Spring boot게시판

안 화 수

게시판

- 게시판 프로젝트의 주요 기능

1. Oracle MyBatis 연동 기능

2. 시퀀스(sequence) 기능

3. 댓글 게시판 기능

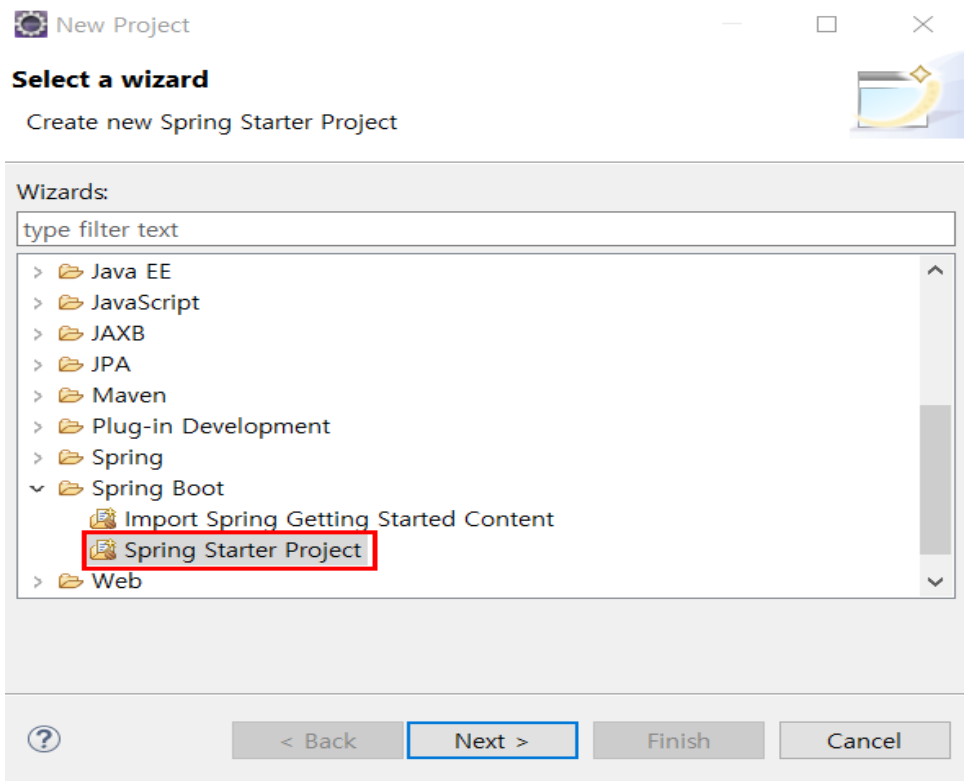
4. 페이징 처리

sbboard01 project를 maven으로 만들어 보자? (dao 클래스로 실행)

sbboard01 project 생성

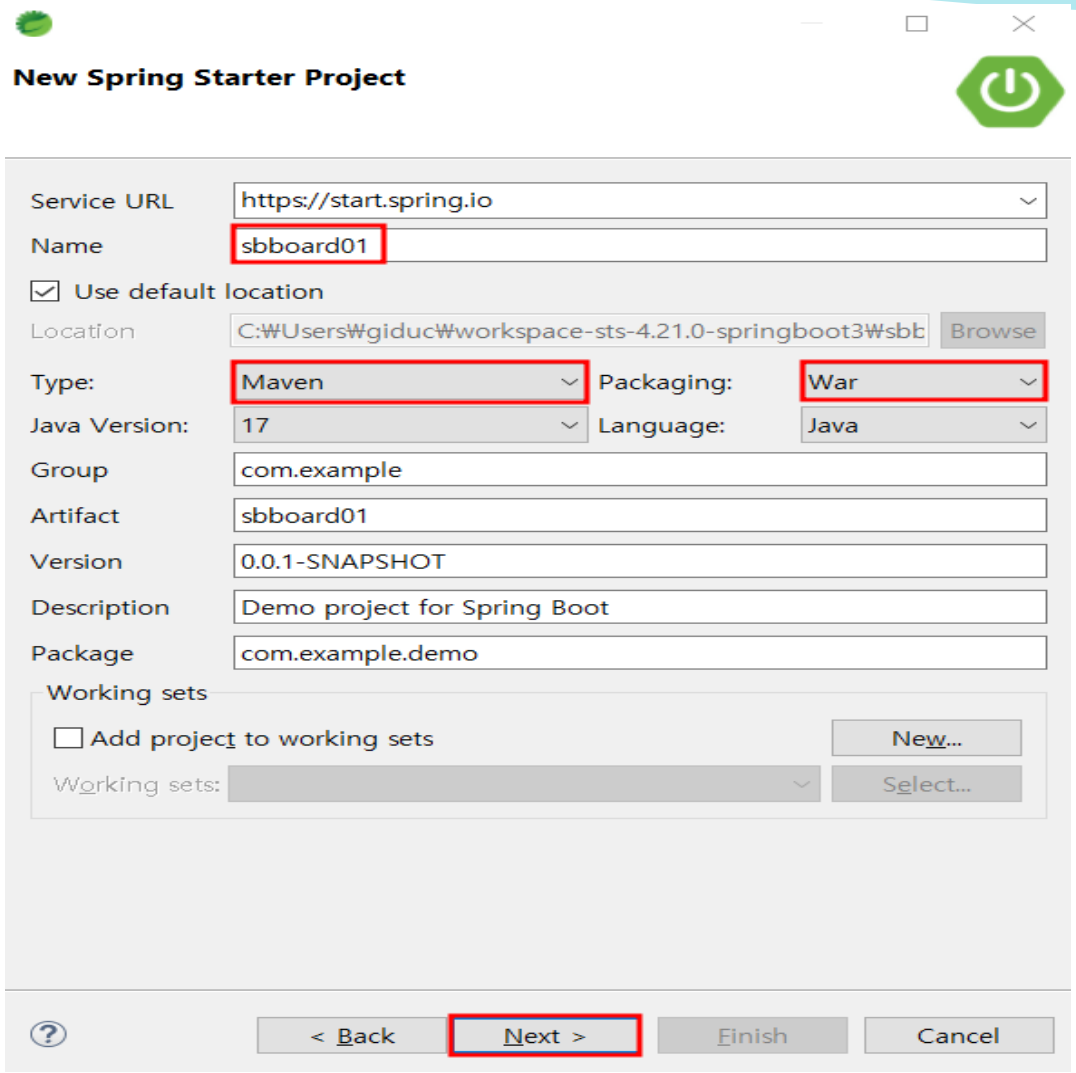
❖ sbboard01 project 생성

[File] – New - Project



sbboard01 project 생성

- sbboard01 project 생성
  - Name : sbboard01
  - Type : Maven, Gradle
  - Packaging : War, Jar



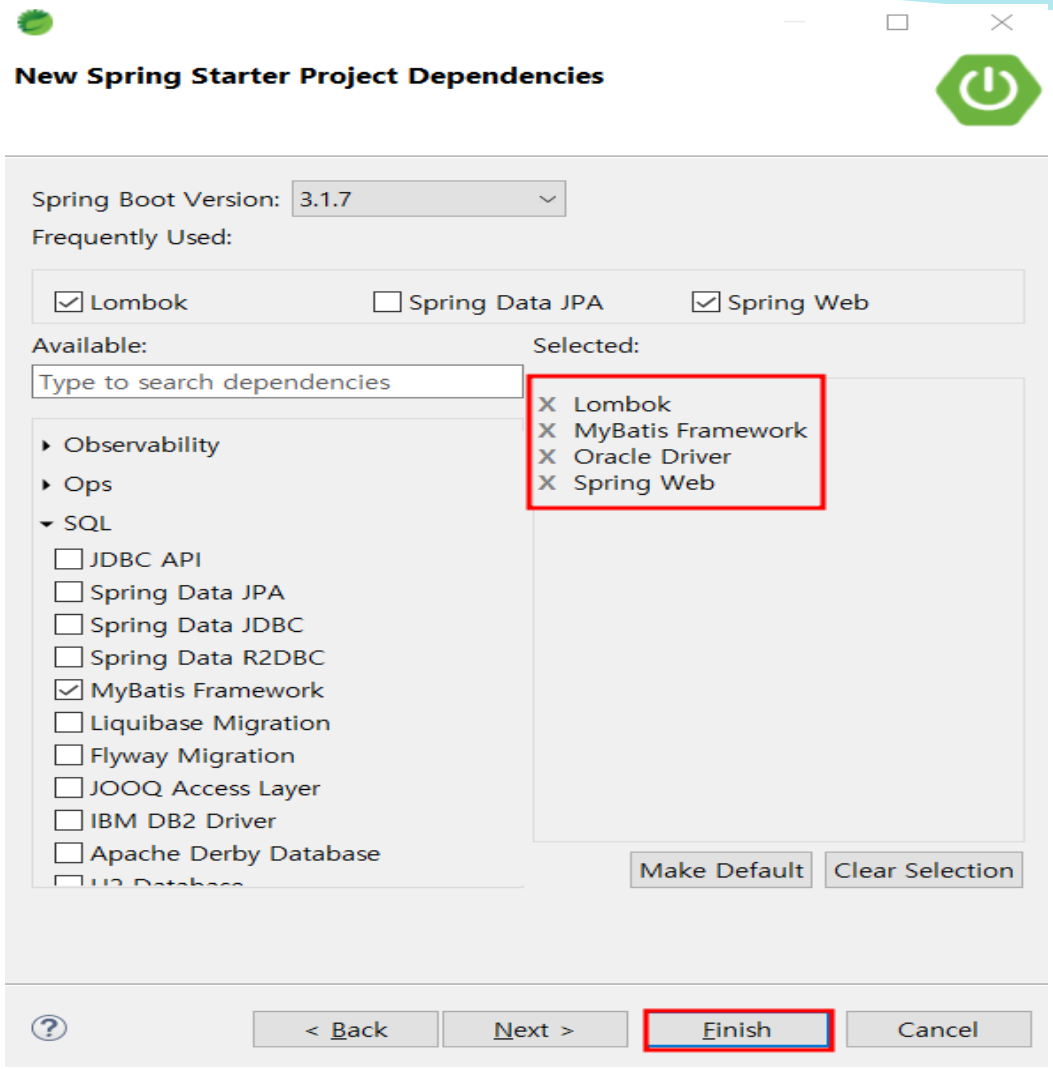
sbboard01 project 생성

- sbboard01 project 생성

- Web
  Spring Web 체크

- SQL
  MyBatis Framework 체크
  Oracle Driver 체크

- Developer Tools
  Lombok 체크



DB설계

❖ 테이블 및 시퀀스 생성

```
create table board53(
    board_num number(38) primary key
  , board_name varchar2(50) not null
  , board_pass varchar2(30) not null
  , board_subject varchar2(100) not null
  , board_content varchar2(4000) not null
  , board_re_ref number
  , board_re_lev number
  , board_re_seq number
  , board_readcount number
  , board_date date
);
create sequence board53_num_seq
                increment by 1 start with 1 nocache;
```

의존 라이브러리 추가

- **pom.xml**

  pom.xml 파일에 필요한 의존 라이브러리를 추가한다.

```
<dependencies>
        <!-- 내장 Tomcat실행시 jsp 파일을 사용하기 위한 의존 라이브러리 -->
        <dependency>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-jasper</artifactId>
                <scope>provided</scope>
        </dependency>
        <!-- jakarta  jstl -->
        <dependency>
                <groupId>org.glassfish.web</groupId>
                <artifactId>jakarta.servlet.jsp.jstl</artifactId>
                <version>3.0.0</version>
        </dependency>
</dependencies>
```

환경 설정파일 수정(1/2)

- **환경 설정 파일 수정**
  main / resources / **application.properties**

```
# port
server.port=80

# prefix and suffix
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

- webapp / **WEB-INF / views** 폴더 생성

환경 설정파일 수정(2/2)

- **환경 설정 파일 수정**
  main / resources / **application.properties**

```
# oracle
spring.datasource.hikari.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.hikari.jdbc-url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.hikari.username=spring
spring.datasource.hikari.password=spring123
```

DatabaseConfiguration 설정

- **DatabaseConfiguration설정 파일 생성 /main/java/com/example/demo/configuration – DatabaseConfiguration.java (1/2)**

```
@Configuration
@PropertySource("classpath:/application.properties")
public class DatabaseConfiguration {

        @Autowired
        private ApplicationContext applicationContext;
        @Bean
        @ConfigurationProperties(prefix = "spring.datasource.hikari")
        public HikariConfig hikariConfig() {
                return new HikariConfig();
        }
        @Bean
        @ConfigurationProperties(prefix = "mybatis.configuration")
        public org.apache.ibatis.session.Configuration mybatisConfig() {
                return new org.apache.ibatis.session.Configuration();
        }
```

DatabaseConfiguration 설정

- **DatabaseConfiguration설정 파일 생성**
  /main/java/com/example/demo/configuration – DatabaseConfiguration.java (2/2)

```
        @Bean
        public DataSource dataSource() {
                DataSource dataSource = new HikariDataSource(hikariConfig());
                return dataSource;
        }
        @Bean
        public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {
                SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();
                sqlSessionFactoryBean.setDataSource(dataSource);
                sqlSessionFactoryBean.setMapperLocations(
                        applicationContext.getResources("classpath:/mapper/*.xml"));
                sqlSessionFactoryBean.setConfiguration(mybatisConfig());
                sqlSessionFactoryBean.setTypeAliasesPackage("com.example.demo.model");
                return sqlSessionFactoryBean.getObject();
        }
        @Bean
        public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactory) {
                return new SqlSessionTemplate(sqlSessionFactory);
        }
}
```

DTO 생성

- **DTO 생성**
  main / java / com / example / demo / model – BoardBean.java

```
@Data
@Alias("board")
public class BoardBean {
        private int board_num;
        private String board_name;
        private String board_pass;
        private String board_subject;        //글제목
        private String board_content;        //글내용
        private int board_re_ref;            //글그룹번호
        private int board_re_lev;            //답변글 깊이
        private int board_re_seq;            //답변글 출력순서
        private int board_readcount;         //조회수
        private String board_date;           //등록날짜
}
```

Controller 생성

- Controller 생성
  /main/java/com/example/demo/controller – BoardController.java

```
@Controller
public class BoardController {

        @Autowired
        private BoardServiceImpl boardService;

        // 게시판 글쓰기 폼
        @RequestMapping(value = "/board_write.do")
        public String board_write() {
                return "board/board_write";
        }
        // 게시판 저장
        @RequestMapping(value = "/board_write_ok.do", method = RequestMethod.POST)
        public String board_write_ok(@ModelAttribute BoardBean board) throws Exception {
//      public String board_write_ok(@RequestParam HashMap board) throws Exception {
                boardService.insert(board);
                return "redirect:/board_list.do";    // 글 목록 요청
        }
```

Service 생성

- **Service 생성**
  /main/java/com/example/demo/service – BoardService.java

```
@Service
public class BoardServiceImpl {

        @Autowired
        private BoardDAOImpl boardDao;

        // 게시판 저장
        public void insert(BoardBean b) throws Exception {
                boardDao.insertBoard(b);
        }
        // 데이터 갯수
        public int getListCount() throws Exception {
                return boardDao.getListCount();
        }
        // 게시판 목록
        public List getBoardList(int page) throws Exception {
                return boardDao.getBoardList(page);
        }
```

Dao 생성

- **Dao 생성**
  /main/java/com/example/demo/dao – BoardDaoImpl.java

```
@Repository
public class BoardDAOImpl {

        @Autowired
        private SqlSession session;

        public void insertBoard(BoardBean board) throws Exception {
                session.insert("boardns.board_insert", board);
        }
        public int getListCount() throws Exception {
                return session.selectOne("boardns.board_count");
        }
        public List<BoardBean> getBoardList(int page) throws Exception {
                List<BoardBean> list = session.selectList("boardns.board_list", page);
                return list;
        }
```

mapper 생성

- **mapper 생성**

main / resources / mapper – board.xml

```
<mapper namespace="boardns">

        <!-- 게시판 저장 -->
        <insert id="board_insert" parameterType="board">
                insert into board53          (board_num,board_name,board_pass,board_subject,
                board_content,board_re_ref,board_re_lev,board_re_seq,board_readcount,board_date)
                values
                (board53_num_seq.nextval,#{board_name},#{board_pass},#{board_subject},
                #{board_content},board53_num_seq.nextval,0,0,0,SYSDATE)
        </insert>

        <!-- 게시판 총게시물 수 -->
        <select id="board_count" resultType="int">
                select count(*) from board53
        </select>
```

View 생성

- View 생성

webapp - index.jsp 생성

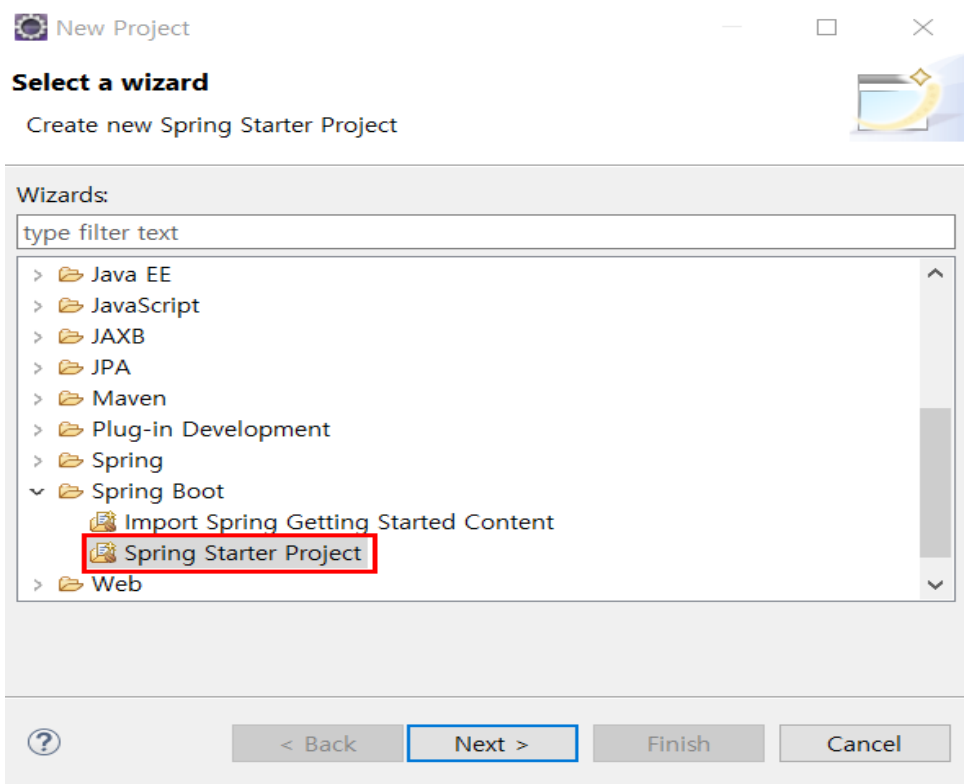
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

<script>
            location.href="boardform";
</script>

</body>
</html>
```

sbboard02 project를 gradle로 만들어 보자? (dao 클래스로 실행)

sbboard02 project 생성

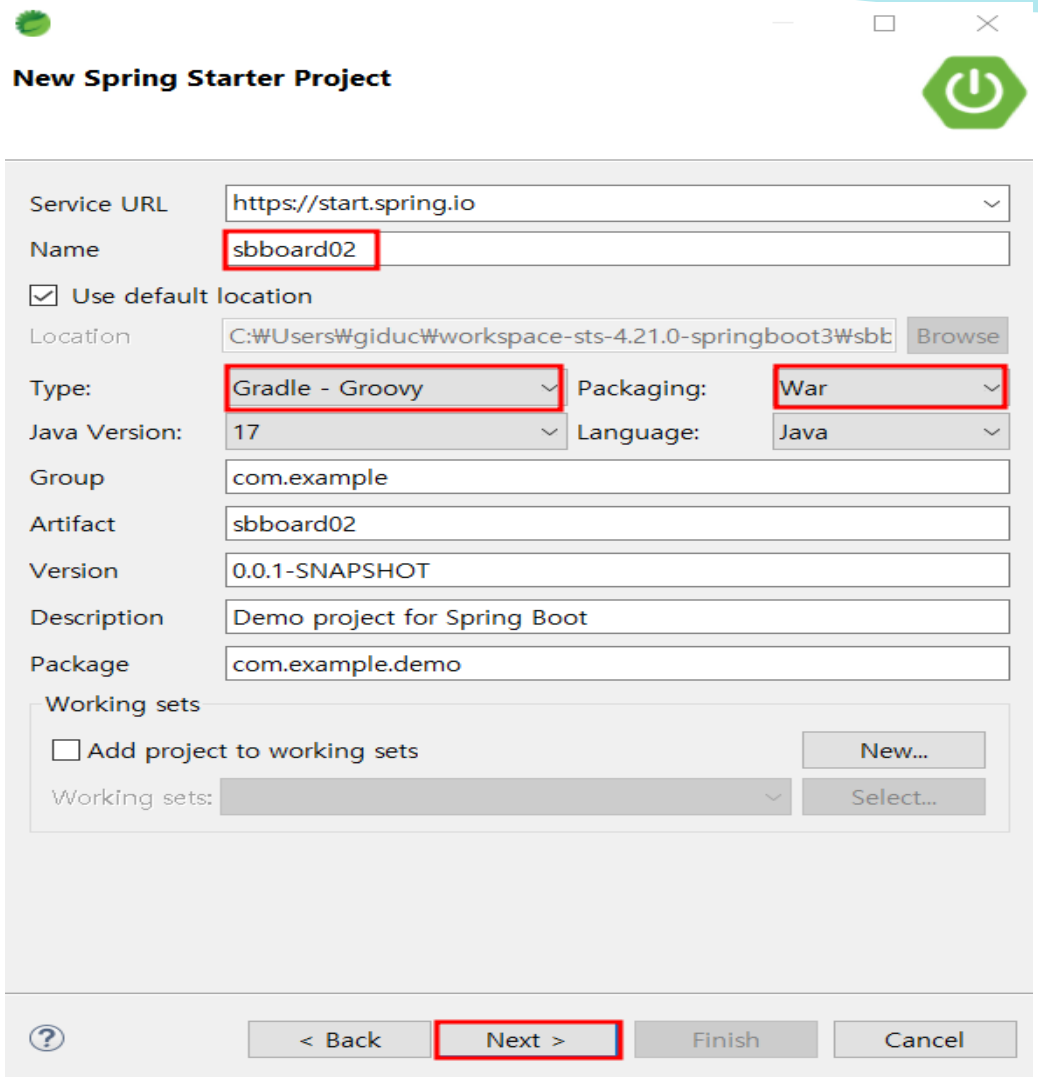
❖ sbboard02 project 생성

[File] – New - Project



sbboard02 project 생성

- sbboard02 project 생성
  - Name : sbboard02
  - Type : Maven, Gradle
  - Packaging : War, Jar



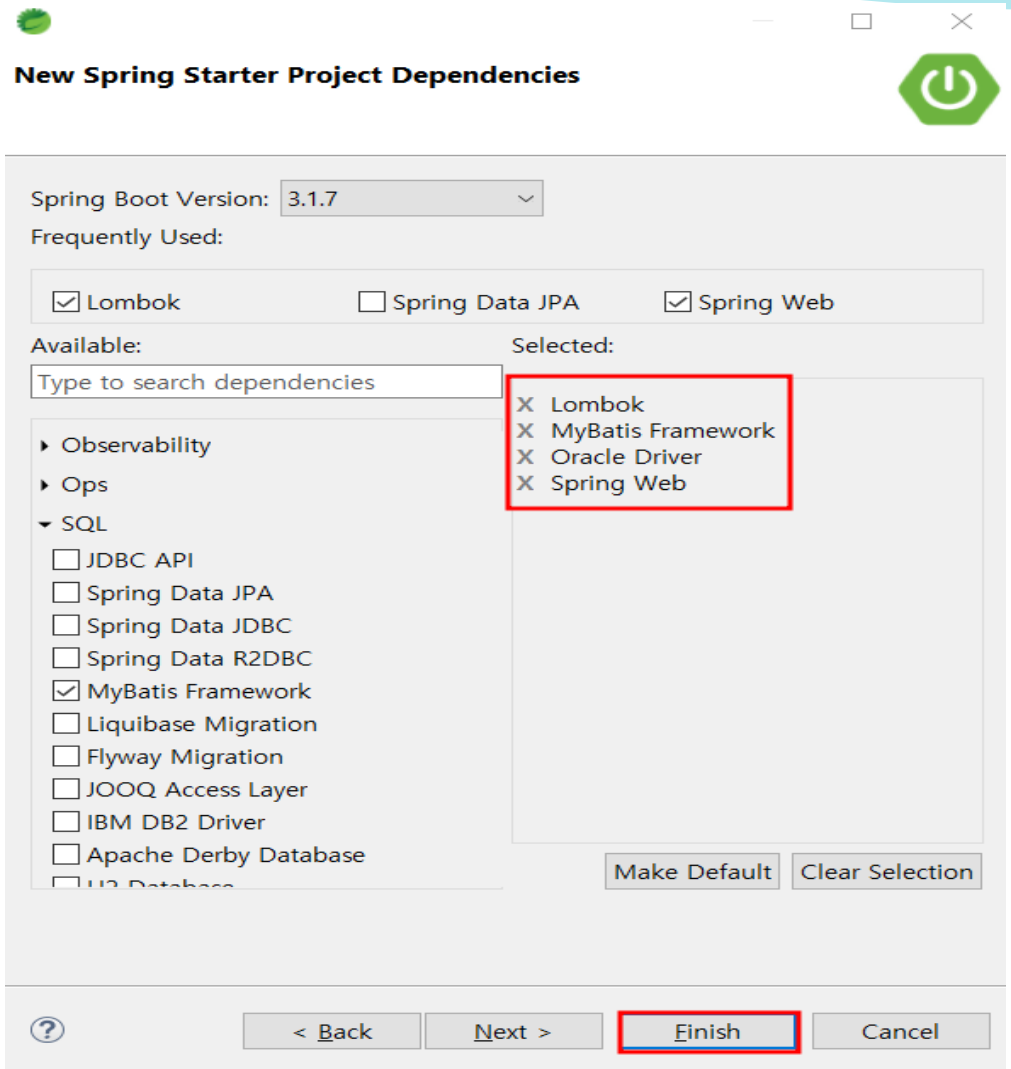
sbboard02 project 생성

- sbboard02 project 생성

- Web
  Spring Web 체크

- SQL
  MyBatis Framework 체크
  Oracle Driver 체크

- Developer Tools
  Lombok 체크



Gradle 환경설정

- Gradle 의 환경설정 파일에 의존 라이브러리 추가
  sbboard02 – build.gradle

```
dependencies {
        // jsp
        implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'

        // jakarta jstl
        implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl:3.0.0'

        implementation 'org.springframework.boot:spring-boot-starter-web'
        implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:3.0.3'
        compileOnly 'org.projectlombok:lombok'
        runtimeOnly 'com.oracle.database.jdbc:ojdbc11'
        annotationProcessor 'org.projectlombok:lombok'
        providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
        testImplementation 'org.springframework.boot:spring-boot-starter-test'
        testImplementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter-test:3.0.3'
}
```

sbboard02 실행

❖ sbboard02 실행

1. sbboard01 project 내용을 모두 sbboard02로 복사한다.

2. sbboard02 project를 내부 Tomcat으로 실행한다.

3. 웹브라우저로 접속한다.

http://localhost/board_write.do

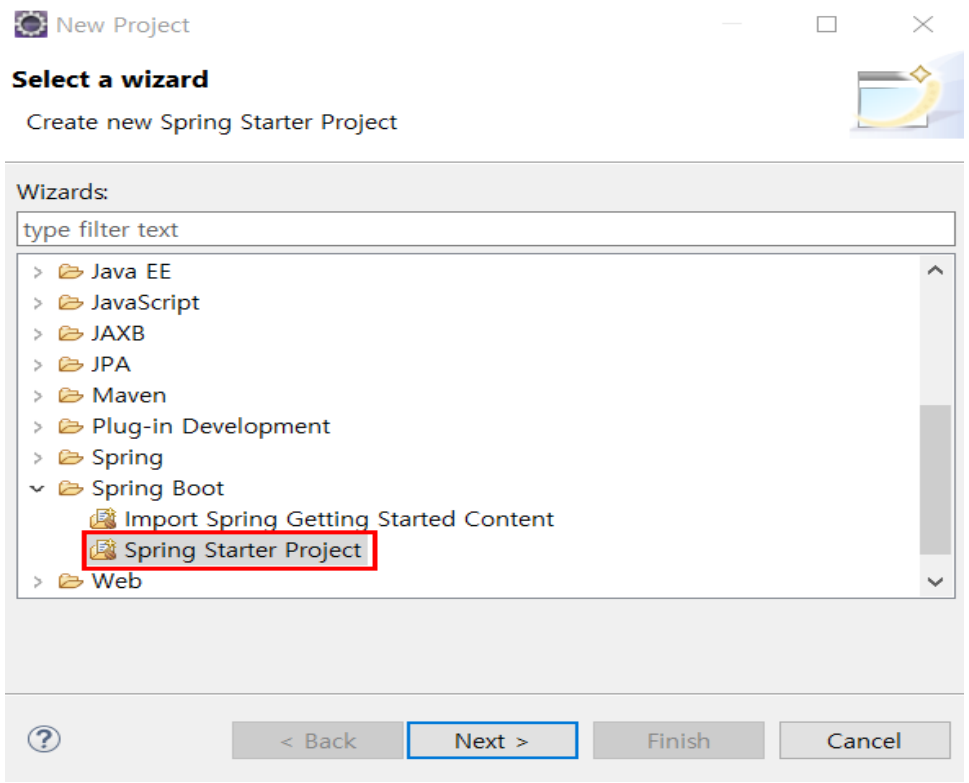
http://localhost/board_list.do

sbboard03 project를 gradle로 만들어 보자? (mapper interface로 실행)

sbboard03 project 생성

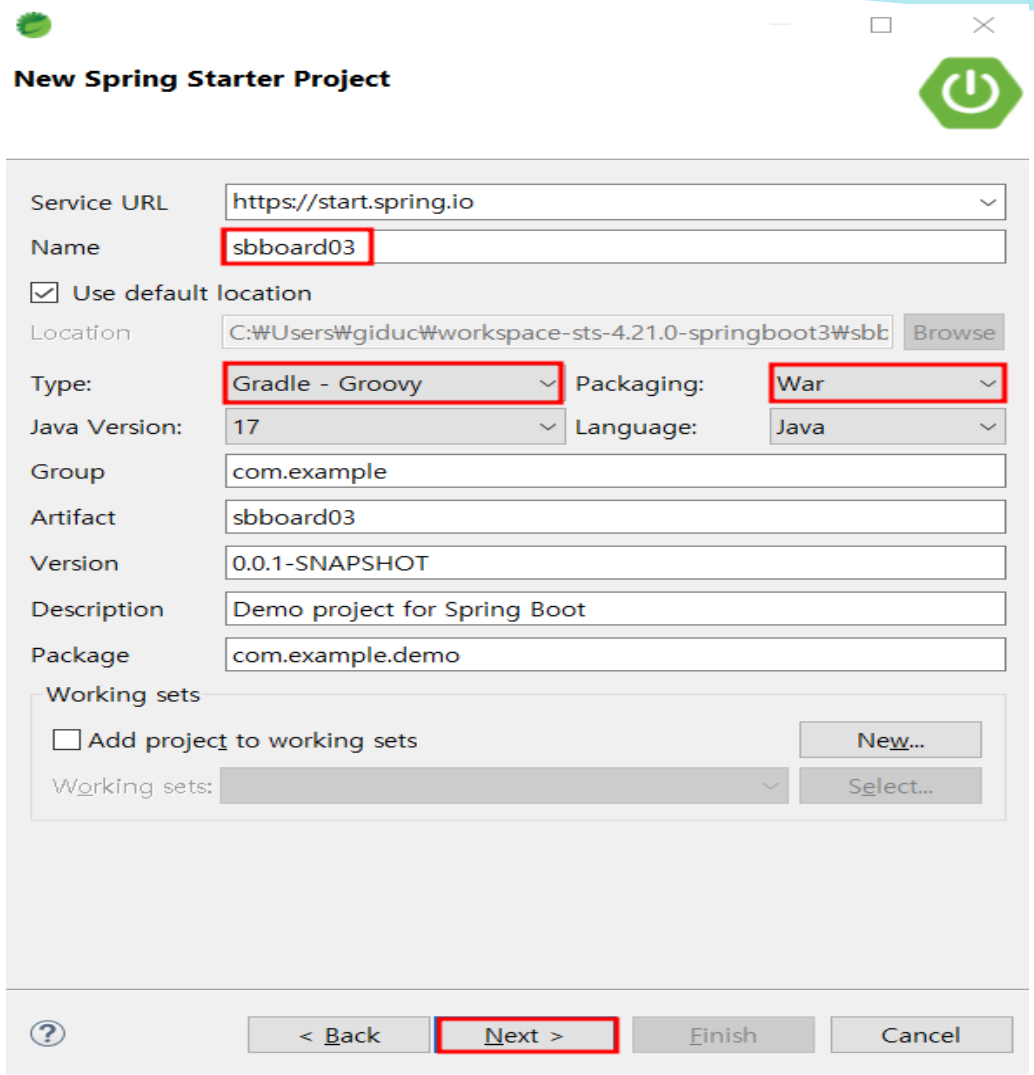
❖ sbboard03 project 생성

[File] – New - Project



sbboard03 project 생성

- sbboard03 project 생성
  - Name : sbboard03
  - Type : Maven, Gradle
  - Packaging : War, Jar



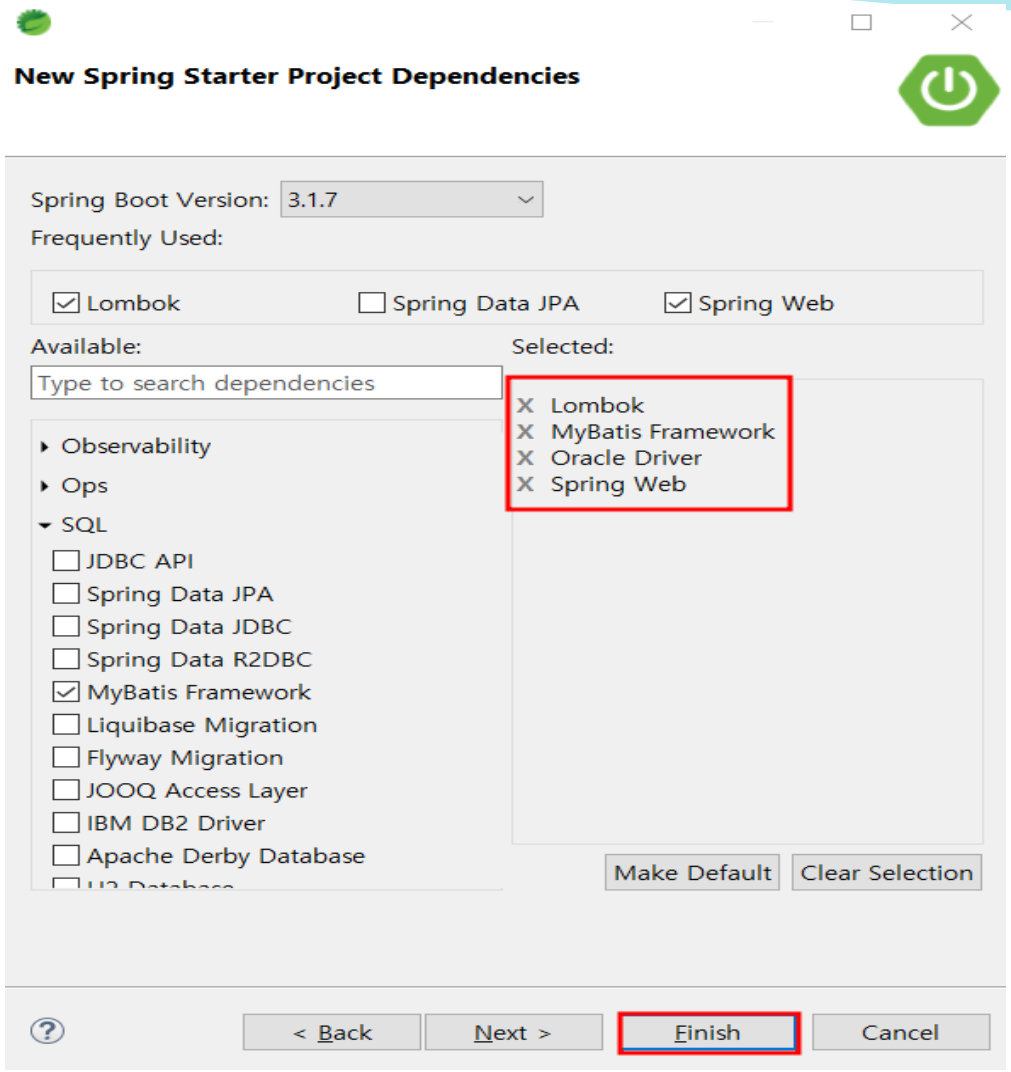
sbboard03 project 생성

- sbboard03 project 생성

- Web
  Spring Web 체크

- SQL
  MyBatis Framework 체크
  Oracle Driver 체크

- Developer Tools
  Lombok 체크



Gradle 환경설정

- Gradle 의 환경설정 파일에 의존 라이브러리 추가
  sbboard03 – build.gradle

```
dependencies {
        // jsp
        implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'

        // jakarta jstl
        implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl:3.0.0'

        implementation 'org.springframework.boot:spring-boot-starter-web'
        implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:3.0.3'
        compileOnly 'org.projectlombok:lombok'
        runtimeOnly 'com.oracle.database.jdbc:ojdbc11'
        annotationProcessor 'org.projectlombok:lombok'
        providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
        testImplementation 'org.springframework.boot:spring-boot-starter-test'
        testImplementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter-test:3.0.3'
}
```

sbboard03 실행

❖ sbboard03 실행

1. sbboard02 project 내용을 모두 sbboard03로 복사한다.

2. sbboard03 project를 내부 Tomcat으로 실행한다.

3. 웹브라우저로 접속한다.

http://localhost/board_write.do

http://localhost/board_list.do

Mapper interface

❖ Mapper interface 실행

1. Dao class 대신에 Mapper interface로 SQL문을 실행한다.

2. Mapper interface 위에는 @Mapper 어노테이션을 사용한다.

3. mapper.xml 파일의 namespace를 mapper interface의 path로 설정한다.

4. Mapper interface의 method명과 mapper.xml 파일의 id명을 일치 시킨다.

Mapper interface 생성

- **Mapper interface 생성**
  /main/java/com/example/demo/mapper – BoardDao.java

```
@Mapper
public interface BoardDao {
        public void insert(BoardBean board);
        public int getCount();
        public List<BoardBean> getBoardList(int page);
        public void hit(int board_num);
        public BoardBean board_cont(int board_num);
        public void edit(BoardBean board);
        public void delete(int board_num);
        public void refEdit(BoardBean board);
        public void reply(BoardBean board);
}
```

mapper 생성

- **mapper 생성**
  main / resources / mapper – board.xml

```
<mapper namespace="com.example.demo.mapper.BoardDao">

        <!-- 게시판 저장 -->
        <insert id="insert" parameterType="board">
                insert into board53 (board_num,board_name,board_pass,board_subject,
                board_content,board_re_ref,board_re_lev,board_re_seq,board_readcount,board_date)
                values
                (board53_num_seq.nextval,#{board_name},#{board_pass},#{board_subject},
                #{board_content},board53_num_seq.nextval,0,0,0,SYSDATE)
        </insert>

        <!-- 게시판 총게시물 수 -->
        <select id="getCount" resultType="int">
                select count(*) from board53
        </select>
```

board1 project
(검색 기능)

board1 project

- board1 project 주요기능

  - 검색기능 : 동적 SQL문

- board1 project import

board1 project를 import 해보자

DB설계

❖ 테이블 생성

```
create table board (
        num number primary key,          -- primary key
        writer varchar2(20) not null,    -- 작성자
        subject varchar2(50) not null,   -- 제목
        content varchar2(500) not null,  -- 내용
        email varchar2(30) ,             -- 이메일
        readcount number default 0,      -- 조회수
        passwd varchar2(12) not null,    -- 비밀번호
        ref number not null,             -- 답변글끼리 그룹
        re_step number not null,         -- 댓글 출력 순서
        re_level number not null,        -- 댓글 레벨
        ip varchar2(20) not null,        -- 작성자 ip
        reg_date date not null,          -- 작성일
        del char(1)                      -- 글삭제유무(y or n)
);
```

DTO 클래스

- DTO클래스 생성

```
@Data
@Alias("board")
public class Board {
        private int num;
        private String writer;
        private String subject;
        private String content;
        private String email;
        private int readcount;
        private String passwd;
        private int ref;
        private int re_step;
        private int re_level;
        private String ip;
        private Date reg_date;
        private String del;
        // page
        private int startRow;
        private int endRow;
        // 검색
        private String search;
        private String keyword;
}
```

동적 SQL문

❖ 동적 SQL문 : Board.xml (1/2)

```
<select id="getTotal" parameterType="board" resultType="int">
        select count(*) from board
        <where>
                <if test="keyword != null and search !='subcon'">
                        ${search} like '%'||#{keyword}||'%'
                </if>
                <if test="keyword != null and search=='subcon'">
                        subject like '%'||#{keyword}||'%' or
                        content like '%'||#{keyword}||'%'
                </if>
        </where>
</select>
```

동적 SQL문

- 동적 SQL문 : Board.xml (2/2)

```
<select id="list" parameterType="board" resultMap="boardResult">
        select * from (select a.*,rowNum rn from (
                select * from board
        <where>
                <if test="keyword != null and search!='subcon'">
                        ${search} like '%'||#{keyword}||'%'
                </if>
                <if test="keyword != null and search=='subcon'">
                        subject like '%'||#{keyword}||'%' or
                        content like '%'||#{keyword}||'%'
                </if>
        </where>
        order by ref desc,re_step) a )
        where rn between #{startRow} and #{endRow}
</select>
```

board2 project
(댓글 기능)

board2 project

- board2 project 주요기능

  - 부모 테이블에 댓글 기능

- board2 project import

board2 project를 import 해보자

DB설계

❖ 부모 테이블 생성

```
create table board (
        num number primary key,          -- primary key
        writer varchar2(20) not null,    -- 작성자
        subject varchar2(50) not null,   -- 제목
        content varchar2(500) not null,  -- 내용
        email varchar2(30) ,             -- 이메일
        readcount number default 0,      -- 조회수
        passwd varchar2(12) not null,    -- 비밀번호
        ref number not null,             -- 답변글끼리 그룹
        re_step number not null,         -- 댓글 출력 순서
        re_level number not null,        -- 댓글 레벨
        ip varchar2(20) not null,        -- 작성자 ip
        reg_date date not null,          -- 작성일
        del char(1)                      -- 글삭제유무(y or n)
);
```

DB설계

- **자식 테이블 생성**

```
create table replyBoard (
        rno number primary key,                          -- primary key
        bno number not null references board(num),       -- foreign key
        replytext varchar2(500) not null,                -- 댓글내용
        replyer varchar2(50) not null,                   -- 댓글 작성자
        regdate date not null,                           -- 작성일
        updatedate date not null                         -- 수정일
);
```