

# 기술 상세

## 게시글 수정

### views/Board/updateView.jsp

수정 페이지에서 정보 입력 후 “수정” 버튼 클릭

### co.kr.yj.controller.BoardController

```
@RequestMapping(value="/updateView",method =  
    RequestMethod.GET)  
model.addAttribute("update", bservice.read(board.getBno()));  
model.addAttribute("scri",scri);  
return "board/updateView";
```

```
@RequestMapping(value="/update", method =  
    RequestMethod.POST)  
bservice.update(board);  
return "redirect:/board/list";
```

### co.kr.yj.service.BoardService

```
@Inject BoardDAO bDao;  
bDao.update(board);
```



### co.kr.yj.dao.BoardDAO

```
@Inject SqlSession sqlSession;  
sqlSession.update("boardMapper.update", board);
```

### resources/mepper/BoardMapper.xml

```
<update id="update" parameterType="kr.co.yj.vo.Board">  
    update board set title = #{title}, content = #{content},  
    redate = now()  
    where bno = #{bno}  
</update>
```

# 기술 상세

게시글 삭제

## 게시판

게시글	
글 제목	안녕하세요
작성자	gksdlwls123
작성일자	2021-07-25 11:50:17.0
내용	spring 게시판 구현!!
수정	삭제
목록	
댓글	

## views/Board/readView.jsp

해당 게시글 작성자 일때 “삭제” 버튼 생성, 클릭

## co.kr.yj.controller.BoardController

```
@RequestMapping(value="/delete")//, method =  
    RequestMethod.POST)  
public String delete(Board board) throws Exception{  
    logger.info("delete");  
    bservice.delete(board.getBno());  
    return "redirect:/board/list";  
}
```

## co.kr.yj.service.BoardService

```
@Inject BoardDAO bDao;  
bDao.delete(board);
```

## co.kr.yj.dao.BoardDAO

```
@Inject SqlSession sqlSession;  
sqlSession.delete("boardMapper.delete", bno);
```

## resources/mepper/BoardMapper.xml

```
<delete id="delete" parameterType="int">  
    delete from board  
    where bno = #{bno}  
</delete>
```

# 기술 상세

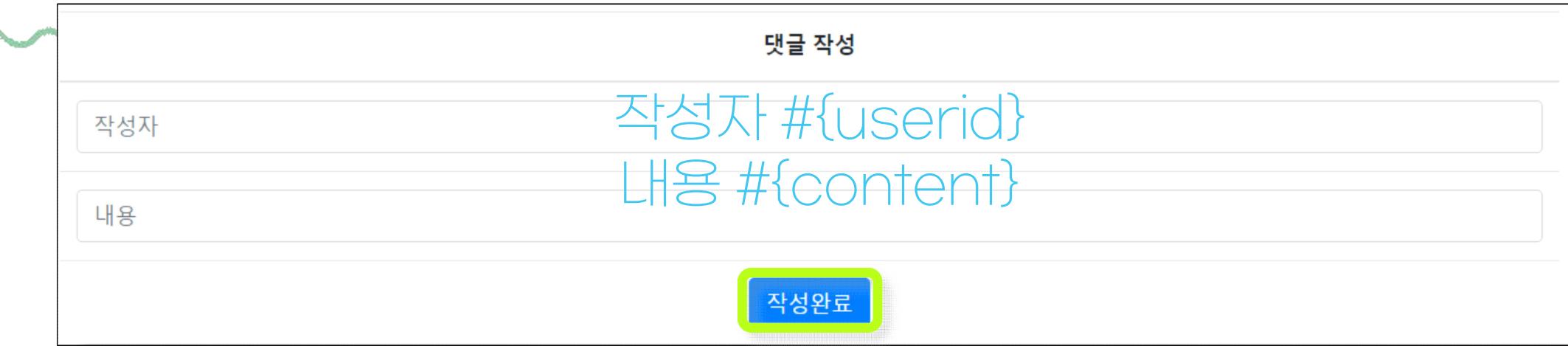
## 댓글 작성 / 수정 / 삭제

### views/Board/readView.jsp

댓글 입력란 작성 후 “작성완료” 버튼 클릭

### co.kr.yj.controller.ReplyController

```
@Inject ReplyService rservice;  
  
@RequestMapping(value= "/replyWrite",  
                 method=RequestMethod.POST)  
public String write(Reply reply, SearchCriteria scri, RedirectAttributes rttr)  
throws Exception{  
    rservice.write(reply);  
    rttr.addAttribute("bno", reply.getBno());  
    rttr.addAttribute("page", scri.getPage());  
    rttr.addAttribute("perPageNum",  
                     scri.getPerPageNum());  
    rttr.addAttribute("searchType", scri.getSearchType());  
    rttr.addAttribute("keyword", scri.getKeyword());  
  
    return "redirect:/board/readView";  
}
```



### co.kr.yj.service.ReplyService

```
@Inject ReplyDAO rDao;  
rDao.writeReply(reply);
```

### co.kr.yj.dao.ReplyDAO

```
@Inject SqlSession sqlSession;  
sqlSession.insert("replyMapper.writeReply", reply);
```

### resources/mapper/ReplyMapper.xml

```
<insert id="writeReply" parameterType="kr.co.yj.vo.Reply">  
    Insert Into Reply(bno, content, userid, date)  
    Values ( #{bno}, #{content}, #{userid}, now() )  
</insert>
```

# 기술 상세

## 댓글 작성 / 수정 / 삭제

댓글 수정

내용 댓글 수정!!!!

수정 취소 댓글 수정

댓글

gksdlwls123 댓글 수정!!!!

gksdlwls123 댓글 삭제해볼까요?!

수정 삭제

게시판

localhost:8000 내용:  
삭제하시겠습니까?

확인 취소

게시글

글 제목 안녕하세요 댓글 삭제

작성자 gksdlwls123

작성일자 2021-07-25 11:50:17.0

내용 spring 게시판 구현!!

수정 삭제 목록

댓글

gksdlwls123 댓글 삭제해볼까요?!

수정 삭제

댓글

gksdlwls123 댓글 수정해볼까요?!

수정 삭제

댓글 작성

성자

내용

작성완료

# 기술 상세

## 생장일지 작성/ 수정 /삭제

han님 안녕하세요 :)

### 21주차 생장일지

#### 게시글

주차	21
작성일자	2021
생장길이	15
화방높이	23
줄기굵기	12
잎길이	5
잎수	23
착과군	5
개화군	7
열매수	20

수정 삭제 목록

### 생장일지

생장일지 작성

생장길이

생장길이 #{length}

화방높이

화방길이 #{height}

줄기굵기

줄기굵기 #{width}

잎 길이

잎길이 #{leaf\_len}

잎수

잎수 #{leaf}

착과군

착과군 #{Fruit\_group}

개화군

개화군 #{flower}

열매수

열매수 #{fruit}

주차

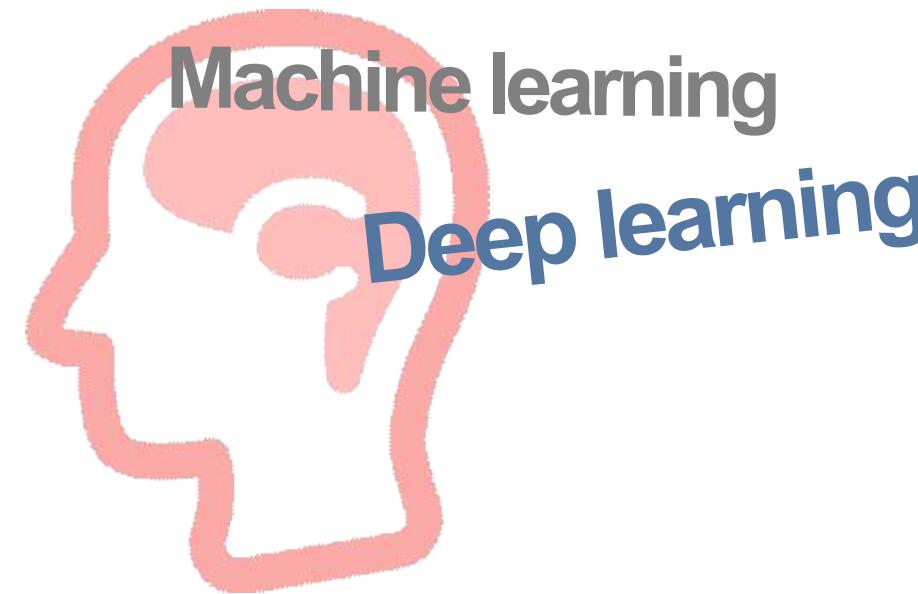
주차 #{week}

파일 선택 선택된 파일 없음

작성완료

# 개선 및 발전

사용자에게 양질의 데이터를 제공하기 위한  
우수농가 사례 데이터 수집 및 분석



농작물 생육정보  
(생장길이, 화방길이, 줄기 굵기, 잎수,  
착과군 등 \_ 토마토 기준)  
를 통해 열매수(수확량) 예측하는  
머신러닝 모델링 시도

## Smart My Farm ver.1.2을 위한 준비

### 공공데이터 포털(Open API)

농촌진흥청에서 공개한 스마트팜 우수농가 데이터 수집

### Python Data Analysis Library

Open API를 통해 수집한 데이터  
Python의 Pandas 라이브러리를 통해 데이터 분석  
결측치값을 평균값으로 대체하여 제거, 모델링을 위한  
데이터 정제 과정 수행

### 사이킷런(sklearn)

python을 대표하는 머신러닝 라이브러리  
선형 회귀분석(LinearRegression), K-최근접 이웃(KNeighborsRegressor)  
의사 결정 나무(DecisionTreeRegressor),  
결정트리 분류기(DecisionTreeClassifier)  
각 모델의 예측값 비교하여 분석 시도

# 데이터 수집 Open API

```
def api_read_growth(farmNum):
    url='http://apis.data.go.kr/1390000/SmartFarmdata/grwdatarqst'
    queryParams = '?' + urlencode(
        {'quote_plus('ServiceKey')': my_servicekey, quote_plus('serviceKey'): decoding, quote_plus('pageSize'): '15',
         quote_plus('pageNo'): '2', quote_plus('searchFrmhsCode'): farmNum})
    response = requests.get(url + queryParams).text.encode('utf-8')
    xmlobj = bs4.BeautifulSoup(response, 'lxml-xml')
    rows = xmlobj.find_all('item')
    rowList = []
    nameList = []
    columnList = []
    rowsLen = len(rows)
    for i in range(0, rowsLen):
        columns = rows[i].find_all()
        columnsLen = len(columns)
        for j in range(0, columnsLen):
            if i == 0:
                nameList.append(columns[j].name)
            eachColumn = columns[j].text
            columnList.append(eachColumn)
        rowList.append(columnList)
        columnList = [] # 다음 row의 값을 넣기 위해 비워준다

    result = pd.DataFrame(rowList, columns=nameList)
    print(result)
    return result
```

Open API를 통해  
농촌진흥원에서 제공하는  
우수한 스마트팜의 생육 정보 수집



# 데이터 수집 Open API

```
def api_read_output(farmNum):
    url = 'http://apis.data.go.kr/1390000/SmartFarmdata/prddatarqst'
    queryParams = '?' + urlencode(
        {'quote_plus('ServiceKey')': my_servicekey, quote_plus('serviceKey'): decoding, quote_plus('pageSize'): '15',
         quote_plus('pageNo'): '2', quote_plus('searchFrmhsCode'): farmNum})
    response = requests.get(url + queryParams).text.encode('utf-8')
    xmlobj = bs4.BeautifulSoup(response, 'lxml-xml')
    rows = xmlobj.find_all('item')
    rowList = []
    nameList = []
    columnList = []
    rowsLen = len(rows)
    for i in range(0, rowsLen):
        columns = rows[i].find_all()
        columnsLen = len(columns)
        for j in range(0, columnsLen):
            if i == 0:
                nameList.append(columns[j].name)
            eachColumn = columns[j].text
            columnList.append(eachColumn)
        rowList.append(columnList)
        columnList = [] # 다음 row의 값을 넣기 위해 비워준다

    result = pd.DataFrame(rowList, columns=nameList)
    print(result)
    return result
```

Open API를 통해  
우수한 스마트팜의 수확량,  
수확시기 데이터 수집



# 데이터 분석 Pandas



수집된 데이터, 분석에  
필요한 독립변수와 종속 변수로  
이루어진 데이터프레임으로 변환  
후 CSV 파일로 저장

```
def make_dataframe(result,start,finsh):
    input = result[['frmWeek', 'grwtLt', 'fcluHg', 'stemThck', 'lefLt', 'lefCunt', 'frtstGrupp', 'flanGrupp', 'hvstCo',
                   'frtstCo', ]].to_numpy()
    input=input[start:finsh]
    names = ['주차', '생장길이', '화방높이', '줄기굵기', '잎길이(엽장)', '잎수', '착과군', '개화군', '수확수', '열매수']
    df = pd.DataFrame(input, columns=names)
    return df

def save_to_csv(dataFrame,fileName):
    dataFrame.to_csv(fileName,encoding="UTF-8")
```

# 데이터 분석 Pandas

```
tomato=pd.read_csv('./data_csv/result.csv')
print(tomato.head())
print(tomato.info())
print(tomato.describe())# 데이터에 0값이 있음을 확인
print('-----데이터 결측값 평균으로 대체하기 위한 처리과정-----')
print('1. 0 값 NaN 처리')
tomato=tomato.replace(0,np.NaN)# 0값을 NaN으로 교체
print(tomato)
print(tomato.info())
print(tomato.describe())
print('2. 결측값 평균값 대체')
tomato.fillna(tomato.mean(),inplace=True) #fillna()로 결측값 평균값으로 대체
print(tomato)
print(tomato.describe())
tomato.to_csv('./data_csv/tomato.csv',encoding='UTF_8')
```

수집된 데이터에 결측값 존재  
결측값 평균값으로 대체하여 제거

# 데이터 분석 Pandas

## 2. 결측값 평균값 대체

	Unnamed: 0	주차	생장길이	...	개화군	수확수	열매수
0	6.57485	40.0	30.250000	...	5.074167	2.568627	9.708333
1	1.00000	41.0	20.708333	...	5.668333	2.568627	11.041667
2	6.57485	42.0	14.300000	...	6.700000	2.568627	14.700000
3	1.00000	43.0	18.100000	...	7.300000	2.568627	16.800000
4	6.57485	43.0	18.750000	...	7.125000	2.568627	15.750000
..	..	..	..	..	..	..	..
190	2.00000	49.0	20.800000	...	3.954167	2.568627	9.000000
191	6.57485	43.0	15.925000	...	6.925000	1.000000	13.750000
192	1.00000	44.0	28.075000	...	7.400000	2.250000	13.250000
193	6.57485	42.0	15.150000	...	3.583333	2.568627	7.750000
194	1.00000	43.0	14.000000	...	4.156250	2.568627	9.750000

[195 rows x 11 columns]

	Unnamed: 0	주차	생장길이	...	개화군	수확수	열매수
count	195.000000	195.000000	195.000000	...	195.000000	195.000000	195.000000
mean	6.574850	20.758974	19.478893	...	12.512355	2.568627	18.376717
std	3.898727	18.624016	4.422946	...	4.240176	0.525514	4.566420
min	1.000000	1.000000	9.333333	...	2.433333	0.500000	5.500000
25%	3.000000	6.000000	16.390278	...	9.941190	2.568627	15.750000
50%	6.574850	11.000000	19.478893	...	12.512355	2.568627	18.250000
75%	9.500000	42.000000	22.291667	...	15.238889	2.568627	21.208333
max	14.000000	52.000000	31.050000	...	23.875000	6.250000	30.500000

결측값 평균값으로 대체 후  
데이터 분석 결과



# 데이터셋 만들기



```
tomato = pd.read_csv('./data_csv/tomato.csv')
tomato_target = tomato['열매수'].to_numpy() #종속변수
# 독립변수
tomato_input = tomato[['생장길이', '화방높이', '줄기굵기', '잎길이(엽장)', '착과군', '개화군']].to_numpy()
## 훈련 세트와 테스트 세트로 나눔
trainin_input, test_input, train_target, test_target = train_test_split(tomato_input, tomato_target, random_state=42)

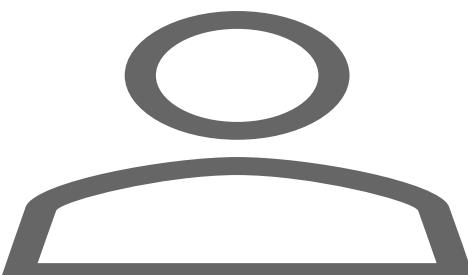
ss = StandardScaler()
ss.fit(trainin_input) #평균 0, 분산 1로 조정
train_scaled = ss.transform(trainin_input)
test_scaled = ss.transform(test_input)
print(train_scaled[:5])
train_target=round(train_target,3)
train_scaled=round(train_scaled,3)
test_scaled=round(test_scaled,3)
test_target=round(test_target,3)
print(train_scaled[:5])
print(train_target.dtype)
```

머신러닝 모델에 학습시킬  
데이터셋 정제

# 적합한 모델 찾기

머신러닝 모델 예측값 비교하여  
가장 알맞은 모델 찾기  
..진행중..

현재 머신러닝, 딥러닝  
학습중입니다.  
더욱 발전된 SmartMyFarm  
ver.1.2을 꼭 보여드리겠습니다



```
#선형 회귀분석
lr = LinearRegression()
lr.fit(train_scaled, train_target)
score = lr.score(test_scaled, test_target)
print('LinearRegression: ', score)
print(lr.coef_, lr.intercept_)

#K-최근접 이웃 알고리즘
knr = KNeighborsRegressor()
knr.fit(train_scaled, train_target)
score = knr.score(test_scaled, test_target)
print(score)

#결정트리나무(회귀)
dtr = DecisionTreeRegressor(max_depth=4)
dtr.fit(train_scaled, train_target)
score = dtr.score(test_scaled, test_target)
print(score)

#결정 트리 분류기
dt = DecisionTreeClassifier(random_state=42)
dt.fit(train_scaled, train_target)
print(dt.score(train_scaled, train_target))
print(dt.score(test_scaled, test_target))
```

# 감사합니다



더욱 발전될 Smart My Farm ver.1.2을  
기대해주세요 : )