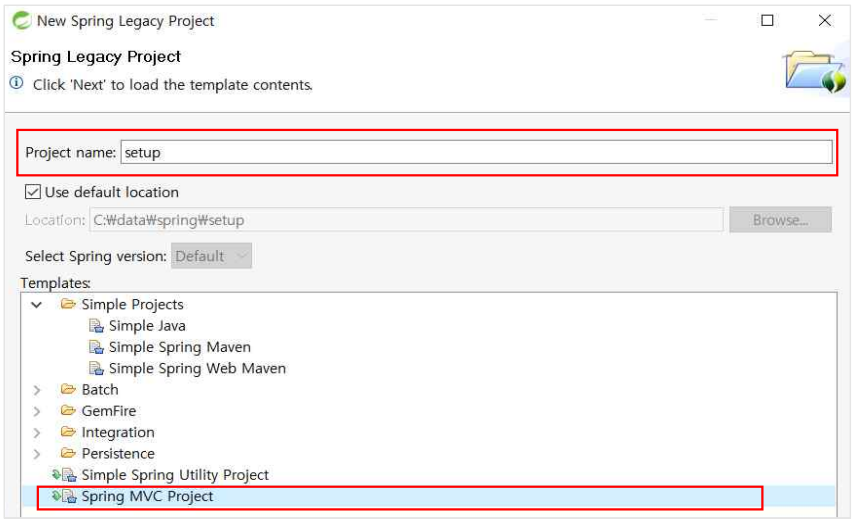
스프링

(Spring Framework)

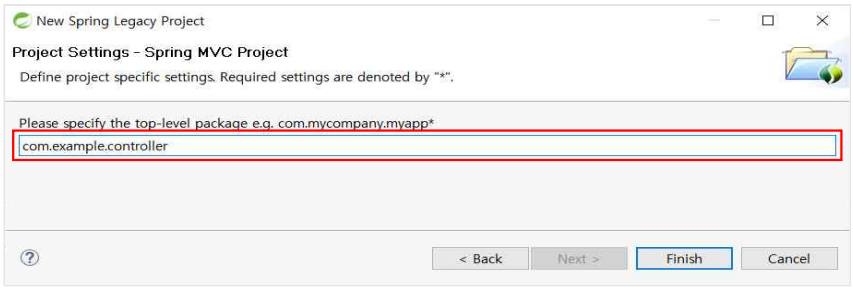
01. 스프링 Legacy 프로젝트 설정

- **MySQL DB 설정**

1) [File]-[New]-[Sprint Legacy Project]-[Sprint MVC Project]를 선택한다.



2) package명에 com.example.controller를 입력한다.



3) [해당 프로젝트를 선택]-[마우스 오른쪽 버튼]-[Properties]-[Project Facets]를 Java 버전을 1.7 이상의 버전을 선택한다.

4) pom.xml에 아래 내용을 수정 한다.

pom.xml

```
<properties>
    <java-version>1.8</java-version>
    <org.springframework-version>4.1.7.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <!--변경 전: <artifactId>servlet-api</artifactId>-->
    <artifactId>javax.servlet-api</artifactId>
    <!--변경 전: <version>2.5</version>-->
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
```

5) pom.xml에 아래 내용을 추가 한다. (자동정렬: ctrl + shift + F)

pom.xml

```
<!-- 새로운 라이브러리 추가 시작 -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.35</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.2.8</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.2.2</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.5.4</version>
</dependency>
<dependency>
    <groupId>org.bgee.log4jdbc-log4j2</groupId>
    <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
    <version>1.16</version>
</dependency>
<!-- 새로운 라이브러리 추가 종료 -->
```

6) MyBatis의 상세로그를 위해 [src/main/resources]폴더에 log4jdbc.log4j2.properties 파일을 생성한다.

[src/main/resources] log4jdbc.log4j2.properties

```
log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
```

7) [src/main/resources]아래에 [mapper] 폴더를 작성한 후 MysqlMapper.xml 파일을 생성한다.

[src/main/resources]-[mapper] MysqlMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.example.mapper.MysqlMapper">
    <select id="getTime" resultType="string">
        select now() today
    </select>
</mapper>
```

8) [src/main/java]-[com.example.mapper] 패키지를 생성한 후 MysqlMapper.java interface 파일을 생성한다.

[src/main/java]-[com.example.mapper] MysqlMapper.java

```
public interface MysqlMapper {
    public String getTime();

    @Select("select now() today2")
    public String getTime2();
}
```

9) [src]-[main]-[webapp]-[WEB-INF]-[spring] root-context.xml에 아래 내용을 추가한다.
Beans Graph 탭이 사라진 경우 xml 파일을 선택하고 우 클릭 한 후 [Spring Tools]-[Add as Bean Configuration] 선택한다.

[src]-[main]-[webapp]-[WEB-INF]-[spring] root-context.xml

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></property>
    <property name="url" value="jdbc:log4jdbc:mysql://127.0.0.1:3306/boarddb"/>
    <property name="username" value="board"/>
    <property name="password" value="pass"/>
</bean>
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="mapperLocations" value="classpath:/mapper/**/*.xml"/>
</bean>
<bean id="mapper"  class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="com.example.mapper"/>
    <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>
</bean>
```

10) [src]-[main]-[webapp]-[WEB-INF] web.xml에 한글처리를 위해 아래 내용을 추가한다.

[src]-[main]-[webapp]-[WEB-INF] web.xml

```
<filter>
    <filter-name>encoding</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
</filter>
<filter-mapping>
      <filter-name>encoding</filter-name>
      <url-pattern>/*</url-pattern>
</filter-mapping>
```

11) 데이터베이스 연결 테스트를 위해 [src/test/java]-[com.example.controller] MysqlTest.java 파일을 생성한다.

[src/test/java]-[com.example.controller] MysqlTest.java

```
//SpringJUnit4ClassRunner.class를 먼저 import한다.
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={"file:src/main/webapp/WEB-INF/spring/**/*.xml"})

public class MysqlTest {
    @Autowired
    private MysqlMapper mapper;

    @Test
    public void getTime() {
        mapper.getTime()
        mapper.getTime2();
    }
}
```

- **SqlSessionTemplate을 이용한 데이터베이스 Connection**

mybatis-spring에서 제공하는 SqlSessionTemplate은 Mybatis의 SqlSession 인터페이스를 구현한 클래스로 기본적인 트랜잭션의 관리나 쓰레드 처리의 안전성 등을 보장해 주고 데이터베이스의 연결과 종료를 책임진다.

1) root-context.xml에 sqlSession bean을 아래와 같이 추가하고 com.exmaple.dao 패키지를 스캔한다.

[src]-[main]-[webapp]-[WEB-INF]-[spring] root-context.xml

```
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate" destroy-method="clearCache">
    <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"/>
</bean>

<context:component-scan base-package="com.example.dao"/>
```

3) com.example.dao 패키지를 생성한 후 BoardDAO interface 파일과 BoardMapper 파일을 생성한다.

[src/main/java]-[com.example.dao] BoardDAO.java

```
public interface BoardDAO {
    public List<HashMap<String, Object>> list();
}
```

[src/main/resources]-[mapper] BoardMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.example.mapper.BoardMapper">
    <select id="list" resultType="hashmap">
        select * from tbl_board
    </select>
</mapper>
```

4) DAO 인터페이스와 Mapper의 작성이 완료되었다면 실제 이를 구현하는 구현 클래스를 작성한다.

[src/main/java]-[com.example.dao] BoardDAOImpl.java

```
@Repository
public class BoardDAOImpl implements BoardDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.BoardMapper";

    @Override
    public List<HashMap<String, Object>> list() {
        return session.selectList(namespace + ".list");
    }
}
```

5) 최종적으로 작성된 코드는 테스트 과정을 거쳐서 설정과 동작 여부를 확인한다.

[src/test/java]-[com.example.controller] MysqlTest.java

```
public class MysqlTest {
    ...

    @Autowired
    private BoardDAO dao;

    @Test
    public void list() {
        dao.list();
    }
}
```

- **Oracle DB를 추가 설정**

1) oracle 관련 jdbc 라이브러리를 프로젝트에 추가한다.

2) [src/main/java]-[com.exmaple.mapper.oracle] 패키지를 생성한 후 OracleMapper.java interface 파일을 생성한다.

[src/main/java]-[com.example.mapper.oracle] OracleMapper.java

```
public interface OracleMapper {
    public String getTime();
}
```

3) [src/main/resources] 아래에 [mapper.oracle] 폴더를 작성한 후 OracleMapper.xml 파일을 생성한다.

[src/main/resources]-[mapper.oracle] OracleMapper.xml

```
<mapper namespace="com.example.mapper.oracle.OracleMapper">
    <select id="getTime" resultType="string">
        select sysdate from dual
    </select>
</mapper>
```

4) [src]-[main]-[webapp]-[WEB-INF]-[spring]-root-context.xml에 아래 내용을 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[spring] root-context.xml

```
<bean id="dataSourceOracle" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"/>
    <property name="url" value="jdbc:log4jdbc:oracle:thin:@127.0.0.1:1521:XE"/>
    <property name="username" value="haksa"/>
    <property name="password" value="pass"/>
</bean>
<bean id="sqlSessionFactoryOracle" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSourceOracle"/>
    <property name="mapperLocations" value="classpath:/mapper.oracle/**/*.xml"/>
</bean>
<bean id="mapperOracle"  class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="com.example.mpper.oracle"/>
    <property name="sqlSessionFactoryBeanName" value="sqlSessionFactoryOracle"/>
</bean>
```

5) SqlSessionTemplate을 사용하는 경우

[src]-[main]-[webapp]-[WEB-INF]-[spring] root-context.xml

```
<bean id="sqlSessionOracle" class="org.mybatis.spring.SqlSessionTemplate" destroy-method="clearCache">
    <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactoryOracle"/>
</bean>
<context:component-scan base-package="com.example.mapper.oracle"/>
```

6) DAO 인터페이스와 Mapper의 작성이 완료되었다면 실제 이를 구현하는 구현 클래스를 작성한다.

[src/main/java]-[com.example.dao_oracle] OracleIDAOImpl.java

```
@Repository
public class OracleDAOImpl implements OracleDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.oracle.OracleMapper";

    @Override
    public String getTime() {
        return session.selectOne(namespace + ".getTime");
    }
}
```

02. 기본적인 게시물 관리(CURD)

• 게시물 관리를 위한 DataBase 작업

1) 사용자 계정을 생성한 후 권한을 부여한다.

boarddb 데이터베이스 생성과 계정 생성

```
create database boarddb;

create user 'board'@'localhost' identified by 'pass';

grant all privileges on board.* to 'board'@'localhost';
```

2) board user 접속 후 게시판 관리를 위한 테이블을 생성한다.

tbl_board 테이블 생성

```
create table tbl_board(
    bno int not null auto_increment,
    title varchar(200) not null,
    content text,
    writer varchar(50) not null,
    regdate datetime default now(),
    updatedate datetime default now(),
    primary key(bno)
);
```

3) 테스트를 위한 샘플 데이터를 입력한다.

tbl_board에 샘플 데이터 입력

```
insert into tbl_board(writer, title, content)
values('red','왜 리액트인가?','한때 자바스크립트는 웹 브라우저에서 간단한 연산을 했지만 현재는 웹 애플리케이션에서 가장 핵심적인 역할을 한다.');

insert into tbl_board(writer, title, content)
values('blue','컴포넌트','컴포넌트를 선언하는 방식은 두 가지이다. 하나는 함수형 컴포넌트이고, 또 다른 하나는 클래스형 컴포넌트이다');

insert into tbl_board(writer, title, content)
values('yellow','배열 비구조화 할당','배열 안에 들어 있는 값을 쉽게 추출할 수 있도록 해 주는 문법이다.');

insert into tbl_board(writer, title, content)
values('yellow','최신 부트스트랩 템플릿 사이트','Bootstrap은 다양한 웹 요소들의 디자인과 기능을 포함하고 있어 손쉽게 사이트를 제작할 수 있다.');
```

• 게시물 관리를 위한 영속계층 CURD 구현

1) 테이블 설계를 기준으로 게시판 관리를 위한 VO 클래스를 작성한다.

[src/main/java]-[com.example.domain] BoardVO.java

```
public class BoardVO {
    private int bno;
    private String title;
    private String content;
    private String writer;
    private Date regdate;
    private Date updatedate;
    //setter()/getter()/toString() 메서드 추가
}
```

2) BoardDAO 인터페이스 작성한다.

[src/main/java]-[com.example.dao] BoardDAO.java

```
public interface BoardDAO {
    public List<BoardVO> list();
    public void insert(BoardVO boardVO);
    public BoardVO read(int bno);
    public void update(BoardVO boardVO);
    public void delete(int bno);
}
```

3) Mapper xml 파일을 작성한다.

[src/main/resources]-[mapper] BoardMappre.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.example.mapper.BoardMapper">
    <select id="list" resultType="com.example.domain.BoardVO">
        select * from tbl_board order by bno desc
    </select>
    <select id="read" resultType="com.example.domain.BoardVO">
        select * from tbl_board where bno=#{bno}
    </select>
    <insert id="insert">
        insert into tbl_board(title, content, writer) values(#{title}, #{content}, #{writer})
    </insert>
    <insert id="delete">
        delete from tbl_board where bno=#{bno}
    </insert>
    <update id="update">
        update tbl_board set title=#{title}, content=#{content} where bno=#{bno}
    </update>
</mapper>
```

4) BoardDAO를 구현한 BoardDAOImpl 파일을 작성한다.

[src/main/java]-[com.example.dao] BoardDAOImpl.java

```
@Repository
public class BoardDAOImpl implements BoardDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.BoardMapper";

    @Override
    public List<BoardVO> list() {
        return session.selectList(namespace + ".list");
    }
    @Override
    public void insert(BoardVO boardVO) {
        session.insert(namespace + ".insert", boardVO);
    }
    @Override
    public BoardVO read(int bno) {
        return session.selectOne(namespace + ".read", bno);
    }
    @Override
    public void update(BoardVO boardVO) {
        session.update(namespace + ".update", boardVO);
    }
    @Override
    public void delete(int bno) {
        session.delete(namespace + ".delete", bno);
    }
}
```

4) BoardDAO 인터페이스의 CURD 메서드들을 테스트해본다.

[src/test/java]-[com.example.controller] BoardTest.java

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={"file:src/main/webapp/WEB-INF/spring/**/*.xml"})

public class BoardTest {
    @Autowired
    BoardDAO boardDAO;

    @Test
    public void list() {
        daoDAO.list();
    }

    @Test
    public void read() {
        daoDAO.read(1);
    }

    @Test
    public void insert() {
        BoardVO boardVO=new BoardVO();
        boardVO.setTitle("부트스트랩");
        boardVO.setContent("부트스트랩이란 프런트엔드 프레임워크로 웹페이지를 쉽게 디자인&개발 할 수 있게 도와주는 라이브러리이다.");
        boardVO.setWriter("user00");
        bardDAO.insert(boardVO);

        boardVO=new BoardVO();
        boardVO.setTitle("실무에서 가장 많이 사용하는 기술 스프링 프레임워크");
        boardVO.setContent("실력 있는 개발자가 되려면 실무에서 자주 사용하는 도구를 깊이 있게 이해해야 합니다.");
        boardVO.setWriter("user00");
        bardDAO.insert(boardVO);
    }

    @Test
    public void update() {
        BoardVO boardVO=new BoardVO();
        boardVO.setBno(255);
        boardVO.setTitle("게시글 제목을 수정합니다.");
        boardVO.setContent("게시글 내용을 수정합니다.");
        boardDAO.update(boardVO);
    }

    @Test
    public void delete() {
        boardDAO.delete(255);
    }
}
```

- **게시물 관리를 위한 프레젠테이션 계층의 CURD 구현**

작업	URL	Method	Parameter	Form
전체 목록	/board/list	GET		/board/list.jsp
등록 처리	/board/insert	GET / POST	BoardVO	/board/insert.jsp
조회 처리	/board/read	GET	bno	/board/read.jsp
삭제 처리	/board/delete	GET / POST	bno	/board/update.jsp
수정 처리	/board/update	GET / POST	BoardVO	/board/update.jsp

1) Tomcat 서버 설정에서 프로젝트 Path를 '/controller'에서 '/'로 변경한다.

- [window]-[Preferences]-[Web] 메뉴에서 CSS Files, HTML Files, JSP Files의 Encoding을 UTF-8로 변경한다.
- [window]-[Preferences]-[General]-[Web Browser] 메뉴에서 Use external web browser로 변경한다.

2) HomeController를 아래와 같이 수정한다.

[src/main/java]-[com.example.controlelr] HomeController.java

```
@Controller
public class HomeController {
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        return "home";
    }
}
```

3) home.jsp에 BootStrap 디자인을 아래와 같이 적용한다. (https://getbootstrap.com/)

[src]-[main]-[webapp]-[WEB-INF]-[views] home.jsp

```
<html>
<head>
    <!-- CSS only -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <!-- JavaScript Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
    <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>
    <title>Home</title>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">Logo</a>
            ...
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                ...
            </div>
        </div>
    </nav>
    <div class="container">
        <div class="row">
            <jsp:include page="${pageName == null ? '' : pageName}"/>
        </div>
    </div>
</body>
</html>
```

4) 각 모듈 분기를 위한 BoardController.java를 작성하고 목록페이지로 이동하기위한 GET 메서드를 추가한다.

[src/main/java]-[com.example.controlelr] BoardController.java

```
@Controller
@RequestMapping("/board/")
public class BoardController {
    @Autowired
    BoardDAO boardDAO;

    @RequestMapping("/list")
    public void list(Model model) {
        model.addAttribute("list", boardMapper.list());
        model.addAttribute("pageName", "board/list.jsp");
        return "/home";
    }
}
```

5) 목록 출력을 위한 list.jsp 페이지를 아래와 같이 작성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] list.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<div class="card">
    <table class="table">
        <thead>
            <tr>
                <th scope="col" width=50>No</th>
                <th scope="col" width=300>Title</th>
                <th scope="col" width=100>Writer</th>
                <th scope="col" width=200>Date</th>
            </tr>
        </thead>
        <tbody>
            <c:forEach items="${list}" var="vo">
            <tr>
                <th scope="row">${vo.bno}</th>
                <td>${vo.title}</td>
                <td>${vo.writer}</td>
                <td><fmt:formatDate pattern="yyyy-mm-dd HH:mm:ss" value="${vo.regdate}"/></td>
            </tr>
            </c:forEach>
        </tbody>
    </table>
</div>
```

6) 게시물 입력을 위해 입력화면으로 이동하는 GET 메서드와 등록을 위한 POST 메서드를 추가한다.

[src/main/java]-[com.example.controlelr] BoardController.java

```
@RequestMapping("/insert")
public String insert(Model model) {
    model.addAttribute("pageName", "board/insert.jsp");
    return "/home";
}

@RequestMapping(value="/insert", method=RequestMethod.POST)
public String insert(BoardVO boardVO){
    boardDAO.insert(boardVO);
    return "redirect:list";
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] insert.jsp

```
<div class="card" style="margin:50px 0px;padding:20px;">
    <form method="post" action="insert">
        <div class="mb-3">
            <label for="exampleFormControlInput1" class="form-label">Title</label>
            <input name="title" type="text" class="form-control" id="exampleFormControlInput1" placeholder="Enter Title">
        </div>
        <div class="mb-3">
            <label for="exampleFormControlTextarea1" class="form-label">Enter Content</label>
            <textarea name="content" class="form-control" id="exampleFormControlTextarea1" rows="5"></textarea>
        </div>
        <div class="mb-3">
            <label for="exampleFormControlInput1" class="form-label">Writer</label>
            <input name="writer" value="blue" type="text" class="form-control"  id="exampleFormControlInput1">
        </div>
        <div class="mb-3">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
    </form>
</div>
```

- redirect 시점에 한 번만 데이터를 전송할 수 있는 addFlashAttribute()라는 기능을 이용한다.

[src/main/java]-[com.example.controlelr] BoardController.java

```
@RequestMapping(value="/insert", method=RequestMethod.POST)
public String insert(BoardVO boardVO, RedirectAttributes rttr){
    rttr.addFlashAttribute("result", "OK");
    boardDAO.insert(boardVO);
    return "redirect:list";
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] list.jsp

```
<script>
    var result = "${result}";
    if(result == 'OK') {
        alert("처리가 완료되었습니다.")
    }
</script>
```

7) 게시물 조회 페이지로 이동을 하기위해 GET 메서드를 추가하고 read.jsp페이지를 작성한다.

[src/main/java]-[com.example.controlelr] BoardController.java

```
@RequestMapping("/read")
public String insert(Model model, int bno) {
    model.addAttribute("boardVO", boardDAO.read(bno));
    model.addAttribute("pageName", "board/read.jsp");
    return "/home";
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] read.jsp

```
<div class="card" style="margin:50px 0px;padding:20px;">
    <form name="frm" method="post">
        <div class="mb-3">
            <label for="exampleFormControlInput1" class="form-label">Title</label>
            <input value="${boardVO.title}" name="title" type="text" class="form-control"
                id="exampleFormControlInput1" placeholder="Enter Title">
        </div>
        <div class="mb-3">
            <label for="exampleFormControlTextarea1" class="form-label">Enter Content</label>
            <textarea name="content" class="form-control"
                id="exampleFormControlTextarea1" rows="5">${boardVO.content}</textarea>
        </div>
        <div class="mb-3">
            <label for="exampleFormControlInput1" class="form-label">Writer</label>
            <input name="writer" readonly value="${boardVO.writer}" type="text" class="form-control"
                id="exampleFormControlInput1" placeholder="Enter Writer">
        </div>
        <div class="mb-3">
            <button type="submit" class="btn btn-warning" id="btnModify">MODIFY</button>
            <button type="submit" class="btn btn-danger" id="btnRemove">REMOVE</button>
            <button type="reset"   class="btn btn-primary">RESET</button>
            <button type="submit" class="btn btn-success" id="btnList">GO LIST</button>
        </div>
    </form>
</div>
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] list.jsp

```
    ...
    <th scope="row">${vo.bno}</th>
    <td><a href="read?bno=${vo.bno}">${vo.title}</a></td>
    ...
```

- **수정, 삭제로의 링크 처리를 위해 read.jsp 페이지를 수정한다.**

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] read.jsp

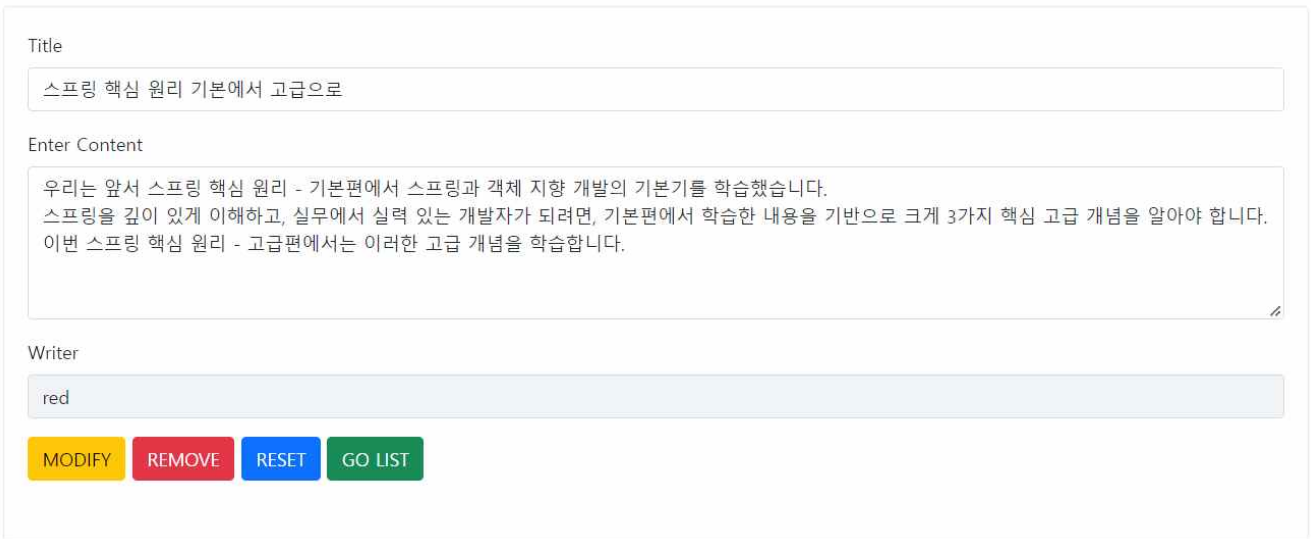
```
<script>
    $(frm).on("submit", function(e){
        e.preventDefault();
    });

    $("#btnModify").on("click", function(e){
        if(!confirm("게시글 내용을 수정하실래요?")) return;
        frm.action="/board/update";
        frm.submit();
    });

    $("#btnRemove").on("click", function(e){
        if(!confirm("게시글 내용을 삭제하실래요?")) return;
        frm.action="/board/delete";
        frm.submit();
    });

    $("#btnList").on("click", function(e){
        location.href="/board/list";
    });
</script>
```

8) 게시물 수정을 위한 화면으로 이동하는 GET 메서드와 수정과 삭제를 위한 POST 메서드를 추가한다.

[src/main/java]-[com.example.controlelr] BoardController.java

```
@RequestMapping(value="/update", method=RequestMethod.POST)
public String  updatePost(BoardVO boardVO, RedirectAttributes rttr) {
    boardDAO.update(boardVO);
    rttr.addFlashAttribute("result", "OK");
    return "redirect:list";
}

@RequestMapping(value="/delete", method=RequestMethod.POST)
public String  deletePost(int bno, RedirectAttributes rttr) {
    boardDAO.delete(bno);
    rttr.addFlashAttribute("result", "OK");
    return "redirect:list";
}
```

- **게시물 조회 페이지 실행결과**



03. 게시물의 페이징과 검색처리

- **게시물 페이징 처리**

1) 페이징 처리를 위한 충분한 양의 데이터를 입력한다.

```
insert into tbl_board(title, content)
(select title, content from tbl_board);
```

2) 페이징 처리를 위한 Criteria.java, PageMaker.java 파일을 생성한다.

[src/main/java]-[com.example.domain] Criteria.java

```
public class Criteria{
    private int page;
    private int perPageNum;
    private String searchType;
    private String keyword;

    public Criteria(){
        this(1, 10);
    }
    public Criteria(int page, int perPageNum) {
        this.page = page;
        this.perPageNum = perPageNum;
    }

    //1페이지인 경우 pageStart는 0
    public int getPageStart(){
        return (this.page - 1) * perPageNum;
    }
    public int getPage() {
        return page;
   }
    public void setPage(int page) {
        this.page = page;
    }
    public int getPerPageNum() {
        return perPageNum;
    }
    public void setPerPageNum(int perPageNum) {
        this.perPageNum = perPageNum;
    }
    public String getSearchType() {
        return searchType;
    }
    public void setSearchType(String searchType) {
        this.searchType = searchType;
    }
    public String getKeyword() {
        return keyword;
    }
    public void setKeyword(String keyword) {
        this.keyword = keyword;
    }
    @Override
    public String toString() {
        return "Criteria [page=" + page + ", perPageNum=" + perPageNum + ", searchType=" + searchType + ", keyword="
                + keyword + "]";
    }
}
```

[src/main/java]-[com.example.domain] PageMaker.java

```
public class PageMaker {
    private int totalCount;
    private int startPage;
    private int endPage;
    private boolean prev;
    private boolean next;
    private int displayPageNum = 10;
    private Criteria cri;

    public void setCri(Criteria cri){
        this.cri = cri;
    }
    public void setTotalCount(int totalCount){
        this.totalCount = totalCount;
        calcData();
    }

    private void calcData(){
        endPage = (int) (Math.ceil(cri.getPage() / (double) displayPageNum) * displayPageNum);
        startPage = (endPage - displayPageNum) + 1;
        int tempEndPage = (int) (Math.ceil(totalCount / (double) cri.getPerPageNum()));

        if(endPage > tempEndPage){
            endPage = tempEndPage;
        }
        prev = startPage == 1 ? false : true;
        next = endPage * cri.getPerPageNum() >= totalCount ? false : true;
    }

    public int getStartPage() {
        return startPage;
    }
    public void setStartPage(int startPage) {
        this.startPage = startPage;
    }
    public int getEndPage() {
        return endPage;
    }
    public void setEndPage(int endPage) {
        this.endPage = endPage;
    }
    public boolean isPrev() {
        return prev;
    }
    public void setPrev(boolean prev) {
        this.prev = prev;
    }
    public boolean isNext() {
        return next;
    }
    public void setNext(boolean next) {
        this.next = next;
    }
    public int getDisplayPageNum() {
        return displayPageNum;
    }
    public void setDisplayPageNum(int displayPageNum) {
        this.displayPageNum = displayPageNum;
    }
    public int getTotalCount() {
        return totalCount;
    }
    public Criteria getCri() {
        return cri;
    }
}
```

3) BoardDAO 인터페이스와 BoardMapper xml을 수정한다.

[src/main/java]-[com.example.dao] BoardDAO.java

```
public List<BoardVO> list(Criteria cri);
public int totalCount();  //전체 데이터의 개수를 리턴 하는 메서드
```

[src/main/resources]-[mapper] BoardMapper.xml

```
<select id="list" resultType="com.example.domain.BoardVO">
    select * from tbl_board order by bno desc
    limit #{pageStart}, #{perPageNum}
</select>
<select id="totalCount" resultType="int">
    select count(*) from tbl_board;
</select>
```

[src/main/java]-[com.example.dao] BoardDAOImpl.java

```
    @Override
    public List<BoardVO> list(Criteria cri) {
        return session.selectList(namespace + ".list", cri);
    }

    @Override
    public int totalCount() {
        return session.selectOne(namespace + ".totalCount");
    }
```

4) BoardController의 list 메서드를 아래와 같이 수정한다.

[src/main/java]-[com.example.controller] BoardController.java

```
@RequestMapping("/list")
public void list(Model model, Criteria cri) {
    cri.setPerPageNum(5);  //한 페이지에 출력되는 데이터 수
    PageMaker pm= new PageMaker();
    pm.setCri(cri);
    pm.setDisplayPageNum(5);  //한 페이지에 출력되는 페이지 수

    pm.setTotalCount(boardMapper.totalCount());
    model.addAttribute("pm", pm);
    model.addAttribute("page", page);
    model.addAttribute("list", boardMapper.list(cri));
}
```

- **Pagination 스타일**

[src]-[main]-[webapp]-[resources] pagination.css

```
.pagination {
    margin: 0px auto;
    margin-top: 20px;
    text-align: center; }

.pagination a {
   padding: 3px 10px;
   text-decoration: none; }

.pagination .active {
   background-color: gray;
   color: white; }

.pagination a:hover {
    background-color: #E2E2E2; }
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] list.jsp

```
<ul class="pagination">
    <c:if test="${pm.prev}">
        <li class="page-item"><a class="page-link" href="${pm.startPage-1}">&laquo;</a></li>
    </c:if>
    <c:forEach begin="${pm.startPage}" end="${pm.endPage}" var="i">
        <c:if test="${page==i}">
            <li class="page-item active"><a class="page-link" href="${i}">${i}</a></li>
        </c:if>
        <c:if test="${page!=i}">
            <li class="page-item"><a class="page-link" href="${i}">${i}</a></li>
        </c:if>
    </c:forEach>
    <c:if test="${pm.next}">
        <li class="page-item"><a class="page-link" href="${pm.endPage+1}">&raquo;</a></li>
    </c:if>
</ul>
<script>
    $(".pagination").on("click", ".page-item .page-link", function(e){
        e.preventDefault();
        location.href="/board/list?page=" + $(this).attr("href");
    });
</script>
```

- Ajax 처리를 위한 JSON 데이터를 이용한 경우

[src]-[main]-[webapp]-[resources] pagination.js

```
function getPagination(data){
    var str="";
    if(data.pm.prev) str += "<a href='" + (data.pm.startPage-1) + "'>&laquo;</a>";
    for(var i=data.pm.startPage; i<=data.pm.endPage; i++){
        var active = data.pm.cri.page == i ? "active" : ""
        str += "<a href='" + i + "' class=" + active + ">" + i + "</a>";
    }
    if(data.pm.next) str += "<a href='"+ (data.pm.endPage+1)+ "'>&raquo;</a>";
    return str;
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] list.jsp

```
<script src="/resources/pagination.js"></script>
<script>
    var page=${param.page==null ? 1:param.page};
    functon getList(){
        ajax.({
            ...
            success: function(data){
                var temp=Handlebars.compile($("#temp").html());
                $("#tbl").html(temp(data));
                $(".pagination").html(getPagination(data));
            }
        });
    }
    ...
    $(".pagination").on("click", "a", function(e){
        e.preventDefault();
        page=$(this).attr("href");
        getList();
   });
</script>
```

- **조회 페이지에서 현재 페이지 유지**

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] list.jsp

```
<div class="card" style="margin:50px 0px;padding:20px;">
    <form name="frm">
        <input type="hidden" name="page" value="${page}">
        <input type="hidden" name="bno"/>
    </form>
        ...
            <td><a href="${vo.bno}">${vo.title}</a></td>
        ...
</div>
<script>
    $(".table").on("click", "tr td a", function(e){
        e.preventDefault();
        frm.bno.value=$(this).attr("href");
        frm.action="read";
        frm.submit();
    });
    ...
</script>
```

[src/main/java]-[com.example.controller] BoardController.java

```
@RequestMapping("/read")
public String insert(Model model, int bno, Criteria cri) {
    model.addAttribute("page", page);
    model.addAttribute("boardVO", boardDAO.read(bno));
    model.addAttribute("pageName", "board/read.jsp");
    return "/home";
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board]-read.jsp

```
<form name="frm" method="post">
    <input type="hidden" name="bno" value="${boardVO.bno}">
    <input type="hidden" name="page" value="${page}">
        ...
</form>
<script>
    $(".btn-success").on("click", function(e){
        frm.action="list";
        frm.method="get";
        frm.submit();
    });
    ...
</script>
```

[src/main/java]-[com.example.controller] BoardController.java

```
@RequestMapping(value="/update", method=RequestMethod.POST)
public String  updatePost(BoardVO boardVO, RedirectAttributes rttr, Criteria cri) {
    boardDAO.update(boardVO);
    rttr.addFlashAttribute("result", "OK");
    rttr.addAttribute("page", cri.getPage());
    return "redirect:list";
}

@RequestMapping(value="/delete", method=RequestMethod.POST)
public String  deletePost(int bno, RedirectAttributes rttr, Criteria cri) {
    boardDAO.delete(bno);
    rttr.addFlashAttribute("result", "OK");
    rttr.addAttribute("page", cri.getPage());
    return "redirect:list";
}
```

- **게시물 검색 처리**

1) 검색을 위한 list.jsp 페이지를 아래와 같이 수정한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] list.jsp

```
<div class="card" style="margin-top:50px; padding-top:15px;">
    <form name="frm">
        <input type="hidden" name="page" value="${page}">
        <input type="hidden" name="bno" />
        <div class="row">
            <div class="col-4">
                <input name="keyword" value="${keyword}" class="form-control" type="search" placeholder="Search">
            </div>
            <div class="col-4">
                <button class="btn btn-primary" type="submit">Search</button>
           </div>
        </div>
    </form>
</div>
<script>
    $(".pagination").on("click", ".page-item .page-link", function(e) {
        e.preventDefault();
       frm.page.value=$(this).attr("href");
        frm.action="list";
        frm.submit();
    });
    $(frm).on("submit", function(e){
        e.preventDefault();
        frm.page.value=1;
        frm.action="list";
        frm.submit();
    });
    ...
</script>
```

2) BoarderDAO 인터페이스와 xml파일을 아래와 같이 수정한다.

[src/main/java]-[com.example.dao] BoardDAO.java

```
public int totalCount(Criteria cri);
```

[src/main/resources]-[mapper] BoardMapper.xml

```
<mapper namespace="com.example.mapper.BoardMapper">
    <select id="list" resultType="com.example.domain.BoardVO">
        select * from tbl_board
        <if test="keyword != null">
            where title like concat('%',#{keyword},'%') or content like concat('%',#{keyword},'%')
        </if>
        order by bno desc
        limit #{pageStart}, #{perPageNum}
    </select>
    ...
    <select id="totalCount" resultType="int">
        select count(*) from tbl_board
        <if test="keyword != null">
            where title like concat('%',#{keyword},'%') or content like concat('%',#{keyword},'%')
        </if>
    </select>
</mapper>
```

[src/main/java]-[com.example.dao] BoardDAOImpl.java

```
@Override
public int totalCount(Criteria cri) {
    return session.selectOne(namespace + ".totalCount", cri);
}
```

- **조회, 수정, 삭제 후 검색 조건 유지**

1) 검색어와 검색 조건을 저장하기 위해 read.jsp에 아래 내용을 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] read.jsp

```
<form name="frm" method="post">
    <input type="hidden" name="bno" value="${boardVO.bno}">
    <input type="hidden" name="page" value="${page}">
    <input type="hidden" name="keyword" value="${keyword}">
    ...
</form>
```

3) 수정, 삭제 작업 후 검색어와 검색조건을 저장하기 위해 BoardController에 아래 내용을 추가한다.

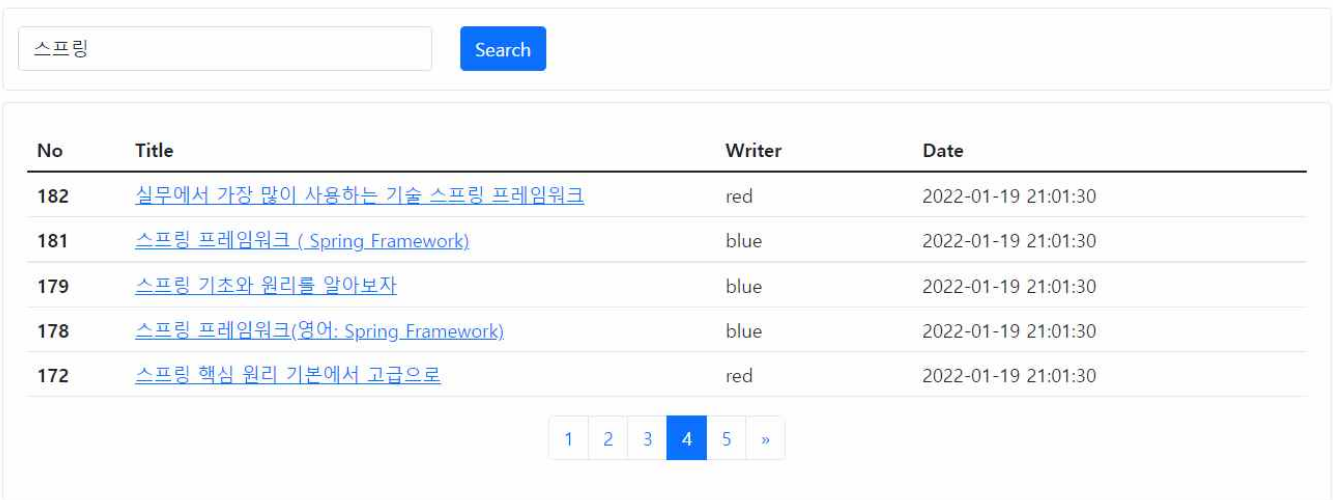
[src/main/java]-[com.example.controller] BoardController.java

```
@RequestMapping(value="/update", method=RequestMethod.POST)
public String  updatePost(BoardVO boardVO, RedirectAttributes rttr, Criteria cri) {
    boardDAO.update(boardVO);
    rttr.addFlashAttribute("result", "OK");
    rttr.addAttribute("page", cri.getPage());
    rttr.addAttribute("keyword", cri.getKeyword());
    return "redirect:list";
}
@RequestMapping(value="/delete", method=RequestMethod.POST)
public String  deletePost(int bno, RedirectAttributes rttr, Criteria cri) {
    boardDAO.delete(bno);
    rttr.addFlashAttribute("result", "OK");
    rttr.addAttribute("page", cri.getPage());
    rttr.addAttribute("keyword", cri.getKeyword());
    return "redirect:list";
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] list.jsp //Ajax처리 시 검색 버튼을 클릭한 후 검색조건 유지

```
<input type="text" id="keyword" value="${param.keyword}">

var keyword="${param.keyword ? '' : param.keyword}";
$("#search").on("click", function(){
    page=1;
    keyword=$("#keyword").val();
    getList();
});
```

- **게시물 목록 최종 결과 페이지**



- **Oracle 데이터베이스에서 게시판 목록 처리**

1) 사용자 계정을 생성한 후 권한을 부여한다.

board user 생성

```
create user board identified by pass;
grant connect, resource, dba to board;
```

2) board user 접속 후 게시판 관리를 위한 테이블을 생성한다.

tbl_board 테이블 생성

```
create table tbl_board(
    bno int not null,
    title varchar(200) not null,
    content varchar(2000),
    writer varchar(50),
    regdate date  default sysdate,
    updatedate date default sysdate
);
alter table tbl_board add constraint pk_board primary key(bno);
```

3) 테스트를 위한 샘플 데이터를 입력한다.

tbl_board에 샘플 데이터 입력

```
insert into tbl_board(bno, title, content, writer)
    values(seq_board.nextval, '제목을 입력 합니다01', '내용을 입력 합니다.01', 'user01');

insert into tbl_board(bno, title, content, writer)
    values(seq_board.nextval, '제목을 입력 합니다02', '내용을 입력 합니다.02', 'user01');

insert into tbl_board(bno, title, content, writer)
    (select seq_board.nextval, title, content, writer from tbl_board);
```

4) 1~10 번째 데이터를 출력하는 SQL문을 실행해 본다.

sql문 작성

```
select * from (select rownum rn, tbl_board.* from tbl_board order by bno desc)
where rn between 1 and 10 ;
```

5) root-context.xml 파일을 수정한다.

[src]-[main]-[webapp]-[WEB-INF]-[spring] root-context.xml

```
<bean id="dataSourceOracle" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"/>
    <property name="url" value="jdbc:log4jdbc:oracle:thin:@127.0.0.1:1521:XE"/>
    <property name="username" value="board"/>
    <property name="password" value="pass"/>
</bean>
<bean id="sqlSessionFactoryOracle" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSourceOracle"/>
    <property name="mapperLocations" value="classpath:/mapper.oracle/**/*.xml"/>
</bean>
<bean id="sqlSessionOracle" class="org.mybatis.spring.SqlSessionTemplate" destroy-method="clearCache">
    <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactoryOracle"></constructor-arg>
</bean>
<context:component-scan base-package="com.example.mapper.oracle"/>
```

6) OracleDAO 인터페이스와 OracleMapper xml 파일을 생성한다.

[src/main/resources]-[mapper_oracle] OracleMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.example.mapper.oracle.OracleMapper">
    <select id="list" resultType="com.example.domain.BoardVO">
        select * from (select ROWNUM rn, tbl_board.* from tbl_board order by bno desc)
        where rn between (#{pageStart} + 1) and (#{pageStart} + #{perPageNum})
    </select>
</mapper>
```

[src/main/java]-[com.example.mapper_oracle] OracleDAO.java

```
public interface OracleMapper {
    public List<BoardVO> list(Criteria cri);
}
```

[src/main/java]-[com.example.mapper.oracle] OracleDAOImpl.java

```
@Repository
public class OracleMapper implements OracleDAOImpl{
    @Autowired
    SqlSession session;
    String namespace ="com.example.mapper.oracle.OracleMapper";

    public List<BoardVO> list(Criteria cri){
        session.selectList(namespace + ".list", cri);
    }
}
```

7) OracleTest.java 파일에서 목록 출력을 테스트 한다.

[src/test/java]-[com.example.controller] OracleTest.java

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={"file:src/main/webapp/WEB-INF/spring/**/*.xml"})

public class OracleTest {
    @Autowired
    OracleDAO oracle;

    @Test
    public void listOracle() {
        Criteria cri=new Criteria();
        cri.setPage(2);
        cri.setPerPageNum(5);
        oracle.list(cri);
    }
}
```

8) 검색 속도를 위해 order by 대신에 index를 이용해 테스트 한다.

[src/main/resources]-[mapper_oracle] OracleMapper.xml

```
<select id="list" resultType="com.example.domain.BoardVO">
    select * from (select /*+index_desc(tbl_board pk_board)*/ ROWNUM rn, tbl_board.* from tbl_board)
    where rn between (#{pageStart} + 1) and (#{pageStart} + #{perPageNum})
</select>
```

04. REST API방식과 Ajax를 이용하는 댓글 처리

- 현재는 모바일 환경이 대두되면서 서버는 브라우저나 모바일에서 필요한 데이터만을 전달하는 API 서버의 형태로 변화고 있다.
- REST는 'Representational State Transfer'의 약어로 하나의 URI는 하나의 고유한 Resource를 대표하도록 설계된다는 개념에 전송 방식을 결합해서 원하는 작업을 지정한다.

• 댓글 처리를 위한 데이터베이스 작업

1) 댓글 작업을 위한 tbl_reply 데이블을 생성한다.

댓글 처리를 위한 테이블 생성

```
create table tbl_reply(
    rno int auto_increment primary key,
    bno int not null,
    reply varchar(1000) not null,
    replyer varchar(50) not null,
    replyDate datetime default now(),
    updateDate datetime default now(),
    foreign key(bno) references tbl_board(bno)
);
```

2) 테스트용 데이터를 입력한다.

테스트용 데이터 생성

```
insert into tbl_reply(bno, replyer, reply)
values(284, 'red', '리액트(React, React.js 또는 ReactJS)는 자바스크립트 라이브러리의 하나로서 사용자 인터페이스를 만들기 위해 사용된다.');

insert into tbl_reply(bno, replyer, reply)
values(284, 'red', '스프링 프레임워크는 자바 진영의 웹 프레임워크이다.');

insert into tbl_reply(bno, replyer, reply)
values(284, 'red', '스프링은 자바 언어 기반의 프레임워크이고, 자바는 객체 지향 언어이다.');

insert into tbl_reply(bno, replyer, reply)
values(284, 'red', '다형성을 쉽게 이해하기 위해 실세계에 비유하여 생각해보자.');

insert into tbl_reply(bno, replyer, reply)
values(284, 'red', '객체 지향 프로그래밍은 추상화, 상속, 다형성 등이 있으며 추가적으로 다중 상속 등의 특징이 존재한다.');
insert into tbl_reply(bno, reply, replyer) (select bno, reply, replyer from tbl_reply);
```

• 댓글 관리를 위한 영속계층 CURD 구현

1) 테이블 설계를 기준으로 게시판 관리를 위한 VO 클래스를 작성한다.

[src/main/java]-[com.example.domain] ReplyVO.java

```
public class ReplyVO {
    private int rno;
    private int bno;
    private String reply;
    private String replyer;

    @JsonFormat(pattern="yyyy-MM-dd hh:mm:ss",timezone ="Asia/Seoul")
    private Date replyDate;

    @JsonFormat(pattern="yyyy-MM-dd hh:mm:ss",timezone ="Asia/Seoul")
    private Date updateDate;
    ...
    //setter()/getter()/toString() 메서드 추가
```

2) Mapper 인터페이스를 작성하거나 DAO 인터페이스와 DAO 구조체를 작성한다.
Mybatis는 두 개 이상의 데이터를 파라미터로 전달하기 위해서는 1)별도의 객체로 구성하거나 2)Map을 이용하거나 3)@Param을 이용해서 이름을 사용한다. @Param의 속성값은 MyBatis에서 SQL을 이용할 때 '#{ }'의 이름으로 사용이 가능하다.

[src/main/java]-[com.example.mapper] ReplyMapper.java

```
public interface ReplyMapper {
    public List<ReplyVO> list(@Param("cri") Criteria cri, @Param("bno") int bno);
    public void insert(ReplyVO vo);
    public ReplyVO read(int rno);
    public void delete(int rno);
    public void update(ReplyVO vo);
    public int totalCount(int bno);
}
```

[src/main/java]-[com.example.dao] ReplyDAO.java

```
public interface ReplyDAO {
    public List<ReplyVO> list(Criteria cri, int bno);
    public void insert(ReplyVO vo);
    public ReplyVO read(int rno);
    public void delete(int rno);
    public void update(ReplyVO vo);
    public int totalCount(int bno);
}
```

[src/main/java]-[com.example.dao] ReplyDAOImpl.java

```
@Repository
public class ReplyDAOImpl implements ReplyDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.ReplyMapper";

    @Override
    public List<ReplyVO> list(Criteria cri, int bno) {
        HashMap<String, Object> map=new HashMap<>();
        map.put("bno", bno);
        map.put("cri", cri);
        return session.selectList(namespace + ".list", map);
    }

    @Override
    public void insert(ReplyVO vo) {
        session.insert(namespace + ".insert", vo);
    }

    @Override
    public ReplyVO read(int rno) {
        return session.selectOne(namespace + ".read", rno);
    }

    @Override
    public void delete(int rno) {
        session.delete(namespace + ".delete", rno);
    }

    @Override
    public void update(ReplyVO vo) {
        session.update(namespace + ".update", vo);
    }

    @Override
    public int totalCount(int bno) {
        return session.selectOne(namespace + ".totalCount", bno);
    }
}
```

3) Mapper xml 파일을 작성하고 DAO 구현 클래스를 작성한다.

[src/main/resources]-[mapper]-ReplyMappre.xml

```
<mapper namespace="com.example.mapper.ReplyMapper">
    <select id="list" resultType="com.example.domain.ReplyVO">
        select * from tbl_reply where bno = #{bno} order by rno desc limit #{cri.pageStart}, #{cri.perPageNum}
    </select>
    <select id="read" resultType="com.example.domain.ReplyVO">
        select * from tbl_reply where rno = #{rno}
    </select>
    <insert id="insert">
        insert into tbl_reply(bno, reply, replyer) values(#{bno}, #{reply}, #{replyer})
    </insert>
    <delete id="delete">
        delete from tbl_reply where rno=#{rno}
    </delete>
    <update id="update">
        update tbl_reply set reply=#{reply}, updatedate=now() where rno=#{rno}
    </update>
    <select id="totalCount" resultType="int">
        select count(*) from tbl_reply where bno=#{bno}
    </select>
</mapper>
```

- 댓글 관리를 위한 프레젠테이션 계층의 CURD 구현

1) 각 모듈 분기를 위한 ReplyController.java를 RestController로 작성한다.

[src/main/java]-[com.example.controleIr] ReplyController.java

```
@RestController
@RequestMapping("/reply")
public class ReplyController {
    @Autowired
    ReplyDAO dao;

    @RequestMapping("/list.json")
    public HashMap<String, Object> list(int bno, Criteria cri){
        HashMap<String, Object> map=new HashMap<>();
        cri.setPerPageNum(5);
        PageMaker pm=new PageMaker();
        pm.setCri(cri);
        pm.setDisplayPageNum(5);
        pm.setTotalCount(dao.totalCount());
        map.put("pm", pm);
        map.put("list", dao.list(cri, bno));
        return map;
    }

    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public void insert(ReplyVO vo) {  //Advanced REST 테스트를 사용하는 경우 @RequestBody 추가
        dao.insert(vo);
    }

    @RequestMapping(value="/read")
    public ReplyVO read(@RequestParam("rno") int rno){  //@RequestParam("rno") 생략가능
        return dao.read(rno);
    }

    @RequestMapping(value="/update", method=RequestMethod.POST)
    public void update(ReplyVO vo){
        dao.update(vo);
    }

    @RequestMapping(value="delete", method=RequestMethod.POST)
    public void delete(int rno) {
        dao.delete(rno);
    }
}
```

- **데이터 출력을 json 타입으로 출력을 원하는 경우**

pom.xml

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.5.4</version>
</dependency>
```

[src/main/java]-[com.example.controleIr] ReplyController.java

```
@RequestMapping(value="/list/{bno}/{page}", method=RequestMethod.GET, produces="application/json;charset=UTF-8")
public ResponseEntity<List<ReplyVO>> list(@PathVariable("bno") int bno, @PathVariable("page") int page){
    Criteria cri=new Criteria();
    cri.setPage(page);
    return new ResponseEntity<>(replyMapper.list(cri, bno), HttpStatus.OK);
}
```

- **데이터 출력을 xml 타입으로 출력을 원하는 경우**

pom.xml

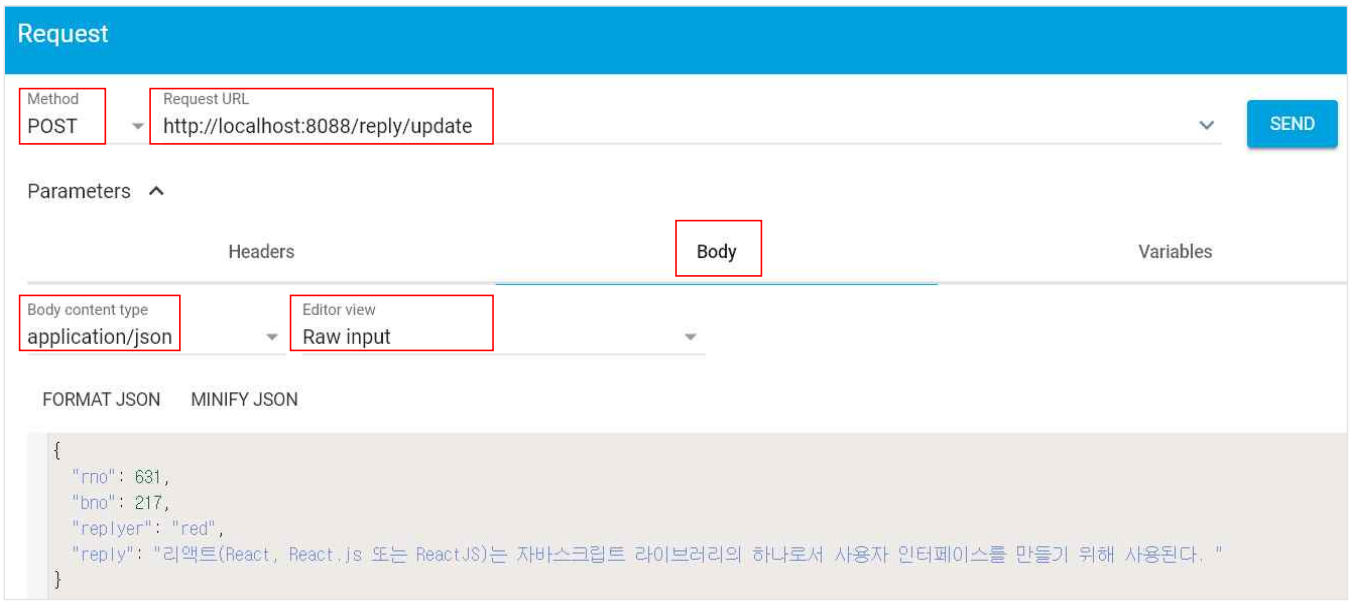
```
<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
    <version>2.5.4</version>
</dependency>
```

[src/main/java]-[com.example.controleIr] ReplyController.java

```
@RequestMapping(value="/list/{bno}/{page}", method=RequestMethod.GET, produces="application/xml;charset=UTF-8")
public ResponseEntity<List<ReplyVO>> list(@PathVariable("bno") int bno, @PathVariable("page") int page){
    Criteria cri=new Criteria();
    cri.setPage(page);
    return new ResponseEntity<>(replyMapper.list(cri, bno), HttpStatus.OK);
}
```

2) [크롬 Browser]-[웹 스토어]에서 'Advanced REST'로 검색한 후 'Advanced Rest Client Application'를 다운받아 설치한다.

- **입력 시 [Method]를 'Post'로 변경하고 [Body Content Type]을 'appication/json'으로 지정한다.**
- **[body]에는 실제 입력될 데이터를 아래와 같이 작성해 준다.**



- **특정 게시물의 목록 출력**

1) home.jsp 파일에 handlebar를 사용하기 위한 라이브러리를 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[views] home.jsp

```
<head>
    ...
    <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.1/handlebars.js"></script>
</head>
```

2) 댓글 목록 출력을 위한 reply.jsp 페이지를 작성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] reply.jsp

```
<div class="card" style="margin:10px 0px;padding:20px;">
    <div id="replies"></div>
    <script id="temp" type="text/x-handlebars-template">
        {{#each list}}
            <div class="card" style="margin-bottom:20px;">
                <div class="card-header">
                    <b>{{rno}}</b>
                    <span>{{replyer}}</span>
                    <span style="float:right;color:gray;font-size:15px;">{{replyDate}}</span>
                </div>
                <div class="card-body">
                    <p class="card-text">{{reply}}</p>
                    <a href="#" class="btn btn-primary">MODIFY</a>
                </div>
            </div>
        {{/each}}
    </script>
    <nav aria-label="Page navigation example" style="margin: 0px auto;">
        <ul class="pagination"></ul>
    </nav>
</div>
<script>
    var bno="${boardVO.bno}";
    var page=1;
    getList();
    //댓글목록 출력 함수
    function getList(){
        $.ajax({
            type: "get",
            url: "/reply/list.json",
            dataType: "json",
            data: {bno:bno, page:page},
            success: function(data){
                var template = Handlebars.compile($("#temp").html());
                $("#replies").html(template(data));
                //Pagination 출력
                var str=""
                if(data.pm.prev) str+=`<li class="page-item"><a class="page-link" href="\${data.pm.startPage-1}">&laquo</a></li>`;
                for(i=data.pm.startPage; i<=data.pm.endPage; i++){
                    var active = data.pm.cri.page==i ? "active" : "";
                    str += `<li class="page-item \${active}"><a class="page-link" href="\${i}">\${i}</a></li>`;
                }
                if(data.pm.next) str+=`<li class="page-item"><a class="page-link" href="\${data.pm.endPage+1}">&raquo</a></li>`;
                $(".pagination").html(str);
            }
        });
    }
    //페이지목록의 페이지 번호를 클릭한 경우
    $(".pagination").on("click", ".page-link", function(e) {
        e.preventDefault();
        page = $(this).attr("href");
        getList();
    });
</script>
```

3) read.jsp 페이지에 댓글 목록 출력 페이지를 추가한다.

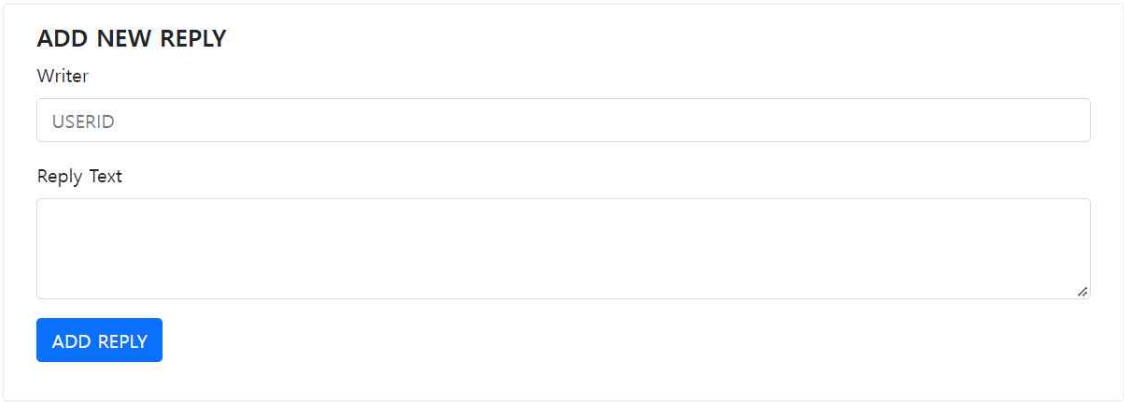
[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] read.jsp

```
<div class="card" style="margin:10px 0px;padding:20px;">
    ...
</div>
<jsp:include page="reply.jsp"/>
```

- **댓글의 등록 처리**

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] reply.jsp

```
<div class="card" >
    <div class="card-body">
        <h5><b>ADD NEW REPLY</b></h5>
        <form name="frmReply">
            <div class="mb-3">
                <label for="exampleFormControlInput1" class="form-label">Writer</label>
                <input name="writer" type="text" class="form-control" id="exampleFormControlInput1" placeholder="USERID">
            </div>
            <div class="mb-3">
                <label for="exampleFormControlTextarea1" class="form-label">Reply Text</label>
                <textarea name="reply" class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>
            </div>
            <button type="submit" class="btn btn-primary">ADD REPLY</button>
        </form>
    </div>
</div>
...
<!-- 댓글 목록 출력 -->
<script>
    $(frmReply).on("submit", function(e){
        e.preventDefault();
        var replyer="blue";
        var reply=$(frmReply.reply).val();
        if(reply=="") {
            alert("댓글 내용을 입력하세요!");
            return;
        }
        $.ajax({
            type: "post",
            url: "/reply/insert",
            data: {bno:bno, replyer:replyer, reply:reply},
            success: function(){
                alert("댓글이 등록되었습니다.");
                $(frmReply.reply).val('');
                getList();
            }
        });
    });
</script>
```



- **댓글의 삭제, 수정 처리**

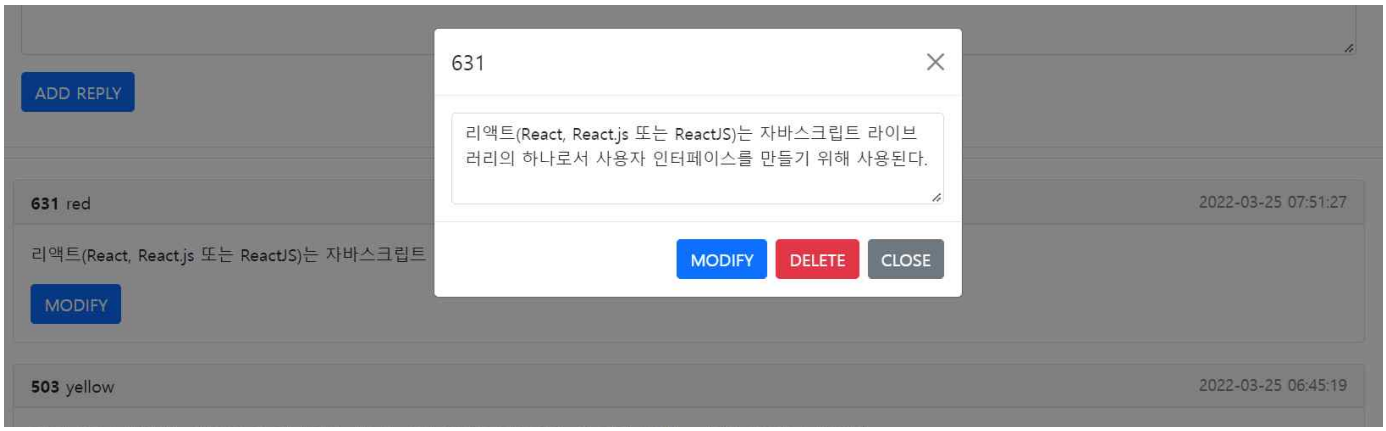
[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] reply.jsp

```
<!-- 댓글 목록 출력 -->
<div class="card" style="margin:10px 0px;padding:20px;">
    <div id="replies"></div>
    <script id="temp" type="text/x-handlebars-template">
    {{#each list}}
        <div class="card" style="margin-bottom:20px;">
            <div class="card-header">
                <b class="rno">{{rno}}</b>
                <span>{{replyer}}</span>
                <span style="float:right;color:gray;font-size:15px;">{{replyDate}}</span>
            </div>
            <div class="card-body">
                <p class="card-text">{{reply}}</p>
                <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#modalReply">
                    MODIFY
                </button>
            </div>
        </div>
    {{/each}}
    </script>
    <!-- 페이지 목록 출력 -->
    <nav aria-label="Page navigation example" style="margin: 0px auto;">
        <ul class="pagination"></ul>
    </nav>
</div>

<!-- 수정 삭제 모달창 -->
<div class="modal fade" id="modalReply" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="rno">Modal title</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">
                <textarea name="reply" class="form-control" id="reply" rows="3" cols="80"></textarea>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-primary">MODIFY</button>
                <button type="button" class="btn btn-danger">DELETE</button>
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">CLOSE</button>
            </div>
        </div>
    </div>
</div>
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] reply.jsp

```
<script>
    ...
    //댓글 목록의 수정버튼을 클릭한 경우
    $("#replies").on("click", ".btn-primary", function(){
        var card=$(this).parent().parent();
        var rno=card.find(".rno").html();
        var reply=card.find(".card-text").html();
        $("#rno").html(rno);
        $("#reply").html(reply);
    });

    //수정  버튼을 클릭한 경우
    $("#modalReply").on("click", ".btn-primary", function(){
        var rno=$("#rno").html();
        var reply=$("#reply").val();
        if(!confirm("댓글을 수정하시겠습니까?")) return;
        $.ajax({
            type: "post",
            url: "/reply/update",
            data: {rno:rno, reply:reply},
            success: function(){
                alert("댓글이 수정되었습니다.");
                getList();
                $('#modalReply').modal('hide');
             }
         });
     });

    //삭제 버튼을 클릭한 경우
    $("#modalReply").on("click", ".btn-danger", function(){
        var rno=$("#rno").html();
        if(!confirm("댓글을 삭제하시겠습니까?")) return;
        $.ajax({
            type: "post",
            url: "/reply/delete",
            data: {rno:rno},
            success: function() {
                alert("댓글이 삭제되었습니다.");
                $('#modalReply').modal('hide');
                getList();
            }
        });
    });
    ...
</script>
```

- **수정 삭제 모달창 실행결과**



05. 스프링에서의 트랜잭션 관리

- **게시글 조회 수와 댓글 수에 대한 처리**

1) 조회 수와 댓글 수를 처리하기 위한 칼럼을 tbl_board 테이블에 추가한다.

tbl_board 테이블에 아래 칼럼 추가삭제

```
alter table tbl_board add column(replycnt int default 0);
alter table tbl_board add column(viewcnt int default 0);
alter table tbl_board drop column replycnt;
```

2) 기존에 존재하는 댓글수를 replycnt에 반영하기 위해 아래의 sql문을 실행한다.

게시글에 대한 댓글수 계산

```
update tbl_board
set replycnt=(select count(rno) from tbl_reply where tbl_reply.bno=tbl_board.bno)
where bno>0;
```

3) 트랜잭션 처리를 위해 아래 내용들을 설정한다.

pom.xml

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
```

4) root-context.xml 파일에 아래 내용을 추가한다.

- [root-context.xml]-[Namespaces]에서 tx와 context를 선택한 후 아래 내용을 추가한다.
- [src/main/java] pakage에 새로운 com.example.service package를 생성한다.

[src]-[main]-[webapp]-[WEB-INF]-[spring] root-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    ...
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource"></property>
    </bean>
    <tx:annotation-driven/>

    <context:component-scan base-package="com.example.dao"/>
    <context:component-scan base-package="com.example.service"/>
</beans>
```

5) 조회수와 댓글수를 출력할 필드를 추가한다.

[src/main/java]-[com.example.domain] BoardVO.java

```
public class BoardVO {
    ..
    private int viewcnt;
    private int replycnt;
    //setter()/ getter() 메서드추가
}
```

6) 조회수 수정과 댓글수 수정 DAO와 DAO 구현체 mapper 파일을 작성한다.

[src/main/java]-[com.example.dao] BoardDAO.java

```
public interface BoardDAO{
    ...
    public void updateViewcnt(int bno);
    public void updateReplycnt(int bno, int amount);
}
```

[src/main/java]-[com.example.dao] BoardDAOImpl.java

```
@Repository
public class BoardDAOImpl implements BoardDAO{
    ...
    @Override
    public void updateViewcnt(int bno) {
        session.update(namespace + ".updateViewcnt", bno);
    }

    @Override
    public void updateReplycnt(int bno, int amount) {
        HashMap<String, Object> map=new HashMap<>();
        map.put("bno", bno);
        map.put("amount", amount);
        session.update(namespace + ".updateReplycnt", map);
    }
}
```

[src/main/resources]-[mapper]-BoardMapper.xml

```
<mapper namespace="com.example.mapper.BoardMapper">
    ...
    <update id="updateViewcnt">
        update tbl_board set viewcnt=viewcnt + 1
        where bno=#{bno}
    </update>
    <update id="updateReplycnt">
        update tbl_board set replycnt=replycnt + #{amount}
        where bno=#{bno}
    </update>
</mapper>
```

7) Board service 인터페이스와 Board service 구현체를 작성한다.

[src/main/java]-[com.example.service] BoardService.java

```
public interface BoardService {
    public BoardVO read(int bno);
}
```

[src/main/java]-[com.example.service] BoardServiceImpl.java

```
@Service
public class BoardServiceImpl implements BoardService{
    @Autowired
    BoardDAO dao;

    @Transactional
    @Override
    public BoardVO read(int bno) {
        dao.updateViewcnt(bno);
        return dao.read(bno);
    }
}
```

8) Reply service 인터페이스와 Reply service 구현체를 작성한다.

[src/main/java]-[com.example.service] ReplyService.java

```
public interface ReplyService {
    public void insert(ReplyVO vo);
    public void delete(int rno);
}
```

[src/main/java]-[com.example.service] ReplyServiceImpl.java

```
@Service
public class ReplyServiceImpl implements ReplyService{
    @Autowired
    BoardDAO bdao;

    @Autowired
    ReplyDAO rdao;

    @Transactional
    @Override
    public void insert(ReplyVO vo) {
        rdao.insert(vo);
        BoardVO boardVO=bdao.read(vo.getBno());
        bdao.updateReplycnt(boardVO.getBno(), 1);
    }

    @Transactional
    @Override
    public void delete(int rno) {
        ReplyVO replyVO=rdao.read(rno);
        bdao.updateReplycnt(replyVO.getBno(), -1);
        rdao.delete(rno);  //bno를 읽은 후 댓글삭제
    }
}
```

9) Board, Reply Controller에 service를 적용한다.

[src/main/java]-[com.example.controller] BoardController.java

```
public class BoardController {
    @Autowired
    BoardService service;
    ...
    @RequestMapping("/read")
    public String insert(Model model, int bno, Criteria cri) {
        ...
        model.addAttribute("boardVO", service.read(bno));
        return "/home";
    }
}
```

[src/main/java]-[com.example.controller] ReplyController.java

```
public class ReplyController {
    @Autowired
    ReplyService service;
    ...

    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public void insert(ReplyVO vo) {
        service.insert(vo);
    }

    @RequestMapping(value="delete", method=RequestMethod.POST)
    public void delete(int rno) {
        service.delete(rno);
    }
}
```

10) list 페이지와 reply 페이지에 조회수와 댓글수 출력을 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] list.jsp

```
<table class="table">
    <thead>
        <tr>
            ...
            <th scope="col" width=200>VIEWCNT</th>
        </tr>
    </thead>
    <tbody>
        <c:forEach items="${list}" var="vo">
            <tr>
                <th scope="row">${vo.bno}</th>
                <td><a href="${vo.bno}">${vo.title}</a> [${vo.replycnt}]</td>
                <td>${vo.writer}</td>
                <td><fmt:formatDate pattern="yyyy-mm-dd HH:mm:ss" value="${vo.regdate}" /></td>
                <td><span class="badge bg-danger">${vo.viewcnt}</span></td>
            </tr>
        </c:forEach>
    </tbody>
</table>
```

- 게시물 목록 페이지에 VIEWCNT 추가 실행결과

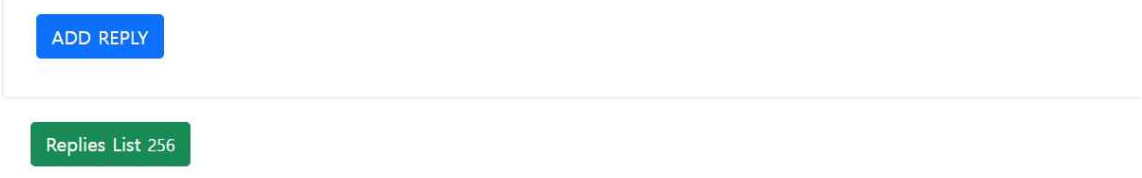
No	Title	Writer	Date	VIEWCNT
217	지금 출항합니다[256]	red	2022-01-19 21:01:30	17
216	스프링 핵심 원리 기본에서 고급으로[5]	red	2022-01-19 21:01:30	5
214	스프링 프레임워크 (Spring Framework)[0]	blue	2022-01-19 21:01:30	0
213	객체 지향 프로그래밍(OOP)[0]	blue	2022-01-19 21:01:30	0
212	스프링 기초와 원리를 알아보자[0]	blue	2022-01-19 21:01:30	0

1 2 3 4 5 »

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] reply.jsp

```
<!-- 댓글 입력 -->
...
<div class="card" style="margin-top:10px; padding: 20px;">
<div class="row" style="margin:20px 0px;">
    <div>
        <span class="btn btn-success" id="btnReplies">
          Replies List <small id="replycnt">[${boardVO.replycnt}]</small>
        </span>
    </div>
</div>
<!-- 댓글 목록 출력 -->
...
<script>
    ...
    $("#viewReplies").hide();
    $("#btnReplies").on("click", function(){
        $("#viewReplies").toggle();
    });
</script>
```

- 댓글 수 출력 버튼 실행 결과



- **메시지 주고받기**

메시지를 보낸 사용자에게는 포인트를 10점 지급, 메시지를 확인하는 사용자에게 포인트 5점을 지급하는 프로그램을 작성하시오.

1) 사용자 테이블과 메시지 테이블을 생성하고 샘플 데이터를 입력한다.

tbl_user 테이블 생성

```
create table tbl_user (
    uid varchar(20) not null primary key,
    upw varchar(200) not null,
    uname varchar(100) not null,
    point int default 0 );
```

tbl_user 테이블에 샘플 데이터 입력

```
insert into tbl_user(uid, upass, uname) values('red', 'pass','홍길동');
insert into tbl_user(uid, upass, uname) values('blue', 'pass','이순신');
insert into tbl_user(uid, upass, uname) values('green', 'pass','강감찬');
insert into tbl_user(uid, upass, uname) values('pink', 'pass','심청이');
```

tbl_message 테이블 생성

```
create table tbl_message (
    mid int auto_increment primary key,
    receiver varchar(20) not null,
    sender varchar(20) not null,
    message text,
    sendDate datetime default now(),
    readDate datetime,
    foreign key(receiver) references tbl_user(uid),
    foreign key(sender) references tbl_user(uid) );
```

tbl_message 테이블에 샘플 데이터 입력

```
insert into tbl_message(receiver, sender, message) values('red','blue', '이순신 토요일에 우리 만날까?');
insert into tbl_message(receiver, sender, message) values('blue','red','홍길동 오랜만이네. 그날은 선약이 있어서 불가능해.');
insert into tbl_message(receiver, sender, message) values('green', 'pink', '심청아 잘 지내지?');
insert into tbl_message(receiver, sender, message) values('pink', 'green', '강감찬 오랜만이네. 반갑다.');
```

2) UserVO 파일과 MessageVO 파일을 생성한다.

[src/main/java]-[com.example.domain] UserVO.java

```
public class UserVO {
    private String uid;
    private String upw;
    private String uname;
    private int point;
    //setter(), getter(), toString() 메서드 추가
}
```

[src/main/java]-[com.example.domain] MessageVO.java

```
public class MessageVO extends UserVO {{
    private int mid;
    private String receiver;
    private String sender;
    private String message;
    @JsonFormat(pattern="yyyy-MM-dd kk:mm:ss", timezone="Asia/Seoul")
    private Date sendDate;
    @JsonFormat(pattern="yyyy-MM-dd kk:mm:ss", timezone="Asia/Seoul")
    private Date readDate;
    private String uname;
    //setter(), getter(), toString 메서드 추가
}
```

3) UserDAO 인터페이스와 구현체, UserMapper xml 파일을 생성한다.

[src/main/java]-[com.example.dao] UserDAO.java

```
public interface UserDAO {
    public List<UserVO> list();
    public UserVO read(String uid);
}
```

[src/main/java]-[com.example.dao] UserDAO.java

```
@Repository
public class UserDAOImpl implements UserDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.UserMapper";

    @Override
    public List<UserVO> list() {
        return session.selectList(namespace + ".list");
    }

    @Override
    public UserVO read(String uid) {
        return session.selectOne(namespace + ".read", uid);
    }
}
```

[src/main/resources]-[mapper] UserMapper.xml

```
<mapper  namespace="com.example.mapper.UserMapper">
    <select id="list" resultType="com.example.domain.UserVO">
        select * from tbl_user
    </select>
    <select id="read" resultType="com.example.domain.UserVO">
        select * from tbl_user  where uid=#{uid}
    </select>
</mapper>
```

4) UserController를 작성한다.

[src/main/java]-[com.exmaple.controlelr] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    UserMapper userMapper;

    @RequestMapping("list")
    public void listUser(Model model) {
        model.addAttribute("list", userMapper.list());
    }

    @RequestMapping("/send")
    public String send(String uid, Model model){
        model.addAttribute("vo", dao.read(uid));
        return "/user/send";
    }

    @RequestMapping("/receive")
    public String receive(String uid, Model model){
        model.addAttribute("vo", dao.read(uid));
        return "/user/receive";
    }
}
```

5) 사용자 목록 출력을 위해 list.jsp 파일을 생성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] list.jsp

```
<div class="page">
    <table>
        <tr class="title">
            <td width=200>아이디</td>
            <td width=200>이름</td>
            <td width=200>보낸메시지</td>
            <td width=200>받은메시지</td>
        </tr>
        <c:forEach items="${list}" var="vo">
            <tr class="center row">
                <td>${vo.uid}</td>
                <td>${vo.uname}</td>
                <td><button class="location.href='send?${vo.uid}'">보낸메시지</button>
                <td><button class="location.href='receive?${vo.uid}'">받은메시지</button>
            </tr>
        </c:forEach>
    </table>
</div>
```

- 사용자목록 페이지 실행 결과

아이디	이름	보낸메시지	받은메시지
blue	이순신	보낸메시지	받은메시지
green	강감찬	보낸메시지	받은메시지
pink	심청이	보낸메시지	받은메시지
red	홍길동	보낸메시지	받은메시지

6) 메시지 보내기 프로그램을 작성한다.

- 사용자 테이블에 포인트를 증가하는 Mapper, DAO를 작성한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<mapper  namespace="com.example.mapper.UserMapper">
    ...
    <update id="updatePoint">
        update tbl_user set point=point+#{amount}
        where uid=#{uid}
    </update>
</mapper>
```

[src/main/java]-[com.example.dao] UserDAO.java

```
public interface UserDAO {
    ...
    public void updatePoint(String uid, int amount);
}
```

[src/main/java]-[com.example.dao] UserDAOImpl.java

```
public class UserDAOImpl implements UserDAO{
    @Override
    public void updatePoint(String uid, int amount) {
        HashMap<String, Object> map=new HashMap<>();
        map.put("uid", uid);
        map.put("amount", amount);
        session.update(namespace + ".updatePoint", map);
    }
}
```

- 메시지를 전송하고 보낸 메시지 목록을 출력하는 Mapper, DAO를 작성한다.

[src/main/resources]-[mapper] MessageMapper.xml

```
<mapper namespace="com.example.mapper.MessageMapperMapper">
    <select id="listSend" resultType="com.example.domain.MessageVO">
        select tbl_message.*, uname from tbl_user, tbl_message
        where receiver=uid and sender=#{uid} order by mid desc
    </select>
    <insert id="insert">
        insert into tbl_message(sender, receiver, message)
        values(#{sender}, #{receiver}, #{message})
    </insert>
</mapper>
```

[src/main/java]-[com.example.dao] MessageDAO.java

```
public interface MessageDAO {
    public List<MessageVO> listSend(String uid);
    public void insert(MessageVO vo);
}
```

[src/main/java]-[com.example.dao] MessageDAOImpl.java

```
@Repository
public class MessageDAOImpl implements MessageDAO{
    @Autowired
    SqlSession session;

    String namespace="com.example.mapper.MessageMapperMapper";

    @Override
    public List<MessageVO> listSend(String uid) {
        return session.selectList(namespace + ".listSend", uid);
    }

    @Override
    public void insert(MessageVO vo) {
        session.insert(namespace + ".insert", vo);
    }
}
```

- 메시지 전송 후 포인트를 10증가하는 Service를 작성한다.

[src/main/java]-[com.example.service] MessageService.java

```
public interface MessageService {
    public void insert(MessageVO vo);
}
```

[src/main/java]-[com.example.service] MessageServiceImpl.java

```
@Service
public class BoardServiceImpl implements BoardService{
    @Autowired
    BoardDAO dao;

    @Transactional
    @Override
    public BoardVO read(int bno) {
        dao.updateViewcnt(bno);
        return dao.read(bno);
    }
}
```

- **메시지 전송을 위한 Controller를 작성한다.**

[src/main/java]-[com.example.controller] MessageController.java

```
@RestController
@RequestMapping("/message")
public class MessageController {
    @Autowired
    MessageDAO messageDAO;

    @Autowired
    UserDAO userDAO;

    @Autowired
    MessageService service;

    @RequestMapping("/send")
    public List<MessageVO> list(String uid){
        return messageDAO.listSend(uid);
    }

    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public void insert(MessageVO vo){
        service.insert(vo);
    }
}
```

- **사용자 포인트 출력을 위한 REST API를 Controller에 추가한다.**

[src/main/java]-[com.example.controller] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    ...
    @RequestMapping("/getPoint")
    @ResponseBody
    public int getPoint(String uid){
        UserVO vo=dao.read(uid);
        return vo.getPoint();
    }
}
```

- **메시지 보내기 보낸 목록 출력 페이지**

- **메시지 보내기와 보낸 메시지 목록 출력 페이지를 작성한다.**

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] send.jsp

```
<style>
    small { color:red; font-weight: bold; }
    .message { overflow: hidden; text-overflow: ellipsis; white-space:nowrap; }
    .ellipsis { width: 300px; }
</style>
<div class="page">
    <!-- 사용자 정보 출력 -->
    <table>
        <tr>
            <td class="title" width=100>회원정보</td>
            <td width=300>${vo.uname} [${vo.uid}]</td>
            <td class="title" width=100>포인트</td>
            <td width=200 id="point">${vo.point}</td>
        </tr>
        <tr>
            <td class="title">받는이</td>
            <td colspan=3>
                <select id="receiver">
                    <c:forEach items="${list}" var="v">
                        <c:if test="${vo.uid!=v.uid}">
                            <option value="${v.uid}">${v.uname} (${v.uid})</option>
                        </c:if>
                    </c:forEach>
                </select>
            </td>
        </tr>
        <tr>
            <td class="title">보낼내용</td>
            <td colspan=3>
                 <input type="text" id="message" placeholder="내용을 입력하세요.">
                 <button id="btnSend">메시지 전송</button>
            </td>
        </tr>
    </table>
    <!-- 보낸 메시지 목록 출력 -->
    <table id="tbl" style="margin-top:10px;"></table>
    <script id="temp" type="text/x-handlebars-template">
        <tr class="title">
            <td width=70>ID.</td>
            <td width=500>메시지</td>
            <td width=150>받은사람</td>
            <td width=150>확인날짜</td>
        </tr>
        {{#each .}}
            <tr>
                <td>{{mid}}</td>
                <td><p class="ellipsis message">{{message}}</p></td>
                <td>{{uname}} ({{receiver}})</td>
                <td><small>{{display readDate}}</small></td>
            </tr>
        {{/each}}
    </script>
    <script>
        Handlebars.registerHelper("display", function(readDate){
            if(!readDate) {
                return "확인안함";
            }else {
                retrun readDate;
        });
    </script>
</div>
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] send.jsp

```
<script>
    var uid="${vo.uid}";
    getList();

    //message 입력상자에서 엔터키를 친 경우
    $("#message").on("keypress", function(e){
        if(e.keyCode == 13) {
            $("#btnSend").click();
        }
    });

    //전송버튼을 클릭한 경우
    $("#btnSend").on("click", function(){
        var receiver=$("#receiver").val();
        var message=$("#message").val();
        if(message==""){
            alert("내용을 입력하세요.");
            $("#message").focus();
            return;
        }

        if(!confirm("메시지를 전송하실래요?")) return;
        //메시지 저장
        $.ajax({
            type: "post",
            url: "/message/insert",
            data: {sender:uid, receiver:receiver, message:message},
            success: function(data) {
                alert("메시지를 전송하였습니다.");
                //포인트 출력
                $.ajax({
                    type: "get",
                    url: "/user/getPoint",
                    data: {uid:uid},
                    success: function(data){
                        $("#point").html(data);
                    }
                });
                $("#message").val("");
                getList();
            }
        });
    });

    //보낸 메시지 목록 출력 함수
    function getList(){
        $.ajax({
            type: "get",
            url: "/message/send",
            data: {uid:uid},
            dataType: "json",
            success: function(data){
                var template = Handlebars.compile($("#temp").html());
                $("#tbl").html(template(data));
            }
        });
    }
</script>
```

7) 받은 메시지 확인 프로그램을 작성한다.

- 받은 메시지 확인을 위한 Mapper, DAO, DAO 구현체를 작성한다.

[src/main/resources]-[mapper] MessageMapper.xml

```
<mapper namespace="com.example.mapper.MessageMapperMapper">
    <select id="listReceive" resultType="com.example.domain.MessageVO">
        select tbl_message.*, uname from tbl_user, tbl_message
        where sender=uid and receiver=#{uid}
        order by mid desc
    </select>
    <select id="read" resultType="com.example.domain.MessageVO">
        select * from tbl_message where mid=#{mid}
    </select>
    <update id="updateReadDate">
       update tbl_message set readDate=now()
       where mid=#{mid}
    </update>
</mapper>
```

[src/main/java]-[com.example.dao] MessageDAO.java

```
public interface MessageDAO {
    public List<MessageVO> listSend(String uid);
    public void insert(MessageVO vo);
    public List<MessageVO> listReceive(String uid);
    public MessageVO read(int mid);
    public void updateReadDate(int mid);
}
```

[src/main/java]-[com.example.dao] MessageDAOImpl.java

```
@Repository
public class MessageDAOImpl implements MessageDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.MessageMapperMapper";
    ...
    @Override
    public List<MessageVO> listReceive(String uid) {
        return session.selectList(namespace + ".listReceive", uid);
    }

    @Override
    public MessageVO read(int mid) {
        return session.selectOne(namespace + ".read", mid);
    }

    @Override
    public void updateReadDate(int mid) {
        session.update(namespace + ".updateReadDate", mid);
    }
}
```

- 받은 메시지 확인 후 ReadDate와 Pont증가를 위한 서비스를 작성한다.

[src/main/java]-[com.example.service] MessageService.java

```
public interface MessageService {
    public void insert(MessageVO vo);
    public MessageVO read(int mid);
}
```

[src/main/java]-[com.example.service] MessageServiceImpl.java

```
public class MessageServiceImpl implements MessageService{
    ...
    @Transactional
    @Override
    public MessageVO read(int mid) {
        MessageVO messageVO=messageDAO.read(mid);
        if(messageVO.getReadDate() == null) {
            userDAO.updatePoint(messageVO.getReceiver(), 5);
            messageDAO.updateReadDate(mid);
            messageVO=messageDAO.read(mid);
        }
        return messageVO;
    }
}
```

- 받은 메시지 확인을 위한 Controller와 출력을 위한 페이지를 작성한다.

[src/main/java]-[com.example.controller] MessageController.java

```
public class MessageController {
    ...
    @RequestMapping("/receive")
    public List<MessageVO> listReceive(String uid){
        return messageDAO.listReceive(uid);
    }

    @RequestMapping(value="/read")
    public MessageVO read(int mid){
        return service.read(mid);
    }
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] receive.jsp

```
<div class="page">
    <table>
        <tr>
            <td class="title" width=100>회원정보</td>
            <td width=300>${vo.uname} [${vo.uid}]</td>
            <td class="title" width=100>포인트</td>
            <td width=200 id="point">${vo.point}</td>
        </tr>
    </table>
    <table id="tbl" style="margin-top:10px;"></table>
    <script id="temp" type="text/x-handlebars-template">
        <tr class="title">
            <td width=50>ID.</td>
            <td width=300>메시지</td>
            <td width=100>보낸사람</td>
            <td width=130>확인날짜</td>
            <td width=100>메시지확인</td>
        </tr>
        {{#each .}}
            <tr class="row{{mid}}">
                <td class="mid">{{mid}}</td>
                <td><p class="message">{{message}}</p></td>
                <td>{{uname}} [{{sender}}]</td>
                <td><small class="readDate">{{display readDate}}</small></td>
                <td><button readDate="{{readDate}}">메시지확인</button>
            </tr>
            <tr class="view{{mid}}" style="display:none;">
                <td colspan=5><p style="padding: 20px 0px;">{{message}}</p></td>
            </tr>
        {{/each}}
    </script>
</div>
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] receive.jsp

```
<script>
    Handlebars.registerHelper("display", function(readDate){
        if(!readDate) return "확인안함"; else return readDate;
    });

    var uid="${vo.uid}";
    getList();

    $("#tbl").on("click", "tr button", function(){
        var row=$(this).parent().parent();
        var mid=row.find(".mid").html();

        $(".view"+mid).toggle();
        if($(this).attr("readDate") != "") return;

        $.ajax({  //메시지읽기
            type: "get",
            url: "/message/read",
            data: {mid:mid},
            dataType: "json",
            success: function(data){
                row.find(".readDate").html(data.readDate);
                $.ajax({  //포인트 출력
                    type:"get",
                    url: "/user/getPoint",
                    data: {uid:uid},
                    success: function(data){
                        $("#point").html(data);
                    }
                });
             }
           });
    });

    function getList(){  //받은 메시지 목록 출력
        $.ajax({
            type: "get",
            url: "/message/receive",
            data: {uid: uid},
            dataType: "json",
            success: function(data){
                var template = Handlebars.compile($("#temp").html());
                $("#tbl").html(template(data));
            }
        });
    }
</script>
```

- **받은 메시지 실행 결과**

회원정보	강감찬 [green]	포인트	30

ID.	메시지	보낸사람	확인날짜	메시지확인
9	강감찬 메시지 확인 부탁해!	이순신 [blue]	2022-03-27 09:26:18	메시지확인
강감찬 메시지 확인 부탁해!				
5	강감찬 토요일에 이순신이랑 만날까?	이순신 [blue]	2022-03-27 09:32:39	메시지확인
3	심청아 잘 지내지?	심청이 [pink]	확인안함	메시지확인

- **계좌 이체**

특정인에게 돈을 이체하는 프로그램을 작성해 본다. 입금인 경우 내 계좌 잔액이 증가하고 출금인 경우 내 계좌 잔액이 감소한다.

1) 이체 프로그램을 위한 테이블을 생성한다.

tbl_account 테이블 생성

```
create table tbl_account(
    ano char(4) primary key,
    aname varchar(20),
    openDate datetime default now(),
    balance double
);
```

tbl_account 테이블에 샘플 데이터 입력 [계좌정보를 저장하는 테이블]

```
insert into tbl_account(ano, aname, balance) values('1001', '홍길동', 1000);
insert into tbl_account(ano, aname, balance) values('1002', '심청이', 2000);
insert into tbl_account(ano, aname, balance) values('1003', '강감찬', 3000);
```

tbl_trade 테이블 생성 [통장의 거래 내역을 저장하는 테이블]

```
create table tbl_trade(
    id int auto_increment primary key,
    ano char(4),
    type char(4),
    tno char(4),
    amount double,
    tradeDate dateTime default now(),
    foreign key(ano) references tbl_account(ano),
    foreign key(tno) references tbl_account(ano)
);
```

tbl_account 테이블에 샘플 데이터 입력 [계좌정보를 저장하는 테이블]

```
insert into tbl_trade(ano, tno, type, amount) values('1001', '1002', '출금', 500);  /*1001 계좌에서 1002 계좌로 500원 출금한다.*/
update tbl_account set balance=balance -500 where ano='1001';  /*1001 계좌의 잔액을 500원 출금한다. */
insert into tbl_trade(ano, tno, type, amount) values('1002', '1001', '입금', 500);  /*1002 계좌에서 1001 계좌로 500원 입금한다.*/
update tbl_account set balance=balance + 500 where ano='1002';  /*1002 계좌의 잔액을 500원 입금한다. */
```

2) AccountVO와 TradeVO 클래스를 생성한다.

[src/main/java]-[com.example.domain] AccaountVO.java

```
public class AccountVO {
    private String ano;
    private String aname;
    private Date openDate;
    private double balance;
    //setter(), getter(), toString() 메서드 추가
}
```

[src/main/java]-[com.example.domain] TradeVO.java

```
public class TradeVO extends AccountVO{
    private int id;
    private String ano;
    private String tno;
    private String type;
    private double amount;
    @JsonFormat(pattern="yyyy-MM-dd kk:mm:ss", timezone="Asia/Seoul")
    private Date tradeDate;
    //setter(), getter(), toString() 메서드 추가
}
```

4) AccountDAO 인터페이스와 구현체, AccaountMapper xml 파일을 생성한다.

[src/main/java]-[com.example.dao] AccountDAO.java

```
public interface AccountDAO {
    public List<AccountVO> list();
    public AccountVO read(String ano);
    public void updateBalance(String ano, double amount);
}
```

[src/main/resources]-[mapper] AccountMapper.xml

```
<mapper namespace="com.example.mapper.AccountMapper">
    <select id="list" resultType="com.example.domain.AccountVO">
        select * from tbl_account
    </select>
    <select id="read" resultType="com.example.domain.AccountVO">
        select * from tbl_account
        where ano = #{ano}
    </select>
    <update id="updateBalance">
        update tbl_acount set balance=balance + #{amount}
    </update>
</mapper>
```

5) AccountController 파일을 작성한다.

[src/main/java]-[com.example.controller] AccountController.java

```
@Controller
@RequestMapping("/account")
public class AccountController {
    @Autowired
    AccountMapper dao;

    @RequestMapping("/list")
    public void list(Model model) {
        model.addAttribute("list", dao.list());
    }
}
```

6) 계좌 목록을 출력하기 위한 list.jsp 파일을 작성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[account] list.jsp

```
<div class="page">
    <table>
        <tr class="title">
            <td width=100>계좌번호</td>
            <td width=100>계좌주명</td>
            <td width=150>통장개설일</td>
            <td width=100>잔액</td>
            <td width=100>통장내역</td>
        </tr>
        <c:forEach items="${list}" var="vo">
            <tr class="center row">
                <td class="uid">${vo.ano}</td>
                <td>${vo.aname}</td>
                <td><fmt:formatDate value="${vo.openDate}" pattern="yyyy-MM-dd HH:mm:ss"/></td>
                <td><fmt:formatNumber value="${vo.balance}" pattern="#,###"/></td>
                <td><button onClick="location.href='read?ano=${vo.ano}'">통장내역</button></td>
            </tr>
        </c:forEach>
    </table>
</div>
```

- **계좌목록 출력 페이지 실행 결과**

계좌번호	계좌주명	통장개설일	잔액	통장내역
1001	홍길동	2022-03-27 17:20:15	1,500	통장내역
1002	심청이	2022-03-27 17:20:15	1,500	통장내역
1003	강감찬	2022-03-27 17:20:15	3,000	통장내역

7) 통장 거래 목록을 출력하기 위해 TradeDAO 인터페이스와 TradeMapper xml 파일을 작성한다.

[src/main/java]-[com.example.DAO] TradeDAO.java

```
public interface TradeDAO {
    public void insert(TradeVO vo);
    public List<TradeVO> list(String ano);
}
```

[src/main/resources]-[mapper] TradeMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.example.mapper.TradMapper">
    <select id="list" resultType="com.example.domain.TradeVO">
        select tbl_trade.*, aname
        from tbl_trade, tbl_account
        where tbl_trade.ano=#{ano} and tbl_trade.tno=tbl_account.ano
        order by id desc
    </select>
    <insert id="insert">
        insert into tbl_trade(ano, tno, type, amount)
        values(#{ano}, #{tno}, #{type}, #{amount})
    </insert>
</mapper>
```

[src/main/java]-[com.example.DAO] TradeDAOImpl.java

```
@Repository
public class TradeDAOImpl implements TradeDAO{
    @Autowired
    SqlSession session;

    String namespace="com.example.mapper.TradMapper";

    @Override
    public List<TradeVO> list(String ano) {
        return session.selectList(namespace + ".list", ano);
    }

    @Override
    public void insert(TradeVO vo) {
        session.insert(namespace + ".insert", vo);
    }
}
```

8) AccountController에 아래 메서드들을 추가한다.

[src/main/java]-[com.example.controller] AccountController.java

```
public class AccountController {
    ...
    @RequestMapping("/getBalance")
    @ResponseBody
    public double getBalance(String ano){
        AccountVO vo=dao.read(ano);
        return vo.getBalance();
    }
}
```

9) TradeServe 인터페이스와 TradeServieImpl 파일을 작성한다.

[src/main/java]-[com.example.service] TradeService.java

```
public interface TradeService {
    public void insert(TradeVO vo);
}
```

[src/main/java]-[com.example.service] TradeServiceImpl.java

```
@Service
public class TradeServiceImpl implements TradeService {
    @Autowired
    AccountDAO accountDAO;

    @Autowired
    TradeDAO tradeDAO;

    @Transactional
    @Override
    public void insert(TradeVO vo) {
        //출금하기
        String ano=vo.getAno();
        String tno=vo.getTno();
        vo.setType("출금");
        tradeDAO.insert(vo);
        accountDAO.updateBalance(ano, -1*vo.getAmount());

         //입금하기
         vo.setAno(tno);
         vo.setTno(ano);
         vo.setType("입금");
         tradeDAO.insert(vo);
         accountDAO.updateBalance(tno, vo.getAmount());
    }
}
```

- 거래내역 출력 페이지 실행 결과

계좌번호	1001	계좌주명	홍길동	잔액	1,800
이체정보	심청이 (1002)		이체		

No.	계좌주명	입출금	입출금액	입출금일
47	심청이 (1001)	입금	500	2022-03-27 19:00:27
44	심청이 (1001)	출금	200	2022-03-27 19:00:14
43	심청이 (1001)	입금	1,000	2022-03-27 17:20:54

10) 거래 내역을 출력하는 read.jsp 파일을 작성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[account] read.jsp

```
<div class="page">
    <table id="account">
        <tr>
            <td class="title" width=100>계좌번호</td>
            <td width=100>${vo.ano}</td>
            <td class="title" width=100>계좌주명</td>
            <td width=100>${vo.aname}</td>
            <td class="title" width=100>잔액</td>
            <td width=100>
                <span id="fmtBalance" style="color:red;font-weight:bold;">
                    <fmt:formatNumber value="${vo.balance}" pattern="#,###"/>
                </span>
                <span id="balance" style="display:none;">${vo.balance}</span>
            </td>
        <tr>
        <tr>
            <td class="title" width=100>이체정보</td>
            <td colspan=5>
                <select id="tno">
                    <c:forEach items="${list}" var="v">
                        <c:if test="${v.ano != vo.ano}">
                            <option value="${v.ano}">${v.aname} (${v.ano})</option>
                        </c:if>
                    </c:forEach>
                </select>
               <input type="text" id="amount">
               <button id="btnTrans">이체</button>
           </td>
        </tr>
    </table>
    <table id="tbl" style="margin-top:10px;"></table>
    <script id="temp" type="text/x-handlebars-template">
        <tr class="title">
            <td width=50>No.</td>
            <td width=170>계좌주명</td>
            <td width=100>입출금</td>
            <td width=100>입출금액</td>
            <td width=200>입출금일</td>
        </tr>
        {{#each .}}
            <tr class="center row" style="{{display type}}">
                <td>{{id}}</td>
                <td>{{aname}} ({{ano}})</td>
                <td>{{type}}</td>
                <td><span class="fmtAmount">{{fmtDisplay amount}}</span></td>
                <td>{{tradeDate}}</td>
            </tr>
        {{/each}}
    </script>
    <script>
        Handlebars.registerHelper("display", function(type){
            if(type=="입금") {
                return "color:blue;";
            }else {
                return "color:red;";
            }
        });

        Handlebars.registerHelper("fmtDisplay", function(amount){
            var fmtAmount = amount.toString().replace(/\B(?<!\.\d*)(?=(\d{3})+(?!\d))/g, ",");
            return fmtAmount;
        });
    </script>
</div>
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[account] read.jsp

```
<script>
    var ano="${vo.ano}";
    getList();

    //이체버튼을 클릭한 경우
    $("#btnTrans").on("click", function(){
        var tno=$("#tno").val();
        var amount=$("#amount").val();
        var balance=$("#balance").html();

        //이체금액 유효성 체크
        if(amount.replace(/[0-9]/g,'') || amount == ""){
            alert("금액을 숫자로 입력하세요!");
            $("#amount").focus();
            return;
        }else if(parseInt(amount) > parseInt(balance)){
            alert("잔액이 부족합니다!");
            $("#amount").focus();
            return;
        }
        if(!confirm(amount + "원을 정말로 이체하실래요?")) return;

        $.ajax({  //계좌이체하기
            type: "post",
            url: "/trade/insert",
            data: {ano:ano, tno:tno, amount:amount},
            success: function(){
                //현재 잔액을 구하기
                $.ajax({
                    type: "get",
                    url: "/account/getBalance",
                    data: {ano:ano},
                    success: function(data){
                        $("#balance").html(data);
                        //금액에 천단위에 컴머 출력
                        var fmtBalance = data.toString().replace(/\B(?<!\.\d*)(?=(\d{3})+(?!\d))/g, ",");
                        $("#fmtBalance").html(fmtBalance);
                        $("#amount").val("");
                        getList();
                    }
                });  //현재잔액 구하기
            }
        });  //계좌 이체하기

    });  //이체버튼을 클릭한 경우

    //거래 내역 출력 함수
    function getList(){
        $.ajax({
            type: "get",
            url: "/trade/list",
            data: {ano:ano},
            dataType: "json",
            success: function(data){
                var template = Handlebars.compile($("#temp").html());
                $("#tbl").html(template(data));
            }
        });
    }
</script>
```

06. Form과 Ajax를 이용한 파일 업로드

- **파일 업로드를 위한 설정**

1) pom.xml 파일에 아래 라이브러리를 추가한다.

pom.xml

```
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.1</version>
</dependency>
```

2) servlet-context.xml 파일에 아래 내용을 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[spring]-[appServlet] servlet-context.xml [c:/zzz/upload] 폴더를 미리 생성한다.

```
<beans:bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <beans:property name="maxUploadSize" value="10485760"/>
</beans:bean>
<beans:bean id="uploadPath" class="java.lang.String">
    <beans:constructor-arg value="c:/upload/"/>
</beans:bean>
```

- **Form, Ajax 방식의 파일 업로드**

1) UploadController를 생성한 후 uploadForm 메서드를 작성한다.

[src/main/java]-[com.example.controller] UploadController.java

```
@Controller
public class UploadController {
    @Resource(name="uploadPath")
    private String path;

    @RequestMapping("/uploadForm")
    public void uploadForm() {

    }

    @RequestMapping(value="/uploadForm", method=RequestMethod.POST)
    public void uploadFormPost(MultipartFile file)throws Exception {
        //@RequestParam("file") MultipartFile[] files
        //여러 개의 파일을 업로드 하는 경우  for(MultipartFile file:files){ ... }
        System.out.println("파일사이즈:" + file.getSize());
        System.out.println("파일타입:" + file.getContentType());
        System.out.println("파일명:" + file.getOriginalFilename());
        UUID uid= UUID.randomUUID();
        String savedName= uid.toString() + "_" + file.getOriginalFilename();
        File target = new File(path, savedName);
        FileCopyUtils.copy(file.getBytes(), target);
    }
```

2) uploadForm.jsp 파일을 작성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views] uploadForm.jsp

```
<body>
    <h1>[파일업로드 Form]</h1>
    <form id="frm" action="uploadForm" method="post" enctype="multipart/form-data">
        <input type="file" name="file"> <!-- 여러개의 파일을 선택하는 경우 multiple -->
        <input type="submit" value="저장">
    </form>
</body>
```

3) UploadController 파일에 uploadAjax 메서드를 작성한다.

[src/main/java]-[com.example.controller] UploadController.java

```
@RequestMapping("/uploadAjax")
public String uploadAjax() {
    return "/uploadAjax";
}

@ResponseBody  //파일 업로드
@RequestMapping(value= "/uploadFile", method=RequestMethod.POST, produces="text/plain;charset=UTF-8")
public String uploadAjaxPost(MultipartFile file) throws Exception {
    UUID uid= UUID.randomUUID();
    String fullName= uid.toString() + "_" + file.getOriginalFilename();
    File target = new File(path, fullName);
    FileCopyUtils.copy(file.getBytes(), target);
    return fullName;
}

@ResponseBody  //파일 삭제
@RequestMapping(value="/deleteFile", method=RequestMethod.POST)
public void deleteFile(String fullName){
    new File(path + "/" + fullName).delete();
}

@ResponseBody  //파일 다운로드
@RequestMapping(value="/downloadFile")
public ResponseEntity<byte[]> downloadFile(String fullName)throws Exception {
    ResponseEntity<byte[]> entity=null;
    FileInputStream in=null;
    try{
        in= new FileInputStream(path + "/" + fullName);
        fullName = fullName.substring(fullName.indexOf("_") + 1);
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_OCTET_STREAM);
        headers.add("Content-Disposition","attachment;filename=\""+new String(fullName.getBytes("UTF-8"),"ISO-8859-1")+"\"");
        entity = new ResponseEntity<byte[]>(IOUtils.toByteArray(in), headers, HttpStatus.CREATED);
    }catch(Exception e){
        entity= new ResponseEntity<byte[]>(HttpStatus.BAD_REQUEST);
    }finally{
        in.close();
    }
    return entity;
}

@ResponseBody  //이미지파일 출력
@RequestMapping("/displayFile")
public byte[] display(String fullName) throws Exception {
    File file=new File(path + fullName);
    return FileCopyUtils.copyToByteArray(file);
}
```

4) uploadAjax.jsp 파일을 작성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views] uploadAjax.jsp

```
<head>
    <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>
    <style>
        small { cursor:pointer; color:blue; font-weight:bold; }
        #upload { width:500px; border:1px solid; margin-bottom:10px; padding:5px; }
        #uploaded { width:510px; border:1px solid; }
    </style>
</head>

<body>
    <h1>[파일업로드 Ajax]</h1>
    <div id="upload"><input type="file" id="file"></div>
    <div id="uploaded"><ul id="uploadedFiles"></ul></div>
</body>

<script>
    $("#file").on("change", function(){
        var file = $("#file")[0].files[0];
        if(file == null) return;

        var formData = new FormData();
        formData.append("file", file);
        $.ajax({
            type:"post",
            url:"/uploadFile",
            processData: false,
            contentType: false,
            data: formData,
            success:function(data) {
                var str = "<li>";
                var idx=data.lastIndexOf(".");
                var fileType=data.substring(idx+1).toLowerCase();
                if(fileType=="png" || fileType=="gif" || fileType=="bmp" || fileType=="jpg" || fileType=="jpeg") {
                   str += "<img src='displayFile?fullName=" + data + "' width=50>";
                }
                str += "<a href='downloadFile?fullName=" + data + "'>" + file.name + "</a>";
                str += "<small fullName='"+ data  + "'>x</small>"
                str += "</li>"
                $("#uploadedFiles").append(str);
            }
        });
    });

    $("#uploadedFiles").on("click", "li small", function() {
        var li=$(this).parent();
        var fullName=$(this).attr("fullName");
        $.ajax({
            type:"post",
            url:"/deleteFile",
            data:{"fullName":fullName},
            success:function(){
                li.remove();
            }
        });
    });
</script>
```

- **게시물에 파일 첨부 처리**

1) 게시물 등록과 첨부파일의 데이터베이스 처리

tab_attach 테이블 생성 SQL

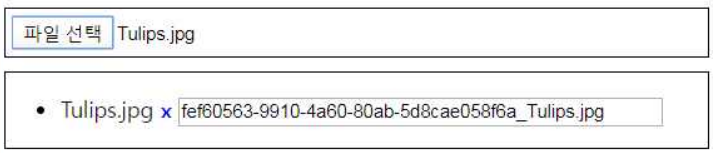
```
create table tbl_attach(
    fullname varchar(150) not null,
    bno int not null,
    regdate datetime default now(),
    primary key(fullname),
    foreign key(bno) references tbl_board(bno)
);
```

- **BoardVO에 첨부 파일명을 저장하기 위한 필드를 추가한다.**

[src/main/java]-[com.example.domain] BoardVO.java

```
private String[] files;
//setter(), getter(), toString() 메서드 새롭게 추가
```

- **게시물 등록 화면에 파일 첨부 부분을 추가한다.**



[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] insert.jsp

```
<style>
    small{
        cursor:pointer;
        color:blue;
        font-weight:bold; }
    #upload{
        width:500px;
        border:1px solid;
        margin-bottom:10px;
        padding:5px; }
    #uploaded{
        width:510px;
        border:1px solid; }
</style>

<div class="page">
    <h1>[게시판 글쓰기]</h1>
    <form name="frm" action="insert" method="post">
        ...
        <div id="upload"><input type="file" id="file"></div>
        <div id="uploaded">
            <ul id="uploadedFiles"></ul>
            <script id="tempFiles" type="text/x-handlebars-template">
                <li>
                    {{fileName}}
                    <small fullName="{{fullName}}">x</small>
                    <input type="text" name="files" value="{{fullName}}" size=50>
                </li>
            </script>
        </div>
    </form>
</div>
```

- **파일 업로드, 삭제 Ajax을 별도의 js 파일로 저장한다.**

[src]-[main]-[webapp]-[resources]-[js] uploadFile.js

```
$(document).ready(function(){
    //파일업로드
    $("#file").on("change", function(){
        var file=$("#file")[0].files[0];
        if(file==null) retun;
        var formData = new FormData();
        formData.append("file", file);
        $.ajax({
            type:"post",
            url:"/uploadFile",
            data:formData,
            processData:false,
            contentType:false,
            success:function(data){
                var temp=Handlebars.compile($("#tempFiles").html());
                tempData = {fullName:data, fileName:file.name}
                $("#uploadedFiles").append(temp(tempData));
            }
        });
    });

    //파일 삭제
    $("#uploadedFiles").on("click", "li small", function(){
        var li=$(this).parent();
        var fullName=$(this).attr("fullName");
        $.ajax({
            type:"post",
            url:"/deleteFile",
            data:{"fullName":fullName},
            success:function(){
                li.remove();
            }
        });
    });
});
```

- **BoardDAO 인터페이스와 BoardMapper xml 파일에 첨부파일 등록 메서드를 추가한다.**

[src/main/java]-[com.example.dao] BoarDAO.java

```
public void addAttach(String fullName);
```

[src/main/resources]-[mapper] BoardMapper.xml

```
<insert id="addAttach">
    insert into tbl_attach(fullName, bno) values(#{fullName}, last_insert_id())
</insert>
```

[src/main/java]-[com.example.dao] BoarDAOImpl.java

```
@Repository
class BoardDAO implements BoardDAO{
    @Autowired
    SqlSession session;
    String namespace = "com.example.mapper.BoardMapper";
    ...
    public void addAttach(String fullName) {
        session.insert(namespace + ".addAttach", fullName);
    }
}
```

- BoardService 인터페이스에 insert 메서드를 추가한다.

[src/main/java]-[com.example.service] BoardService.java

```
public void insert(BoardVO vo);
```

[src/main/java]-[com.example.service] BoardServiceImpl.java

```
@Transactional
@Override
public void insert(BoardVO vo) {
    boardMapper.insert(vo);
    String[] files=vo.getFiles();
    if(files == null) return;
    for(String fullName:files) {
        boardMapper.addAttach(fullName);
    }
}
```

- BoardController insert 메서드를 수정한다.

[src/main/java]-[com.example.controller] BoardController.java

```
@RequestMapping(value="/insert", method=RequestMethod.POST)
public String insertPost(BoardVO vo, RedirectAttributes rttr) {
    rttr.addFlashAttribute("result", "SUCCESS");
    boardService.insert(vo);
    return "redirect:list";
}
```

2) 게시물 조회와 첨부파일의 데이터베이스 처리

- 첨부파일 목록을 출력하기 위해 BoardDAO 인터페이스와 BoardMapper xml 파일에 getAttach 메서드를 추가한다.

[src/main/java]-[com.example.DAO] BoardDAO.java

```
public List<String> getAttach(int bno);
```

[src/main/resources]-[mapper] BoardMapper.xml

```
<select id="getAttach" resultType="string">
    select fullName from tbl_attach where bno = #{bno}
</select>
```

[src/main/java]-[com.example.dao]-BoarDAOImpl.java

```
@Repository
public void BoardDAO implements BoardDAO{
    ...
    public List<String> getAttach(int bno) {
        reeturn session.selectList(namespace + ".getAttach", bno);
    }
}
```

- BoardController 파일에 첨부파일 출력을 위한 메서드를 추가한다.

[src/main/java]-[com.example.controller] BoardController.java

```
@ResponseBody
@RequestMapping(value="/getAttach", method=RequestMethod.GET, produces="application/json;charset=UTF-8")
public List<String> getAttach(int bno) throws Exception{
    return boardDAO.getAttach(bno);
}
```

- read.jsp 파일에 첨부파일 출력 영역을 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] read.jsp

```
<style>
    small{cursor:pointer; color:blue; font-weight:bold;}
    #upload{width:500px; border:1px solid; margin-bottom:10px; padding:5px; margin-top:10px;}
    #uploaded{width:510px; border:1px solid;}
</style>
<div class="page">
    <form name="frm">
        ...
        <div id="upload"><input type="file" id="file"></div>
        <div id="uploaded">
            <ul id="uploadedFiles"></ul>
            <script id="tempFiles" type="text/x-handlebars-template">
                <li><a href="/downloadFile?fullName={{fullName}}">{{fileName}}</a></li>
            </script>
        </div>
    </form>
    ...
</div>
<script>
    var bno="${vo.bno}";
    getAttach();
    ...
    //첨부파일명 읽어오기
    function getAttach(){
        $.ajax({
            type: "get",
            url: "/getAttach",
            data: {"bno":bno},
            success:function(data){
                var temp=Handlebars.compile($("#tempFiles").html());
                $(data).each(function(){
                    var idx=this.indexOf("_") + 1;
                    var fileName = this.substring(idx);
                    tempData = {"fullName":this, "fileName":fileName};
                    $("#uploadedFiles").append(temp(tempData));
                });
            }
        });
    }
</script>
```

3) 게시물의 수정과 첨부 파일의 데이터베이스 처리

[src/main/java]-[com.example.DAO] BoardDAO.java

```
public void replaceAttach(String fullName, String bno);
public void deleteAttach(int bno);
```

[src/main/resources]-[mapper] BoardMapper.xml

```
<insert id="replaceAttach">
    insert into tbl_attach(fullName, bno) values(#{fullName}, #{bno})
</insert>
<delete id="deleteAttach">
    delete tbl_attach where bno = #{bno}
</delete>
```

[src/main/java]-[com.example.DAO] BoardDAOImpl.java

```
public void replaceAttach(String fullName, String bno) {
    HashMap<String, Object> map=new HashMap<>();
    map.put("fullName, fullName);
    map.put("bno", bno);
    session.insert(namespace + ".replaceAttach", map);
}

public void deleteAttach(String bno) {
    session.insert(namespace + ".deleteAttach", bno);
}
```

[src/main/java]-[com.example.service] BoardService.java

```
public void update(BoardVO vo);
```

[src/main/java]-[com.example.service] BoardServiceImpl.java

```
@Transactional
@Override
public void update(BoardVO vo){
    boardMapper.update(vo);

    int bno=vo.getBno();
    boardMapper.deleteAttach(bno);
    String[] files=vo.getFiles();
    if(files == null) {
        return;
    }

    for(String fullName: files){
        boardMapper.replaceAttach(fullName, bno);
    }
}
```

[src/main/java]-[com.example.controller] BoardController.java

```
@RequestMapping(value="/update", method=RequestMethod.POST)
public String  updatePost(BoardVO vo, RedirectAttributes rttr, Criteria cri) {
    boardService.update(vo);
}
```

4) 게시물의 삭제와 첨부 파일의 데이터베이스 처리

[src/main/java]-[com.example.service] BoardService.java

```
public void delete(int bno);
```

[src/main/java]-[com.example.service] BoardServiceImpl.java

```
@Transactional
@Override
public void delete(int bno) {
    boardMapper.deleteAttach(bno);
    boardMapper.delete(bno);
}
```

[src/main/java]-[com.example.controller] BoardController.java

```
@RequestMapping(value="/delete", method=RequestMethod.POST)
public String  deletePost(int bno, RedirectAttributes rttr, Criteria cri) {
    boardService.delete(bno);
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] update.jsp

```
<script>
    $("#btnDelete").click(function() {
        if(!confirm("삭제하시겠습니까?")) return false;
        var total=$("#total").html();
        if(total > 0) { alert("댓글이 달린 게시물을 삭제할 수 없습니다."); return; }

        $("#uploadedFiles li small").each(function(){
            var fullName=$(this).attr("fullName");
            $.ajax({
                type:"post",
                url:"/deleteFile",
                data:{"fullName":fullName},
                success:function(){ }
            });
        });

       frm.action="delete";
       frm.submit();
    });
</script>
```

- **상품관리에 파일 첨부 처리**

1) 상품 관리를 위한 테이블을 생성하고 Sample 데이터를 입력한다.

tbl_product 테이블 생성 SQL

```
create table tbl_product(
    pcode char(4) primary key,
    pname varchar(200) not null,
    price int,
    image varchar(200)
);
```

tbl_product에 Sample 데이터 입력 SQL

```
insert into tbl_product(pcode, pname, price, image)
values values('P1001', '명태냉면1인분', 8700, 'product/img01.jpg');
insert into tbl_product(pcode, pname, price, image)
values values('P1002', '오징어볶음1인분', 8900, 'product/img02.jpg');
insert into tbl_product(pcode, pname, price, image)
values values('P1003', '채끝 찹 스테이크', 9100, 'product/img03.jpg');
```

2) ProductVO를 생성한다.

[src/main/java]-[com.example.domain] ProductVO.java

```
public class ProductVO {
    private String pcode;
    private String pname;
    private int price;
    private String image;
    //setter(), getter(), toString() 메서드 추가
}
```

3) 상품 등록 프로그램을 작성한다.

[src/main/java]-[com.example.dao] ProductDAO.java

```
public interface ProductDAO {
    public void insert(ProductVO vo);
}
```

[src/main/resources]-[mapper] ProductMappre.xml

```
<mapper namespace="com.example.mapper.ProductMapper">
    <insert id="insert">
        insert into tbl_product(pcode, pname, price, image)
        values(#{pcode}, #{pname}, #{price}, #{image})
    </insert>
</mapper>
```

[src/main/java]-[com.example.DAOImpl] ProductDAOImpl.java

```
@Repository
public class ProductDAOImpl implements ProductDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.ProductMapper";

    @Override
    public void insert(ProductVO vo) {
        session.insert(namespace + ".insert", vo);
    }
}
```

[src/main/java]-[com.example.controller] ProductController.java

```
@Controller
public class ProductController {
    @Resource(name="uploadPath")
    private String path;

    @Autowired
    ProductDAO dao;

    @RequestMapping("insert")
    public String insert() {
        return "/home";
    }

    @RequestMapping(value="insert", method=RequestMethod.POST)
    public String insertPost(ProductVO vo, MultipartHttpServletRequest multi) throws Exception {
        MultipartFile file = multi.getFile("file");

        if(!file.isEmpty()) {  //대표이미지가 있으면 업로드
            String image = "product/" + System.currentTimeMillis() +"_" + file.getOriginalFilename();
            file.transferTo(new File(path + image));
            vo.setImage(image);
        }
        mapper.insert(vo);
        return "redirect:/list";
    }

    //이미지파일 브라우저에 출력
    @RequestMapping("/display")
    @ResponseBody
    public ResponseEntity<byte[]> display(String fileName) throws Exception {
        ResponseEntity<byte[]> result=null;

        if(!fileName.equals("")) {  //display fileName이 있는 경우
            File file=new File(path + fileName);
            HttpHeaders header=new HttpHeaders();
            header.add("Content-Type", Files.probeContentType(file.toPath()));
            result=new ResponseEntity<>(FileCopyUtils.copyToByteArray(file), header, HttpStatus.OK);
        }
        return result;
    }
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[product] insert.jsp

```
<div class="page">
    <form name="frm" action="insert" method="post" enctype="multipart/form-data">
        <table border=1>
            <tr>
                <td width=100>상품코드</td><td width=400><input type="text" name="pcode"></td>
            </tr>
            <tr>
                <td>상품이름</td><td><input type="text" name="pname"></td>
            </tr>
            <tr>
                <td>상품가격</td><td><input type="text" name="price" value="0"></td>
            </tr>
            <tr>
                <td>대표이미지</td>
                <td>
                    <img src="https://via.placeholder.com/150x120" id="image" width=150/>
                    <input type="file" name="file"/>
                </td>
            </tr>
        </table>
        <div>
            <input type="submit" value="저장">
            <input type="reset" value="취소">
        </div>
    </form>
</div>
<script>
    //대표 이미지 미리보기
    $("#image").on("click", function(){
        $(frm.file).click();
    });

    $(frm.file).on("change", function(e){
        //var file=$(frm.file)[0].files[0];
        $("#image").attr("src", URL.createObjectURL(e.target.files[0]));
    });
</script>
```

4) 상품 목록 출력 프로그램을 작성한다.

[src/main/java]-[com.example.dao] ProductDAO.java

```
public interface ProductDAO {
    public void insert(ProductVO vo);
    public List<ProductVO> list();
}
```

[src/main/resources]-[mapper] ProductMappre.xml

```
<mapper namespace="com.example.mapper.ProductMapper">
    <select id="list" resultType="com.example.domain.ProductVO">
        select * from tbl_product
    </select>
</mapper>
```

[src/main/java]-[com.example.DAOImpl] ProductDAOImpl.java

```
@Repository
public class ProductDAOImpl implements ProductDAO {
    ...
    @Override
    public ProductVO list() {
        session.insert(namespace + ".list");
    }
}
```

[src/main/java]-[com.example.controller] ProductController.java

```
@RequestMapping("list")
public void list(Model model){
    model.addAttribute("list", dao.list());
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[product] list.jsp

```
<div>
    <div style="width:480px;background:yellow;overflow:hidden;">
    <c:forEach items="${list}" var="vo">
        <div class="box">
            <c:if test="${vo.image!=null && vo.image!=''}">
                <img src="display?fileName=${vo.image}" width=100>
            </c:if>
            <c:if test="${vo.image==null || vo.image==''}">
                <img src="https://via.placeholder.com/100x80" width=100>
            </c:if>
            <div>
                <a href="read?pcode=${vo.pcode}">${vo.pcode}</a>
            </div>
            <div>${vo.pname}</div>
            <div>${vo.price}원</div>
        </div>
    </c:forEach>
</div>
```

5) 상품 정보 읽기 페이지를 작성한다.

[src/main/java]-[com.example.dao] ProductDAO.java

```
public ProductVO read(String pcode);
```

[src/main/resources]-[mapper] ProductMappre.xml

```
<select id="read" resultType="com.example.domain.ProductVO">
    select * from tbl_product
    where pcode = #{pcode}
</select>
```

[src/main/java]-[com.example.DAOImpl] ProductDAOImpl.java

```
@Repository
public class ProductDAOImpl implements ProductDAO {
    ...
    @Override
    public ProductVO read(String pcode) {
        session.insert(namespace + ".read", pcode);
    }
}
```

[src/main/java]-[com.example.controller] ProductController.java

```
@RequestMapping("/read")
public void read(String pcode, Model model) {
    model.addAttribute("vo", mapper.read(pcode));
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[product] read.jsp

```
<form name="frm" action="update" method="post" enctype="multipart/form-data">
    ...
    <td>
        <c:if test="${vo.image!=null && vo.image!=''}">
            <img src="display?fileName=${vo.image}" width=150>
        </c:if>
        <c:if test="${vo.image==null || vo.image==''}">
            <img src="http://placehold.it/150x120" width=150>
        </c:if>
        <input type="hidden" name="image" value="${vo.image}"/>
        <input type="file" name="file"/>
    </td>
    ...
    <input type="submit" value="저장">
    <input type="reset" value="취소">
    <input type="button" value="삭제" id="btnDelete">
    <input type="button" value="목록" onClick="location.href='list'">
</form>
<script>
    ...
    $("#btnDelete").on("click", function(){
        if(!confirm("삭제하실래요?")) return;
        frm.action="delete";
        frm.submit();
    });
</script>
```

6) 상품 정보 수정 삭제 프로그램을 작성한다.

[src/main/java]-[com.example.dao] ProductDAO.java

```
public void update(ProductVO vo);
public void delete(String pcode);
```

[src/main/resources]-[mapper] ProductMappre.xml

```
<update id="update">
    update tbl_product
    set pname=#{pname}, price=#{price}, image=#{image}
    where pcode=#{pcode}
</update>
<delete id="delete">
    delete from tbl_product where pcode=#{pcode}
</delete>
```

[src/main/java]-[com.example.DAOImpl] ProductDAOImpl.java

```
@Repository
public class ProductDAOImpl implements ProductDAO {
    ...
    @Override
    public void update(ProductVO vo) {
        session.insert(namespace + ".update", vo);
    }

    @Override
    public void delete(String pcode) {
        session.delete(namespace + ".delete", pcode);
    }
}
```

[src/main/java]-[com.example.controller] ProductController.java

```
@RequestMapping(value="update", method=RequestMethod.POST)
public String update(ProductVO vo, MultipartHttpServletRequest multi)throws Exception {
    MultipartFile file = multi.getFile("file");
    if(!file.isEmpty()) {  //수정할 대표이미지가 있으면 새로운 이미지로 업로드
        String oldImage=vo.getImage();
        if(!oldImage.equals("")) new File(path + File.separator + oldImage).delete();  //기존 이미지가 있으면 삭제
        String image = System.currentTimeMillis() + file.getOriginalFilename();
        file.transferTo(new File(path + File.separator + image));
        vo.setImage(image);
    }
    mapper.update(vo);
    return "redirect:list";
}

@RequestMapping(value="delete", method=RequestMethod.POST)
public String delete(ProductVO vo) {
    String image=vo.getImage();
    if(!image.equals("")) new File(path + File.separator + image).delete();  //기존 이미지가 있으면 삭제
    mapper.delete(vo.getPcode());
    return "redirect:list";
}
```

7) 상품 첨부파일들을 입력하는 프로그램을 작성한다.

tbl_attach 테이블 생성 SQL

```
create table tbl_attach(
    image varchar(150) not null primary key,
    pcode char(4)  not null,
    regdate datetime default now(),
    foreign key(pcode) references tbl_product(pcode)
);
```

[src/main/java]-[com.example.domain] ProductVO.java

```
public class ProductVO {
    ...
    private ArrayList<String> images;
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[product] insert.jsp

```
    ...
    <td class="title"><input type="button" id="btnImage"  value="상품이미지"></td>
    <td>
        <input type="file" name="files" accept="image/*" multiple>
        <div id="listFile"></div>
    </td>
    ...
<script>
    $("#btnImage").on("click", function() {
         $(frm.files).click();
    });

    $(frm.files).change(function() {  //이미지 미리보기
        var files =$(frm.files)[0].files;
        var html = "";
        $.each(files, function(index, file) {
            html += "<img src='" + URL.createObjectURL(file) + "'>";
            $('#listFile').html(html);
        });
    });
</script>
```

[src/main/java]-[com.example.dao]-[ProductDAO.java]

```
public void addAttach(String image, String pcode);
```

[src/main/resources]-[mapper]-[ProductMappre.xml]

```
<insert id="addAttach">
    insert into tbl_attach(image, pcode) values(#{image}, #{pcode})
</insert>
```

[src/main/java]-[com.example.DAOImpl] ProductDAOImpl.java

```
public void tbl_attach(String image String pcode) {
    HashMap<String, Object> map = new HahsMap<>();
    map.put("image", image);
    map.put("pcode", pcode);
    session.insert(namespace + ".addAttach", map);
}
```

[src/main/java]-[com.example.service]-[ProductService.java]

```
public interface ProductService {
    public void insert(ProductVO vo);
}
```

[src/main/java]-[com.example.service]-[ProductServiceImpl.java]

```
@Service
public class ProductServiceImpl implements ProductService{
    @Autowired
    ProductMapper dao;

    @Transactional
    @Override
    public void insert(ProductVO vo) {
        dao.insert(vo);
        //첨부파일이 있으면 insert
        ArrayList<String> images=vo.getImages();
        if(images.size() > 0) {
            for(String image:images) {
                mapper.addAttach(image, vo.getPcode());
            }
        }
    }
}
```

[src/main/java]-[com.example.controller] ProductController.java

```
@RequestMapping(value="insert", method=RequestMethod.POST)
public String insertPost(ProductVO vo, MultipartHttpServletRequest multi) throws Exception {
    ...
    List<MultipartFile> files = multi.getFiles("files");
    ArrayList<String> images=new ArrayList<String>();

    for (MultipartFile attFile:files) {
        if(!attFile.isEmpty()) {
            String image=System.currentTimeMillis() + attFile.getOriginalFilename();
            attFile.transferTo(new File(path + File.separator + image));
            images.add(image);
        }
    }
    vo.setImages(images);
    service.insert(vo);
    return "redirect:/list";
}
```

8) 상품 첨부파일들을 출력하는 프로그램을 작성한다.

[src/main/java]-[com.example.dao] ProductDAO.java

```
public List<String> getAttach(String pcode);
```

[src/main/resources]-[mapper] ProductMappre.xml

```
<select id="getAttach" resultType="string">
    select image from tbl_attach where pcode = #{pcode}
</select>
```

[src/main/java]-[com.example.dao] ProductDAOImpl.java

```
public List<String> getAttach(String pcode) {
    session.selectList(namespace + ".getAttach", pcode);
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[product] read.jsp

```
<form name="frm" action="update" method="post" enctype="multipart/form-data">
    ...
    <td class="title"><input type="button" id="btnImage" value="상품이미지"></td>
    <td>
        <input type="file" name="files" accept="image/*" multiple>
        <div id="listFile">
            <c:forEach items="${list}" var="image"><img src="/display?fileName=${image}"></c:forEach>
        </div>
    </td>
    ...
</form>
```

[src/main/java]-[com.example.controller] ProductController.java

```
@RequestMapping("/read")
public void read(String pcode, Model model){
    model.addAttribute("vo", mapper.read(pcode));
    model.addAttribute("list", mapper.getAttach(pcode));
}
```

9) 상품 첨부파일들을 수정하는 프로그램을 작성한다.

[src/main/java]-[com.example.dao] ProductDAO.java

```
public void delAttach(String pcode);
```

[src/main/resources]-[mapper] ProductMappre.xml

```
<delete id="delAttach">
    delete from tbl_attach
    where pcode=#{pcode}
</delete>
```

[src/main/java]-[com.example.dao] ProductDAOImpl.java

```
public List<String> delAttach(String pcode) {
    session.delete(namespace + ".delAttach", pcode);
}
```

[src/main/java]-[com.example.service] ProductService.java

```
public void update(ProductVO vo);
```

[src/main/java]-[com.example.service] ProductServiceImpl.java

```
@Transactional
public void update(ProductVO vo) {
    dao.update(vo);
    ArrayList<String> images=vo.getImages();

    if(images.size() > 0) {
        dao.delAttach(vo.getPcode());
        for(String image:images) {
            dao.addAttach(image, vo.getPcode());
        }
    }
}
```

[src/main/java]-[com.example.controller] ProductController.java

```
@RequestMapping(value="update", method=RequestMethod.POST)
public String update(ProductVO vo, MultipartHttpServletRequest multi) throws Exception {
    ...
    List<MultipartFile> files = multi.getFiles("files");
    ArrayList<String> images=new ArrayList<String>();
    for (MultipartFile attFile:files) {
        if(!attFile.isEmpty()) {
            if(images.size()==0) {  //기존 이미지가 있으면 처음 한번만 삭제
                List<String> oldImages=dao.getAttach(vo.getPcode());
                for(String oldImage:oldImages) {
                    new File(path + File.separator + oldImage).delete();
                }
             }
            String image=System.currentTimeMillis() + attFile.getOriginalFilename();
            attFile.transferTo(new File(path + File.separator + image));
            images.add(image);
        }
    }
    vo.setImages(images);
    service.update(vo);
    return "redirect:list";
}
```

10) 상품 삭제 프로그램을 수정 한다.

[src/main/java]-[com.example.service] ProductService.java

```
public void delete(String pcode);
```

[src/main/java]-[com.example.service] ProductService.java

```
@Transactional
public void delete(String pcode) {
    mapper.delAttach(pcode);
    mapper.delete(pcode);
}
```

[src/main/java]-[com.example.service] ProductServiceImpl.java

```
@RequestMapping(value="delete", method=RequestMethod.POST)
public String delete(ProductVO vo){
    String image=vo.getImage();
    if(!image.equals("")) new File(path + File.separator + image).delete();

    List<String> oldImages=dao.getAttach(vo.getPcode());
    for(String oldImage:oldImages) {
         new File(path + File.separator + oldImage).delete();
    }
    service.delete(vo.getPcode());
    return "redirect:list";
}
```

7. 인터셉터를 활용한 로그인 처리

- **로그인 프로그램 작성**

1) 사용자 테이블을 생성하고 샘플 데이터를 입력한다.

tbl_user 테이블 생성

```
create table tbl_user(
    uid varchar(50) not null,
    upass varchar(50) not null,
    uname varchar(100) not null,
    upoint int not null,
    primary key(uid)
);
```

tbl_user 테이블에 데이터 입력

```
insert into tbl_user(uid, upass, uname) values('user01', 'pass', '홍길동');
insert into tbl_user(uid, upass, uname) values('user02', 'pass', '강감찬');
insert into tbl_user(uid, upass, uname) values('user03', 'pass', '이순신');
insert into tbl_user(uid, upass, uname) values('user04', 'pass', '심청이');
```

2) UserVO를 생성한다.

[src/main/java]-[com.example.domain] UserVO.java

```
public class UserVO {
    private String uid;
    private String upass;
    private String uname;
    private int upoint;
    //setter(), getter(), toString() 메서드 추가
}
```

3) 사용자 로그인을 위한 UserMapper 인터페이스와 UserMapper xml에 login 메서드를 작성한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<mapper  namespace="com.example.mapper.UserMapper">
    <select id="read" resultType="com.example.domain.UserVO">
        select * from tbl_user where uid=#{uid}
    </select>
</mapper>
```

[src/main/java]-[com.example.dao] UserDAO.java

```
public interface UserDAO {
    public UserVO read(String uid);
}
```

[src/main/java]-[com.example.dao] UserDAOImpl.java

```
@Repository
public class UserDAOImpl implements UserDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.UserMapper";

    @Override
    public UserVO read(String uid) {
        return session.selectOne(namespace + ".read", uid);
    }
}
```

4) UserController를 작성한다.

[src/main/java]-[com.example.controller] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    UserDAO dao;

    @RequestMapping("/login")
    public String login(Model model) {
        model.addAttribute("pageName", "user/login.jsp");
        return "/home";
    }

    @RequestMapping(value="/login", method=RequestMethod.POST)
    @ResponseBody
    public int loginPost(String uid, String upass, HttpSession session) {
        int result=0;  //아이디가 존재하지 않는 경우

        UserVO vo = dao.read(uid);
        if(vo != null){
            if(vo.getUpass().equals(upass)) {
                result = 1;  //로그인 성공한 경우
                session.setAttribute("uid", uid);  //session에 로그인 정보 저장
            }else {
                resut = 2;  //비밀번호가 일치하지 않는 경우
            }
        }
        return result;
    }
}
```

5) login.jsp 페이지를 작성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] login.jsp

```
<div class="page'>
    <form name="frm">
        <table width=300 border=1>
            <tr><td>아이디</td><td><input type="text" name="uid" size=10></td></tr>
            <tr><td>비밀번호</td><td><input type="password" name="upass" size=10></td></tr>
            <tr><td colspan=2><input type="submit" value="로그인"></td></tr>
        </table>
    </form>
</div>
<script>
    $(frm).on("submit", function(e) {
        e.preventDefault();
        var uid=$(frm.uid).val();
        var upass=$(frm.upass).val();
        $.ajax({
            type: "post",
            url: "/login",
            data: {uid:uid, upass:upass},
            success: function(data) {
                if(data==0) {
                    alert("아이디가 존재하지 않습니다!");
                }else if(data==2) {
                    alert("비밀번호가 일치하지 않습니다!")
                }else {
                    location.href="/";
                }
            }
        });
    });
</script>
```

6) 게시글 목록 화면에 로그인 페이지를 연결한다.

[src]-[main]-[webapp]-[WEB-INF]-[views] home.jsp

```
<div id="menu">
    <c:if test="${uid != null}">
        ${uid}님<button onClick="location.href='/user/logout'">로그아웃</button>
    </c:if>
    <c:if test="${uid == null}">
        <button onClick="location.href='/user/login'">로그인</button>
    </c:if>
    ...
</div>
```

[src/main/java]-[com.example.controller] UserController.java

```
public class UserController {
    @Autowired
    UserDAO dao;

    @RequestMapping("/login")
    public String login(Model model) {
        model.addAttribute("pageName", "user/isnert");
        return "/home";
    }

    @RequestMapping("/logout")
    public String logout(HttpSession session) {
        session.invalidate();  //session.removeAttribute("uid");
        return "redirect:/";
    }
    ...
}
```

- **인터셉터를 이용한 로그인**

1) LoginInterceptor 파일을 작성한다. (로그인 되지 않았을 경우 로그인 페이지로 이동한다.)

[src/main/java]-[com.example.intercptor] LoginInterceptor.java

```
public class LoginInterceptor extends HandlerInterceptorAdapter {
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)throws Exception {
        if(request.getSession().getAttribute("uid") == null){
            response.sendRedirect("/user/login");
        }
        return super.preHandle(request, response, handler);
    }
}
```

2) servlet-context.xml 파일에 LoginInterceptor를 등록한다. (insert이동시 LoginInterceptor에서 가로채기를 한다.)

[src]-[main]-[webapp]-[WEB-INF]-[spring]-[appServlet] servlet-context.xml

```
    <beans:bean id="LoginInterceptor" class="com.example.interceptor.LoginInterceptor"/>
    <interceptors>
        <interceptor>
            <mapping path="/board/insert"/>
            <beans:ref bean="LoginInterceptor"/>
        </interceptor>
    </interceptors>
```

3) 게시물 등록 페이지에 로그인한 아이디 정보를 작성자에 출력한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] insert.jsp

```
...
<tr>
    <td>작성자</td>
    <td><input type="text" name="writer" value="${uid}" readonly size=10></td>
</tr>
```

4) 게시물 조회 페이지에 대한 Interceptor도 아래와 같이 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[spring]-[appServlet] servlet-context.xml

```
    ...
    <beans:bean id="LoginInterceptor" class="com.example.interceptor.LoginInterceptor"/>
    <interceptors>
        <interceptor>
            <mapping path="/board/insert"/>
            <mapping path="/board/read"/>
        </interceptor>
    </interceptors>
    ...
```

5) 로그인후 이동하고자 했던 주소로 이동하기 위해 주소를 session에 저장해 둔다.

[src/main/java]-[com.example.intercptor] LoginInterceptor.java

```
public class LoginInterceptor extends HandlerInterceptorAdapter{
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)throws Exception {
        if(request.getSession().getAttribute("uid") == null) {
            String path = request.getServletPath();
            String query = request.getQueryString() == null ? "" : "?" + request.getQueryString();
            request.getSession().setAttribute("dest", path + query);

            response.sendRedirect("/user/login");
        }
        return super.preHandle(request, response, handler);
    }
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] login.jsp

```
<script>
    ...
    $.ajax({
        type: "post",
        url: "/user/login",
        data: {uid: uid, upass:upass},
        success: function(data) {
            if(data==0){
                alert("아이디가 존재하지 않습니다!");
            }else if(data==2) {
                alert("비밀번호가 일치하지 않습니다!")
            }else {
                var dest="${dest}";
                if(dest==null || dest=="") dest="/";
                location.href=dest;
            }
        }
    });
    ...
</script>
```

6) 게시물 조회 페이지에서 게시물을 작성한 사용자만 댓글 삭제 수정 버튼이 보이보록 한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[board] read.jsp

```
<script id="temp" type="text/x-handlebars-template">
{{#each list}}
    <div rno="{{rno}}">
        <span class="replyer">{{replyer}}</span>
        <span class="replyDate">{{replyDate}}</span>
        <div class="reply">{{reply}}</div>
        <div style="display:{{display replyer}}">
            <button class="btnUpdate">수정</button>
            <button class="btnDelete">삭제</button>
        </div>
    </div>
{{/each}}
</script>
<script>
    Handlebars.registerHelper("display", function(replyer) {
        if(replyer == "${uid}") return "none";
    });
</script>
```

- **자동 로그인과 쿠키**

1) login.jsp 페이지에 아래 내용을 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] login.jsp

```
<div class="page'>
    <form name="frm">
        <table width=300 border=1>
            ...
            <tr>
                <td colspan=2>
                    <input type="submit" value="로그인">
                    <input type="checkbox" name="chkLogin">로그인 상태 유지
                </td>
            </tr>
        </table>
    </form>
</div>
<script>
    $(frm).on("submit", function(e) {
        ...
        var isLogin = $(frm.chkLogin).is(':checked') ? true : false;
        $.ajax({
            type: "post",
            url: "/user/login",
            data: {uid: uid, upass:upass, isLogin:isLogin},
            success: function(data) {
                if(data==0){
                    alert("아이디가 존재하지 않습니다!");
                }else if(data==2) {
                    alert("비밀번호가 일치하지 않습니다!")
                }else {
                    var dest="${dest}";
                    if(dest==null || dest=="") dest="/";
                    location.href=dest;
                }
            }
        });
    });
</script>
```

---

2) 로그인에 성공하면 쿠키에 로그인 정보를 저장한다.

[크롬]-[개발자 도구]-[Application]-[Cookies]에 쿠키가 생성되는지 확인해 본다.

[src/main/java]-[com.example.controller]-UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    UserDAO dao;

    @RequestMapping(value="/login", method=RequestMethod.POST)
    @ResponseBody
    public int loginPost(String uid, String upass, HttpSession session, boolean isLogin, HttpServletResponse response) {
        int result=0;
        UserVO vo = dao.login(uid);
        if(vo != null){
            if(upass.equals(vo.getUpass())) {
                result = 1;  //로그인 성공한 경우
                session.setAttribute("uid", uid);  //session에 로그인정보 저장
                //로그인 상태유지를 체크한 경우 쿠키에 로그인 정보 저장
                if(isLogin) {
                    Cookie cookie = new Cookie("uid", uid);
                    cookie.setPath("/");
                    cookie.setMaxAge(60*60*24*7);
                    response.addCookie(cookie);
                }
                ...
            }
        }
        return result;
    }

    @RequestMapping("/logout")
    public String logout(HttpSession session, HttpServletRequest request, HttpServletResponse response) {
        session.invalidate();
        Cookie cookie = WebUtils.getCookie(request, "uid");
        //쿠키에 로그인 정보가 저장된 경우 로그인 정보 삭제
        if(cookie != null) {
            cookie.setPath("/");
            cookie.setMaxAge(0);
            response.addCookie(cookie);
        }
        return "redirect:/";
    }
}
```

3) 로그인 정보가 쿠키에 저장되어있는지 HomeController의 home 메서드에서 확인 후 존재하면 Session에 저장한다.

[src/main/java]-[com.example.controller] HomeController.java

```
@Controller
public class HomeController {
    @RequestMapping("/")
    public void home(Model model, HttpServletRequest request) {
        Cookie cookie = WebUtils.getCookie(request, "uid");
        if(cookie != null) {
            request.getSession().setAttribute("uid", cookie.getValue());
        }
    }
}
```

- **네이버 API를 이용한 로그인**

1) 네이버 아이디 로그인을 위한 설정

- [네이버 개발자센터]-[네이버 아이디로 로그인] -[오픈 API 이용신청]을 선택한다.
- [환경추가]-[PC 웹]-[서비스 URL]에 http://localhost:8080을 입력한다.
- [네이버 아이디로 로그인 Callback URL]에 http://localhost:8080/user/loginNaver을 입력한다.

2) 네이버 아이디 로그인을 위한 Documents

- [네이버 개발자센터]-[Documents]-[네이버 아이디로 로그인] 메뉴를 선택한다.
- [API 명세]-[로그인 API명세] 메뉴를 선택한다.
- login.jsp에 APIExamNaverLogin.html 내용을 추가하고 loginNaver.jsp에 callback.html 내용을 추가한다.

1) login.jsp 파일에 네이버 아이디로 로그인하는 버튼을 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] login.jsp

```
<head>
    ...
    <script type="text/javascript" src="https://static.nid.naver.com/js/naverLogin_implicit-1.0.3.js" charset="utf-8"></script>
    <script type="text/javascript" src="http://code.jquery.com/jquery-1.11.3.min.js"></script>
</head>
<body>
    ...
    <div id="naver_id_login"></div>
</body>
<script type="text/javascript">
    var naver_id_login = new naver_id_login("UwWNHZrWWCMIxZajQ_Xz", "http://localhost:8088/user/loginNaver");
    var state = naver_id_login.getUniqState();
    naver_id_login.setButton("green", 3, 40);
    naver_id_login.setDomain(c);
    naver_id_login.setState(state);
    naver_id_login.init_naver_id_login();  //naver_id_login.setPopup(); 팝업창으로 열린다.
</script>
```

2) UserController에 로그인 버튼 클릭 후 이동할 주소를 추가한다.

[src/main/java]-[com.example.controller] UserController.jsp

```
@RequestMapping("loginNaver")
public void loginNaver(){ }
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] login.jsp

```
<script type="text/javascript">
    var naver_id_login = new naver_id_login("UwWNHZrWWCMIxZajQ_Xz", "http://localhost:8088/user/naverLogin");
    // 접근 토큰 값 출력
    //alert(naver_id_login.oauthParams.access_token);
    // 네이버 사용자 프로필 조회
    naver_id_login.get_naver_userprofile("naverSignInCallback()");
    function naverSignInCallback() {  // 네이버 사용자 프로필 조회 이후 프로필 정보를 처리할 callback function
        //alert(naver_id_login.getProfileData('email'));
        //alert(naver_id_login.getProfileData('nickname'));
        //alert(naver_id_login.getProfileData('age'));
        location.href="/user/loginNaverResult?email=" + naver_id_login.getProfileData('email');
    }
</script>
```

3) UserController에 로그인이 성공한 후 이동할 주소를 출력한다.

[src/main/java]-[com.example.controller] UserController.jsp

```
@RequestMapping("loginNaverResult")
public String loginNaverResult(String email, HttpSession session) {
    session.setAttribute("uid", email);
    return "redirect:/board/list";
}
```

- **회원 가입 비밀번호 암호화**

1) 비밀번호 암호화를 위한 설정을 한다.

pom.xml *추가 라이브러리 중 가장 위에 설정한다.

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>4.1.0.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>4.1.0.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>4.1.0.RELEASE</version>
</dependency>
```

[src]-[main]-[webapp]-[WEB-INF]-[spring] security-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/security
       http://www.springframework.org/schema/security/spring-security.xsd">
    <beans:bean id="bcryptPasswordEncoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>
</beans:beans>
```

[src]-[main]-[webapp]-[WEB-INF] web.xml

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

2) UserDAO 인터페이스와 UserDAO.xml 파일에 회원가입을 위한 메서드를 추가한다.

[src/main/java]-[com.example.dao] UserDAO.java

```
public class UserDAO {
    ...
    public void insert(UserVO vo);
}
```

[src/main/resources]-[mapper] BoardMapper.xml

```
<insert id="insert">
    insert into tbl_user(uid, upass, uname)
    values(#{uid}, #{upass}, #{uname})
</insert>
```

[src/main/java]-[com.example.dao] UserDAOImpl.java

```
@Repository
public class UserDAOImpl implements UserDAO {
    ...
    public void insert(UserVO vo) {
        session.insert(namespace + ".insert", vo);
    }
}
```

3) UserController에 회원가입 REST API를 추가한다.

[src/main/java]-[com.example.controller] UserController.java

```
...
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    UserDAO dao;

    @Autowired
    BCryptPasswordEncoder passwordEncoder;

    @RequestMapping("/login")
    public String login(Model model) {
        ...
    }

    @RequestMapping(value="/login", method=RequestMethod.POST)
    @ResponseBody
    public int loginPost(String uid, String upass, HttpSession session) {
        ...
    }

    @RequestMapping("/insert")
    public String insert(Model model) {
        model.addAttribute("pageName", "user/insert.jsp");
        return "/home";
    }

    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public void insertPost(UserVO vo) {
        String password = passEncoder.encode(vo.getUpass());
        vo.setUpass(password);
        dao.insert(vo);
    }
}
```

4) login.jsp 파일에 회원가입 버튼을 추가한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user] login.jsp

```
...
<tr>
    <td colspan=2>
        <div style="font-size:12px;">
            <input type="submit" value="로그인">
            <input type="checkbox" name="chkLogin">로그인 상태 유지
            <a href="insert">회원가입</a>
        </div>
    </td>
</tr>
...
```

5) insert.jsp 파일에 회원가입 페이지를 작성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[user]-insert.jsp

```
<div class="page">
    <h1>회원가입</h1>
    <form name="frm" method="post">
        <table border=1 width=400>
            <tr>
               <td width=150>회원 아이디</td>
               <td><input type="text" name="uid" size=20></td>
            </tr>
            <tr>
                <td width=150>회원 비밀번호</td>
                <td><input type="password" name="upass" size=20></td></tr>
            <tr>
                <td width=150>회원 성명</td>
                <td><input type="text" name="uname" size=20></td>
            </tr>
        </table>
        <div>
            <input type="submit" value="가입">
            <input type="reset" value="취소">
        </div>
    </form>
</div>
```

6) 로그인시 입력한 비밀번호를 암호화하여 비교한다.

[src/main/java]-[com.example.controller] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    UserDAO dao;

    @Autowired
    BCryptPasswordEncoder passwordEncoder;
    ...

    @RequestMapping(value="/login", method=RequestMethod.POST)
    @ResponseBody
    public int loginPost(String uid, String upass, HttpSession session, boolean isLogin, HttpServletResponse response) {
        int result = 0;
        UserVO vo = dao.read(uid);
        if(vo != null) {
            if(passwordEncoder.matches(upass, vo.getUpass())) {
                result = 1;  //로그인 성공한 경우
                session.setAttribute("uid", uid);  //session에 로그인정보 저장
                //로그인 상태유지를 체크한 경우 쿠키에 로그인 정보 저장
                if(isLogin) {
                    Cookie cookie = new Cookie("uid", uid);
                    cookie.setPath("/");
                    cookie.setMaxAge(60*60*24*7);
                    response.addCookie(cookie);
                }
                ...
            }
        }
        return result;
    }
    ...
}
```

08. jsoup과 selenium을 활용한 웹 크롤링

- **jsoup을 이용한 크롤링**

1) pom.xml에 라이브러리를 추가한다.

pom.xml

```
<dependency>
    <groupId>org.jsoup</groupId>
    <artifactId>jsoup</artifactId>
    <version>1.12.1</version>
</dependency>
```

2) CrawlingController에서 네이버 증권 Top종목, CGV 영화차트에 대한 데이터를 크롤링을 한다.

[src/main/java]-[com.example.controller] CrawlingController.java

```
@Controller
@RequestMapping("/crawl")
public class CrawlingController {
    @RequestMapping("/jsoup")
    public String jsoup(Model model){
        model.addAttribute("pageName", "crawl/jsoup.jsp");
        return "/home";
    }

    @RequestMapping("/cgv.json")
    @ResponseBody
    public ArrayList<HashMap<String, Object>> cgv() {
        ArrayList<HashMap<String, Object>> array=new ArrayList<>();
        try{
            Document doc=Jsoup.connect("http://www.cgv.co.kr/movies/?lt=1&ft=0").get();
            Elements elements=doc.select(".sect-movie-chart ol");
            for(Element e:elements.select("li")){
                HashMap<String, Object> map=new HashMap<>();
                map.put("title", e.select(".title").text());
                map.put("image", e.select("img").attr("src"));
                map.put("date", e.select(".txt-info strong").text());
                map.put("link", "http://www.cgv.co.kr" + e.select(".box-image a").attr("href"));
                if(!map.get("title").equals("")) array.add(map);
            }
        }catch(Exception e){ System.out.println("cgv 오류:" + e.toString()); }
        return array;
    }

    @RequestMapping("/finance.json")
    @ResponseBody
    public ArrayList<HashMap<String, Object>> finance() {
        ArrayList<HashMap<String, Object>> array=new ArrayList<>();
        try{
            Document doc=Jsoup.connect("https://finance.naver.com/").get();
            Elements elements=doc.select("#_topItems1 tr");
            for(Element e:elements){
                HashMap<String, Object> map=new HashMap<>();
                map.put("title", e.select("a").text());
                map.put("price", e.select("td").get(0).text());
                map.put("range", e.select("td").get(1).text());
                map.put("rate", e.select("td").get(2).text());
                array.add(map);
            }
        }catch(Exception e) { System.out.println("finance 오류:" + e.toString()); }
        return array;
    }
}
```

3) 크롤링 결과를 출력할 jsoup.jsp 페이지를 작성한다.

[src]-[main]-[webapp]-[WEB-INF]-[views]-[craw] jsoup.jsp

```
<div class="page">
    <table id="tblFinance"></table>
    <script id="tempFinance" type="text/x-handlebars-template">
        <tr>
            <td width=200>{{title}}</td>
            <td width=100>{{price}}</td>
            <td width=100>{{range}}</td>
            <td width=100>{{rate}}</td>
            <td><button id="stop">중지</td>
        </tr>
    </script>

    <table id="tbl"></table>
    <script id="temp" type="text/x-handlebars-template">
        <tr>
            <td colspan=4><button id="download">선택 이미지 다운로드</button></td>
        </tr>
    {{#each .}}
        <tr class="cgv">
            <td><input type="checkbox" class="chk"></td>
            <td class="image">{{image}}</td>
            <td>{{title}}</td>
            <td>{{date}}</td>
        </tr>
    {{/each}}
    </script>
</div>

<script>
    getList();

    function getList(){
        $.ajax({
            type: "get",
            url: "/crawl/cgv.json",
            dataType: "json",
            success: function(data){
                var temp=Handlebars.compile($("#temp").html());
                $("#tbl").html(temp(data));
            }
        });
    }

    $.ajax({  //주식 정보를 2초 간격으로 한 종목씩 출력한다.
        type: "get",
        url: "/crawl/finance.json",
        success: function(data) {
            var temp=Handlebars.compile($("#tempFinance").html());

            var finance = [];
            var i=0;
            $(data).each(function(){
                finance[i]=this;
                i++;
            });

            var i=0;
            var interval = setInterval(function(){
                $("#tblFinance").html(temp(finance[i % finance.length]));
                 i++;
            }, 2000);

           $("#tblFinance").on("click", "#stop", function(){
               clearInterval(interval);
            });
        }
    });
</script>
```

4) 이미지 다운로드를 위한 Controller를 작성한다.

[src/main/java]-[com.example.controller] CrawlingController.java

```
@ResponseBody
@RequestMapping(value="/download", method=RequestMethod.POST)
public void download(String image){
    try {
        String strPath = "c:/zzz/download/";
        String strFile=image.substring(image.lastIndexOf("/") + 1);

        File filePath= new File(strPath);
        if(!filePath.exists()) {
            filePath.mkdir();
        }

        URL url = new URL(image);
        InputStream in = url.openStream();
        OutputStream out = new FileOutputStream(strPath + System.currentTimeMillis() + "_" + strFile);
        FileCopyUtils.copy(in, out);
    }catch(Exception e){
        System.out.println("download 오류:" + e.toString());
    }
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views]-[craw] jsoup.jsp

```
$("#tbl").on("click", "#download", function(){
    if(!confirm("선택한 이미지를 저장하실래요?")) return;

    $("#tbl .cgv .chk:checked").each(function(){
        var chk=$(this);
        var cgv = $(this).parent().parent();
        var image = cgv.find(".image").html();

        $.ajax({
            type: "post",
            url: "/crawl/download",
            data: {image:image},
            success:function(){
                chk.prop("checked", false);
            }
        });
    });
});
```

- 크롤링 실행 결과 페이지

- **selenium을 이용한 크롤링**

1) pom.xml에 selenium을 위한 라이브러리를 추가한다.

pom.xml

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>
```

2) http://chromedriver.chromium.org/downloads 에서 크롬 브라우저 버전 확인 후 다운로드 받는다.

3) CrawlingController에 CGV의 더보기 영화 정보를 크롤링 하기위해 아래 내용을 추가한다.

[src/main/java]-[com.example.controller] CrawlingController.java

```
@Controller
@RequestMapping("/crawl")
public class CrawlingController {
    ...
    @RequestMapping("movie.json")
    @ResponseBody
    public List<Map<String, Object>> movie(){
        List<Map<String, Object>> array=new ArrayList<>();
        try {
            //브라우저가 눈에 보이지 않고 내부적으로 돈다.
            ChromeOptions options = new ChromeOptions();
            options.addArguments("headless");

            //다운로드 받은 ChromeDriver 위치를 넣어주고 드라이버 객체를 생성한다.
            System.setProperty("webdriver.chrome.driver", "c:/chromedriver.exe");
            WebDriver driver = new ChromeDriver(options);

            //이동을 원하는 URL로 접속한다.
            driver.get("http://www.cgv.co.kr/movies/");

            WebElement contents = driver.findElement(By.id("contents"));
            WebElement more = contents.findElement(By.className("link-more"));
            more.click();

            List<WebElement> elements =driver.findElements(By.cssSelector(".sect-movie-chart ol li"));
            System.out.println("사이즈:" + elements.size());

            for(WebElement e:elements){
                Map<String, Object> map=new HashMap<String, Object>();
                String title = e.findElement(By.className("title")).getText();
                String image=e.findElement(By.tagName("img")).getAttribute("src");
                String info=e.findElement(By.cssSelector(".txt-info")).getText();
                map.put("title", title);
                map.put("image", image);
                map.put("info", info);
                array.add(map);
            }

            driver.quit();

        }catch(Exception e) {
            System.out.println("movie 오류: " + e.toString());
        }
        return array;
    }
}
```

4) 네이버 쇼핑에서 노트북관련 상품을 크롤링 한 후 출력한다.

[src/main/java]-[com.example.controller] CrawlingController.java

```
@Controller
@RequestMapping("/crawl")
public class CrawlingController {
    ...
    @RequestMapping("shop.json")
    @ResponseBody
    public List<Map<String, Object>> shop(){
        List<Map<String, Object>> array=new ArrayList<>();
        try {
            ChromeOptions options = new ChromeOptions();
            //options.addArguments("headless");
            options.setPageLoadStrategy(PageLoadStrategy.NORMAL);

            System.setProperty("webdriver.chrome.driver", "c:/driver/chromedriver.exe");
            WebDriver driver = new ChromeDriver(options);
            //System.out.println(driver.getPageSource());
            driver.get("https://shopping.naver.com/home/p/index.naver");

            WebElement frmSearch = driver.findElement(By.name("search"));
            WebElement inputQuery = frmSearch.findElement(By.name("query"));
            WebElement btnSearch = frmSearch.findElement(By.className("co_srh_btn"));
            inputQuery.sendKeys("노트북");
            btnSearch.click();
            Thread.sleep(1000);

            ((JavascriptExecutor)driver).executeScript("window.scrollTo(0, document.body.scrollHeight)");
            Thread.sleep(5000);

            List<WebElement> es=driver.findElements(By.cssSelector(".list_basis .basicList_item__2XT81"));

            for(int i=1; i<=es.size(); i++){
                Map<String, Object> map=new HashMap<String, Object>();

               String path="//*[@id='__next']/div/div[2]/div[2]/div[3]/div[1]/ul/div/";
               WebElement title=driver.findElement(By.xpath( path + "div[" + i + "]/li/div/div[2]/div[1]/a"));
               WebElement price=driver.findElement(By.xpath( path + "div[" + i + "]/li/div/div[2]/div[2]/strong/span/span[2]"));
               WebElement link=driver.findElement(By.xpath(path + "div[" + i + "]/li/div/div[2]/div[2]/strong/a"));

               String strTitle=title.getText();
               String strPrice=price.getText();
               String strLink=link.getAttribute("href");

               map.put("title", strTitle);
               map.put("price", strPrice);
               map.put("link", strLink);

               array.add(map);
            }

            driver.quit();
        }catch(Exception e){
            System.out.println("shop.json 오류:" + e.toString());
        }
        return array;
    }
    ...
}
```

5) 네이버 쇼핑에서 노트북관련 상품을 크롤링 한 후 출력한다.

[src/main/java]-[com.example.controller] CrawlingController.java

```
@Controller
@RequestMapping("/crawl")
public class CrawlingController {
    ...
    @RequestMapping("shop.json")
    @ResponseBody
    public List<Map<String, Object>> shop(){
        List<Map<String, Object>> array=new ArrayList<>();
        try {
            ChromeOptions options = new ChromeOptions();
            //options.addArguments("headless");
            options.setPageLoadStrategy(PageLoadStrategy.NORMAL);

            System.setProperty("webdriver.chrome.driver", "c:/driver/chromedriver.exe");
            WebDriver driver = new ChromeDriver(options);
            //System.out.println(driver.getPageSource());
            driver.get("https://shopping.naver.com/home/p/index.naver");

            WebElement frmSearch = driver.findElement(By.name("search"));
            WebElement inputQuery = frmSearch.findElement(By.name("query"));
            WebElement btnSearch = frmSearch.findElement(By.className("co_srh_btn"));
            inputQuery.sendKeys("노트북");
            btnSearch.click();
            Thread.sleep(1000);

            ((JavascriptExecutor)driver).executeScript("window.scrollTo(0, document.body.scrollHeight)");
            Thread.sleep(5000);

            List<WebElement> es=driver.findElements(By.cssSelector(".list_basis .basicList_item__2XT81"));

            for(int i=1; i<=es.size(); i++){
                Map<String, Object> map=new HashMap<String, Object>();

               String path="//*[@id='__next']/div/div[2]/div[2]/div[3]/div[1]/ul/div/";
               WebElement title=driver.findElement(By.xpath( path + "div[" + i + "]/li/div/div[2]/div[1]/a"));
               WebElement price=driver.findElement(By.xpath( path + "div[" + i + "]/li/div/div[2]/div[2]/strong/span/span[2]"));
               WebElement link=driver.findElement(By.xpath(path + "div[" + i + "]/li/div/div[2]/div[2]/strong/a"));

               String strTitle=title.getText();
               String strPrice=price.getText();
               String strLink=link.getAttribute("href");

               map.put("title", strTitle);
               map.put("price", strPrice);
               map.put("link", strLink);

               array.add(map);
            }

            driver.quit();
        }catch(Exception e){
            System.out.println("shop.json 오류:" + e.toString());
        }
        return array;
    }
    ...
}
```

6) 네이버 쇼핑에서 노트북관련 상품을 크롤링 한 후 출력한다.

[src/main/java]-[com.example.controller] CrawlingController.java

```
@Controller
@RequestMapping("/crawl")
public class CrawlingController {
    ...
    @RequestMapping("/naver.json")
    @ResponseBody
    public List<Map<String, Object>> shop() {
        List<Map<String, Object>> array=new ArrayList<>();
        try {
            ChromeOptions options = new ChromeOptions();
            //options.addArguments("headless");
            options.setPageLoadStrategy(PageLoadStrategy.NORMAL);

            System.setProperty("webdriver.chrome.driver", "c:/driver/chromedriver.exe");
            WebDriver driver = new ChromeDriver(options);
            //System.out.println(driver.getPageSource());
            driver.get("https://shopping.naver.com/home/p/index.naver");

            WebElement frmSearch = driver.findElement(By.name("search"));
            WebElement inputQuery = frmSearch.findElement(By.name("query"));
            WebElement btnSearch = frmSearch.findElement(By.className("co_srh_btn"));
            inputQuery.sendKeys("노트북");
            btnSearch.click();
            Thread.sleep(1000);

            //스크롤바를 마지막으로 이동한다.
            ((JavascriptExecutor)driver).executeScript("window.scrollTo(0, document.body.scrollHeight)");
            Thread.sleep(5000);

            List<WebElement> es=driver.findElements(By.cssSelector(".list_basis .basicList_item__2XT81"));

            for(int i=1; i<=es.size(); i++) {
                Map<String, Object> map=new HashMap<String, Object>();

               String path="//*[@id='__next']/div/div[2]/div[2]/div[3]/div[1]/ul/div/";
               WebElement title=driver.findElement(By.xpath( path + "div[" + i + "]/li/div/div[2]/div[1]/a"));
               WebElement price=driver.findElement(By.xpath( path + "div[" + i + "]/li/div/div[2]/div[2]/strong/span/span[2]"));
               WebElement link=driver.findElement(By.xpath(path + "div[" + i + "]/li/div/div[2]/div[2]/strong/a"));

               String strTitle=title.getText();
               String strPrice=price.getText();
               String strLink=link.getAttribute("href");

               map.put("title", strTitle);
               map.put("price", strPrice);
               map.put("link", strLink);

               array.add(map);
            }
            driver.quit();
        }catch(Exception e) {
            System.out.println("shop.json 오류:" + e.toString());
        }
        return array;
    }
    ...
}
```

7) 다음 쇼핑에서 노트북관련 상품을 크롤링한 후 출력한다.

```
http://localhost:8088/crawl/daum.json?qurey=노트북
```

[src/main/java]-[com.example.controller] CrawlingController.java

```
@Controller
@RequestMapping("/crawl")
public class CrawlingController {
    ...
    @RequestMapping("/daum.json")
    @ResponseBody
    public List<Map<String,Object>> daum(String query){
        List<Map<String,Object>> array= new ArrayList<>();
        try {
            ChromeOptions options=new ChromeOptions();
            //options.addArguments("headless");

            System.setProperty("webdriver.chrome.driver", "c:/data/chromedriver.exe");
            WebDriver driver=new ChromeDriver(options);

            driver.get("https://shoppinghow.kakao.com/search/" + query);

            //1초 기다린 후 실행한다.
            Thread.sleep(1000);

            //스크롤바를 마지막으로 이동한다.
            JavascriptExecutor js=(JavascriptExecutor)driver;
            js.executeScript("window.scrollTo(0, document.body.scrollHeight)");

            //1초 기다린 후 실행한다.
            Thread.sleep(1000);

            List<WebElement> es=driver.findElements(By.cssSelector(".wrap_prod_list .wrap_prod"));

            for(WebElement e:es) {
                Map<String,Object> map=new HashMap<>();

                WebElement title=e.findElement(By.className("link_g"));
                WebElement image=e.findElement(By.className("thumb_img"));
                WebElement price=e.findElement(By.cssSelector(".wrap_price .lowest"));

                String strTitle = title.getText();
                String strImage = image.getAttribute("src");
                String strPrice = price.getText();

                map.put("title", strTitle);
                map.put("image", strImage);
                map.put("price", strPrice);

                array.add(map);
            }
            driver.close();
        }catch(Exception e){
            System.out.println("daum 오류:" + e.toString());
        }
        return array;
    }
    ...
}
```

09. 스프링 부트를 이용한 프로젝트 생성

- **스프링 부트 프로젝트 생성**

1) [Help]-[Eclipse Marketplace]-[Find]에서 'sts'입력 후 Spring Tools(aks Spring Tool Suite4)x.x.x.RELEASE를 설치한다.
2) [File]-[New]-[Other]-[Spring Boot]-[Spring Starter Project]를 선택한다.
3) 프로젝트 생성 시 [Type]을 'Maven'으로 선택하고 Package에는 'com.exmaple.web'로 지정한다.
4) 라이브러리는 아래와 같이 체크한다.

- [Developer tools] 의 'Spring Boot DevTools'
- [Template Engines]의 'Thymleaf'
- [Web]의 'Spring Web'

- **스프링 부트 프로젝트 실행**

1) port를 변경하려면 application.properties 파일에 변경하려는 포트번호를 입력한다. 기본포트는 8080이다.

[src/main/resources] application.properties

```
server.port = 8088
```

2) HomeController를 작성하면 Application 실행 파일에서 Controller 패키지를 스캔한다.

[src/main/java]-[com.example.controller] HomeController.java

```
...
@Controller
public class HomeController {
    @ResponseBody
    @RequestMapping("/hello")
    public String hello() {
        return "Hello World!";
    }
}
```

3) [해당프로젝트]-[Run As]-[Spring Boot App] 메뉴를 선택한다. 스프링 부트는 톰캣서버가 내장되어 있다.
4) 브라우저를 실행한 후 주소에 'http://localhost:8088/hello'을 입력하면 'Hello World!'가 출력된다.

- **스프링 부트의 뷰(view) 처리**

뷰로 사용할 타임리프(Thymeleaf)는 컨트롤러가 전달하는 데이터를 이용해 동적으로 화면을 만들어주는 역할을 하는 뷰 템플릿 엔진이다.
1) HomeController에 루트('/') 매핑을 추가한다.

[src/main/java]-[com.example.controller] HomeController.java

```
...
import org.springframework.ui.Model;
@Controller
public class HomeController {
    @ResponseBody
    @RequestMapping("/hello")
    public String hello() {
        return "Hello World!";
    }

    @RequestMapping("/")
    public String home(Model model) {
        model.addAttribute("pageName", "about");
        return "home";
    }
}
```

2) 타임리프(Thymeleaf)를 이용해 View 페이지를 작성한다.

[src/main/resources]-[templates] header.html

```
<div class="row mb-5 justify-content-center">
    <div class="col">
        <div class="text-center"><img src="https://via.placeholder.com/960x150" width="100%"></div>
        <div class="my-3">
            <span class="pe-3"><a href="/">Home</a></span>
        </div>
    </div>
</div>
```

[src/main/resources]-[templates] about.html

```
<div class="row">
    <div class="col">
        <h1 class="text-center">프로그램소개</h1>
    </div>
</div>
```

[src/main/resources]-[templates] footer.html

```
<div class="row my-5">
    <div class="col">
        <hr>
        <h3 class="text-center">Copyright 2023. 인천일보아카데미</h3>
    </div>
</div>
```

[src/main/resources]-[templates] home.html

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>홈페이지</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
    <link rel="stylesheet" href="/css/style.css"/>
</head>
<body>
    <div class="container">
        <div th:include="header"></div>
        <div th:include="${pageName}"></div>
        <div th:include="footer"></div>
    </div>
</body>
</html>
```

3) 페이지를 위한 스타일 파일을 작성한다.

[src/main/resources]-[static]-[css]-[style.css]

```
@charset "UTF-8";

@font-face {
    font-family: 'GmarketSansMedium';
    src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2001@1.1/GmarketSansMedium.woff') format('woff');
    font-weight: normal;
    font-style: normal;
}

* {
    font-family: 'GmarketSansMedium';
}
```

- **스프링 부트에서의 MySQL 데이터베이스 접속**

1) pom.xml에 Mysql Database와 Mybatis 사용을 위해 아래 라이브러리를 추가한다.

pom.xml

```
<dependencies>
    ...
    <!-- 라이브러리 추가 시작 -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.29</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-starter</artifactId>
        <version>3.0.1</version>
    </dependency>
    <!-- 라이브러리 추가 종료 -->
    ...
</dependencies>
```

build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.1.1'
    id 'io.spring.dependency-management' version '1.1.0'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:3.0.1'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    runtimeOnly("com.mysql:mysql-connector-j")
}

tasks.named('test') {
    useJUnitPlatform()
}
```

2) application.properties 파일에 아래 내용을 추가하고 DatabaseConfig 파일을 작성한다.

[src/main/resources] application.properties

```
server.port = 8088
spring.datasource.hikari.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.jdbc-url=jdbc:mysql://localhost/webdb?serverTimezone=UTC
spring.datasource.hikari.username=web
spring.datasource.hikari.password=pass
```

[src/main/java]-[com.example] MysqlConfig.java

```
package com.example;

import javax.sql.DataSource;
import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.io.Resource;
import org.springframework.core.io.support.PathMatchingResourcePatternResolver;
import org.mybatis.spring.SqlSessionFactoryBean;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;

@Configuration
@PropertySource("classpath:/application.properties")
public class MysqlConfig {
    @Bean
    @ConfigurationProperties(prefix="spring.datasource.hikari")
    public HikariConfig hikariConfig() {
        return new HikariConfig();
    }

    @Bean
    public DataSource dataSource() throws Exception {
        DataSource dataSource = new HikariDataSource(hikariConfig());
        System.out.println("dataSource:" + dataSource.toString());
        return dataSource;
    }

    @Bean
    public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {
        SqlSessionFactoryBean sessionFactory=new SqlSessionFactoryBean();
        sessionFactory.setDataSource(dataSource);
        Resource[] resources=new PathMatchingResourcePatternResolver().getResources("classpath:/mapper/*Mapper.xml");
        sessionFactory.setMapperLocations(resources);
        return sessionFactory.getObject();
    }

    @Bean
    public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactioy) throws Exception {
        return new SqlSessionTemplate(sqlSessionFactioy);
    }
}
```

3) Mapper 파일(SQL문 관리)들을 저장할 'mapper' 폴더를 생성하고 MysqlMapper.xml 파일을 작성한다.

[src/main/resources]-[mapper] MysqlMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.mapper.MysqlMapper">
    <select id="now" resultType="string">
        select now();
    </select>
</mapper>
```

4) DAO 인터페이스와 DAO implements 클래스를 통해 데이터베이스에 접속한다.

[src/main/java]-[com.example.dao] MysqlDAO.java

```
public interface MysqlDAO {
    public String now();
}
```

[src/main/java]-[com.example.dao] MysqlDAOImpl.java

```
@Repository
public class MysqlDAOImpl implements MysqlDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.MysqlMapper";

    @Override
    public String now() {
        return session.selectOne(namespace + ".now");
    }
}
```

5) 작성한 DAO 클래스를 테스트한다.

[src/test/java]-[com.example] ApplicationTests.java

```
@SpringBootTest
class Ex02ApplicationTests {
    @Autowired
    MysqlDAO dao;

    @Test
    void now() {
        System.out.println("현재날짜시간:" + dao.now());
    }
}
```

- **SQL Log설정 (Log4dbc를 사용한 쿼리 로그)**

MyBatis는 JDBC의 PreparedStatement를 이용해서 SQL을 처리하므로 파라미터(?)를 확인하려면 'log4jdbc-log4j2'를 사용한다.

1) log4jdbc를 dependency에 추가한다.

pom.xml

```
<dependency>
    <groupId>org.bgee.log4jdbc-log4j2</groupId>
    <artifactId>log4jdbc-log4j2-jdbc4.1</artifactId>
    <version>1.16</version>
</dependency>
```

2) application.properties의 hikari 설정을 아래와 같이 수정한다.

[src/main/resources] application.properties

```
...
spring.datasource.hikari.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy
spring.datasource.hikari.jdbc-url=jdbc:log4jdbc:mysql://localhost/webdb?serverTimezone=UTC
```

3) log4jdbc.log4j2.properties 생성

[src/main/resources] log4jdbc.log4j2.properties

```
log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
log4jdbc.dump.sql.maxlinelength=0
log4jdbc.auto.load.popular.drivers=false
log4jdbc.drivers=com.mysql.cj.jdbc.Driver
```

- **사용자 등록 프로그램 작성**

1) users 테이블을 생성하고 샘플 데이터를 입력한다.

users 테이블 생성

```
create table users(
    uid varchar(10) not null primary key,
    upass varchar(200) not null,
    uname varchar(20) not null,
    address varchar(500),
    phone varchar(20),
    regdate datetime default now()
);
```

users 테이블에 Sample Data 입력

```
insert into users(uid, upass, uname) values('blue', 'pass', '홍길동');
insert into users(uid, upass, uname) values('red', 'pass', '이순신');
insert into users(uid, upass, uname) values('white', 'pass', '강감찬');
insert into users(uid, upass, uname) values('green', 'pass', '성춘향');
```

2) [com.example.domain] 패키지를 생성한 후 UserVO 클래스를 정의한다.

[src/main/java]-[com.example.domain] UserVO.java

```
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;

public class UserVO {
    private String uid;
    private String upass;
    private String uname;
    private String phone;
    private String address;
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss", timezone="Asia/Seoul")
    private Date regdate;
    //...setter()/getter()/toString()
}
```

3) UserMapper, UserDAO 인터페이스, UserDAOImpl 클래스 파일을 작성한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.mapper.UserMapper">
    <insert id="insert">
        insert into users(uid, upass, uname, phone, address)
        values(#{uid}, #{upass}, #{uname}, #{phone}, #{address})
    </insert>
</mapper>
```

[src/main/java]-[com.example.dao] UserDAO.java

```
public interface UserDAO {
    public void insert(UserVO vo);
}
```

[src/main/java]-[com.example.dao] UserDAOImpl.java

```
@Repository
public class UserDAOImpl implements UserDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.UserMapper";

    @Override
    public void insert(UserVO vo) {
        session.insert(namespace + ".insert", vo);
    }
}
```

4) UserDAO 인터페이스의 insert 메서드를 테스트한다.

[src/test/java]-[com.example] UserDAOTest.java

```
@SpringBootTest
public class UserDAOTest {
    @Autowired
    UserDAO dao;

    @Test
    void insert() {
        UserVO vo=new UserVO();
        vo.setUid("black");
        vo.setUpass("pass");
        vo.setUname("이블랙");
        vo.setPhone("010-1010-2020");
        vo.setAddress("인천 남구 학익동");
        dao.insert(vo);
        vo=new UserVO();
        vo.setUid("pink");
        vo.setUpass("pass");
        vo.setUname("박핑크");
        vo.setPhone("010-1010-3030");
        vo.setAddress("서울 강남구 압구정동");
        dao.insert(vo);
    }
}
```

5) UserController에 입력(insert) 매핑을 추가하고 insert.jsp 페이지를 작성한다.

[src/main/java]-[com.example.controller] UserController.java

```
package com.example.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/user")
public class UserController {
    @RequestMapping("/insert")
    public String insert(Model model) {
        model.addAttribute("pageName", "user/insert");
        return "home";
    }
}
```

[src/main/resources]-[templates]-[user] insert.jsp

```
<div class="row">
    <div class="col">
        <h1 class="text-center mb-5">사용자등록</h1>
        <form class="card p-5" method="post">
            <input name="uid"
                class="form-control my-2" placeholder="사용자 아이디">
            <input name="upass" type="password"
                class="form-control my-2" placeholder="사용자 비밀번호">
            <input name="uname"
                class="form-control my-2" placeholder="사용자 이름">
            <input name="phone"
                class="form-control my-2" placeholder="사용자 전화번호">
            <input name="address"
                class="form-control my-2" placeholder="사용자 주소">
            <div class="text-center mt-3">
                <button class="btn btn-primary px-5 me-3">등록</button>
                <button class="btn btn-secondary px-5" type="reset">취소</button>
            </div>
        </form>
    </div>
</div>
```

6) 사용자 등록 페이지를 메뉴에 등록한다.

[src/main/resources]-[templates] header.html

```
<div class="row mb-5 justify-content-center">
    <div class="col">
        <div class="text-center"><img src="https://via.placeholder.com/960x150" width="100%"></div>
        <div class="my-3">
            <span class="pe-3"><a href="/">Home</a></span>
            <span class="pe-3"><a href="/user/insert">사용자등록</a></span>
        </div>
    </div>
</div>
```

7) UserController에 사용자 등록 POST 매핑을 추가한다.

[src/main/java]-[com.example.controller] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    UserDAO dao;

    @RequestMapping("/insert")
    public String insert(Model model) {
        model.addAttribute("pageName", "user/insert");
        return "home";
    }

    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public String insert(UserVO vo) {
        dao.insert(vo);
        return "redirect:/";
    }
}
```

- **사용자 목록 프로그램 작성**

1) UserMapper xml 파일에 목록 출력 mapper를 추가한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.mapper.MemberMapper">
    <insert id="insert">
        insert into users(uid, upass, uname, phone, address)
        values(#{uid}, #{upass}, #{uname}, #{phone}, #{address})
    </insert>
    <select id="list" resultType="com.example.domain.UserVO">
        select * from users
    </select>
</mapper>
```

2) UserMapper 인터페이스에 list 메서드를 추가하고 테스트한다.

[src/main/java]-[com.example.dao] UserDAO.java

```
public interface UserDAO {
    public void insert(UserVO vo);
    public List<UserVO> list();
}
```

[src/main/java]-[com.example.dao] UserDAOImpl.java

```
@Repository
public class UserDAOImpl implements UserDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.UserMapper";

    @Override
    public void insert(UserVO vo) {
        session.insert(namespace + ".insert", vo);
    }

    @Override
    public List<UserVO> list() {
        return session.selectList(namespace + ".list");
    }
}
```

[src/main/test]-[com.example] UserDAOTest.java

```
@SpringBootTest
public class UserDAOTest {
    @Autowired
    UserDAO dao;
    ...
    @Test
    void list() {
        List<UserVO> users=dao.list();
        for(UserVO vo:users) {
            System.out.println(vo.toString());
        }
    }
}
```

3) UserController에 목록(list) 매핑을 추가하고 list.jsp 페이지를 작성한다.

[src/main/java]-[com.example.controller] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    UserDAO dao;

    @RequestMapping("/insert")
    public String insert(Model model) {
        model.addAttribute("pageName", "user/insert");
        return "home";
    }

    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public String insert(UserVO vo) {
        dao.insert(vo);
        return "redirect:/";
    }

    @RequestMapping("/list")
    public String list(Model model) {
        model.addAttribute("users", dao.list());
        model.addAttribute("pageName", "/user/list");
        return "home";
    }
}
```

[src/main/resoruces]-[templates]-[user] list.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center">사용자목록</h1>
        <table class="table table-striped mt-5">
            <tr>
                <td>아이디</td>
                <td>이름</td>
                <td>전화번호</td>
                <td>주소</td>
            </tr>
            <tr th:each="user:${users}">
                <td th:text="${user.uid}"></td>
                <td th:text="${user.uname}"></td>
                <td th:text="${user.phone}"></td>
                <td th:text="${user.address}"></td>
            </tr>
        </table>
    </div>
</div>
```

4) 사용자 목록 페이지를 메뉴에 등록한다.

[src/main/resoruces]-[templates] header.html

```
<div class="row mb-5 justify-content-center">
    <div class="col">
        <div class="text-center"><img src="https://via.placeholder.com/960x150" width="100%"></div>
        <div class="my-3">
            <span class="pe-3"><a href="/">Home</a></span>
            <span class="pe-3"><a href="/user/insert">사용자등록</a></span>
            <span class="pe-3"><a href="/user/list">사용자목록</a></span>
        </div>
    </div>
</div>
```

- **사용자 정보 읽기 프로그램 작성**

1) UserMapper xml 파일에 사용자 정보 읽기 mapper를 추가한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<mapper namespace="com.example.mapper.UserMapper">
    <insert id="insert">
        insert into users(uid, upass, uname, phone, address)
        values(#{uid}, #{upass}, #{uname}, #{phone}, #{address})
    </insert>
    <select id="list" resultType="com.example.domain.UserVO">
        select * from users
    </select>
    <select id="read" resultType="com.example.domain.UserVO">
        select * from users where uid=#{uid}
    </select>
</mapper>
```

2) UserMapper 인터페이스에 read 메서드를 추가하고 테스한다.

[src/main/java]-[com.example.dao] UserDAO.java

```
public interface UserDAO {
    public void insert(UserVO vo);
    public List<UserVO> list();
    public UserVO read(String uid);
}
```

[src/main/java]-[com.example.dao] UserDAOImpl.java

```
@Repository
public class UserDAOImpl implements UserDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.UserMapper";

    @Override
    public void insert(UserVO vo) {
        session.insert(namespace + ".insert", vo);
    }

    @Override
    public List<UserVO> list() {
        return session.selectList(namespace + ".list");
    }

    @Override
    public UserVO read(String uid) {
        return session.selectOne(namespace + ".read", uid);
    }
}
```

[src/main/test]-[com.example] UserDAOTest.java

```
@SpringBootTest
public class UserDAOTest {
    @Autowired
    UserDAO dao;
    ...
    @Test
    void read() {
        System.out.println(dao.read("manager"));
    }
}
```

3) UserController에 목록(list) 매핑을 추가하고 read.jsp 페이지를 작성한다.

[src/main/java]-[com.example.controller] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    UserDAO dao;
    ...
    @RequestMapping("/read/{uid}")
    public String read(@PathVariable("uid") String id, Model model) {
        model.addAttribute("user", dao.read(id));
        model.addAttribute("pageName", "/user/read");
        return "home";
    }
}
```

[src/main/resources]-[templates]-[user] read.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center mb-5">사용자정보</h1>
        <div class="card p-5">
            <div class="mb-3">사용자 아이디: <span th:text="${user.uid}"></span></div>
            <div class="mb-3">사용자 성명: <span th:text="${user.uname}"></span></div>
            <div class="mb-3">사용자 전화: <span th:text="${user.phone}"></span></div>
            <div class="mb-3">사용자 주소: <span th:text="${user.address}"></span></div>
        </div>
        <div class="text-center mt-3">
            <button class="btn btn-primary px-5">정보수정</button>
        </div>
    </div>
</div>
```

4) 사용자 정보 페이지를 목록 페이지에 등록한다.

[src/main/resources]-[templates]-[user] list.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center">사용자목록</h1>
        <table class="table table-striped mt-5">
            <tr>
                <td>아이디</td>
                <td>이름</td>
                <td>전화번호</td>
                <td>주소</td>
            </tr>
            <tr th:each="user:${users}">
                <td><a th:text="${user.uid}" th:href="|/user/read/${user.uid}|"></a></td>
                <td th:text="${user.uname}"></td>
                <td th:text="${user.phone}"></td>
                <td th:text="${user.address}"></td>
            </tr>
        </table>
    </div>
</div>
```

- **사용자 정보 수정 프로그램 작성**

1) UserMapper xml 파일에 사용자 수정 mapper를 추가한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<update id="update">
    update users
    set uname=#{uname}, phone=#{phone}, address=#{address}
    where uid=#{uid}
</update>
```

2) UserMapper 인터페이스에 update 메서드를 추가하고 테스트한다.

[src/main/java]-[com.example.dao] UserDAO.java

```
public interface UserDAO {
    ...
    public void update(UserVO vo);
}
```

[src/main/java]-[com.example.dao] UserDAOImpl.java

```
public class UserDAOImpl implements UserDAO {
    ..
    @Override
    public void update(UserVO vo) {
        session.update(namespace + ".update", vo);
    }
}
```

[src/main/test]-[com.example] UserDAOTest.java

```
@SpringBootTest
public class UserDAOTest {
    ...
    @Test
    void update() {
        UserVO vo=new UserVO();
        vo.setUid("admin");
        vo.setUname("슈퍼 관리자");
        vo.setPhone("010-1010-2020");
        vo.setAddress("인천 남구 학익동 인천일보아카데미");
        dao.update(vo);
    }
}
```

3) UserController에 수정(update) 매핑을 추가하고 update.jsp 페이지를 작성한다.

[src/main/java]-[com.example.controller] UserController.java

```
public class UserController {
    ...
    @RequestMapping("/update/{uid}")
    public String update(@PathVariable("uid") String id, Model model) {
        model.addAttribute("user", dao.read(id));
        model.addAttribute("pageName", "/user/update");
        return "home";
    }

    @RequestMapping(value="/update/{uid}", method=RequestMethod.POST)
    public String update(@PathVariable("uid") String id, UserVO vo) {
        dao.update(vo);
        return "redirect:/user/read/" + id;
    }
}
```

- jquery를 사용을 위해 home 페이지에 라이브러리를 추가한다.

[scr/main/resources]-[themplates] home.html

```
<head>
    <meta charset="UTF-8">
    <title>홈페이지</title>
    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
    ...
</head>
```

[scr/main/resources]-[themplates]-[user] update.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center mb-5">정보수정</h1>
        <form class="card p-5" method="post" name="frm">
            <input name="uid" class="form-control my-2" th:value="${user.uid}" readonly>
            <input name="uname" class="form-control my-2" th:value="${user.uname}">
            <input name="phone" class="form-control my-2" th:value="${user.phone}">
            <input name="address" class="form-control my-2" th:value="${user.address}">
            <div class="text-center mt-3">
                <button class="btn btn-primary px-5 me-3" type="submit">수정</button>
                <button class="btn btn-secondary px-5" type="reset">취소</button>
            </div>
        </form>
    </div>
</div>
<script>
    $(frm).on("submit", function(e) {
        e.preventDefault();
        let uname=$(frm.uname).val();
        let phone=$(frm.phone).val();
        let address=$(frm.address).val();
        if(uname=="" || phone=="" || address=="" ) {
            alert("이름과 전화번호 주소를 입력하세요!");
        }else {
            if(confirm("사용자 정보를 수정하실래요?")){
                frm.submit();
            }
        }
    });
</script>
```

4) 사용자 수정 페이지를 사용자 정보 페이지에 등록한다.

[scr/main/resources]-[themplates]-[user] read.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center mb-5">사용자정보</h1>
        ...
        <div class="mb-3">사용자 주소: <span th:text="${user.address}"></span></div>
        <div class="text-center mt-3">
            <a th:href="|/user/update/${user.uid}|">
                <button class="btn btn-primary px-5">정보수정</button>
            </a>
        </div>
    </div>
</div>
```

- **사용자 등록에서 이미지 업로드**

1) 사용자 이미지 파일명을 저장하기 위해 photo 필드를 tbl_user 테이블에 추가한 후 UserVO에 photo 필드를 추가한다.

tbl_user 테이블 수정

```
alter table users add column photo varchar(200);
```

[src/main/java]-[com.example.domain] UserVO.java

```
public class UserVO {
    private String uid;
    private String upass;
    private String uname;
    private String phone;
    private String address;
    private Date regdate;
    private String photo;
    //...setter()/getter()/toString() 추가
}
```

2) 파일 업로드에 필요한 라이브러리 추가와 설정 작업을 한다.

pom.xml

```
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.1</version>
</dependency>
```

[src/main/resources/] application.properties

```
server.port = 8088
spring.datasource.hikari.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.jdbc-url=jdbc:mysql://localhost/webdb?serverTimezone=UTC
spring.datasource.hikari.username=web
spring.datasource.hikari.password=pass
spring.servlet.multipart.maxFileSize=10485760
spring.servlet.multipart.maxRequestSize=10485760
```

3) UserMapper 인터페이스 insert 메서드를 아래와 같이 수정한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<mapper namespace="com.example.mapper.UserMapper">
    <insert id="insert">
        insert into users(uid, upass, uname, phone, address, photo)
        values(#{uid}, #{upass}, #{uname}, #{phone}, #{address}, #{photo})
    </insert>
    <select id="list" resultType="com.example.domain.UserVO">
        select * from users
    </select>
    <select id="read" resultType="com.example.domain.UserVO">
        select * from users where uid=#{uid}
    </select>
    <update id="update">
        update users
        set uname=#{uname}, phone=#{phone}, address=#{address}
        where uid=#{uid}
    </update>
</mapper>
```

4) 사용자 입력 페이지를 아래와 같이 수정한다.

[src/main/resources]-[templates]-[user] insert.html

```
<div class="row">
    <div class="col">
        <h1 class="text-center mb-5">사용자등록</h1>
        <form class="card p-5" method="post" name="frm" enctype="multipart/form-data">
            <input name="uid"
                class="form-control my-2" placeholder="사용자 아이디">
            <input name="upass" type="password"
                class="form-control my-2" placeholder="사용자 비밀번호">
            <input name="uname" class="form-control my-2" placeholder="사용자 이름">
            <input name="phone" class="form-control my-2" placeholder="사용자 전화번호">
            <input name="address" class="form-control my-2" placeholder="사용자 주소">
            <div>
                <img id="photo" src="https://via.placeholder.com/200x250" width="200">
                <input name="file" type="file" class="form-control mt-3">
            </div>
            <div class="text-center mt-3">
                <button class="btn btn-primary px-5 me-3">등록</button>
                    <button class="btn btn-secondary px-5" type="reset">취소</button>
            </div>
        </form>
    </div>
</div>
<script>
    //이미지 미리보기
    $(frm.file).on("change", function(e) {
        $("#photo").attr("src", URL.createObjectURL(e.target.files[0]));
    });
</script>
```

5) UserController의 등록(insert) 매핑을 아래와 같이 수정한다.

[src/main/java]-[com.example.controller] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    UserDAO dao;
    ...
    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public String insert(UserVO vo, MultipartHttpServletRequest multi)throws Exception {
        MultipartFile file=multi.getFile("file");
        //업로드 파일이 존재하는 경우
        if(!file.isEmpty()) {
            //업로드 폴더가가 없으면 폴더생성
            String filePath="/upload/photo/";
            File forder = new File("c:" + filePath);
            if(!forder.exists()) forder.mkdir();
            //파일업로드
            String fileName = System.currentTimeMillis() + ".jpg";
            file.transferTo(new File("c:" + filePath + fileName));
            vo.setPhoto(filePath + fileName);
        }
        dao.insert(vo);
        return "redirect:/";
    }
}
```

- **사용자 이미지 출력하기**

1) 이미지 출력(display) 매핑을 아래와 같이 작성한다.

[src/main/java]-[com.example.controller] HomeController.java

```
package com.example.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.apache.commons.io.IOUtils;
import java.io.*;

@Controller
public class HomeController {
    @ResponseBody
    @RequestMapping("/hello")
    public String hello() {
        return "Hello World!";
    }

    @RequestMapping("/")
    public String home(Model model) {
        model.addAttribute("pageName", "about");
        return "home";
    }

    @RequestMapping("/display")
    public @ResponseBody byte[] display(String fileName)throws Exception {
        InputStream in = new FileInputStream("c:" + fileName);
        return IOUtils.toByteArray(in);
    }
}
```

2) 사용자 정보 출력 페이지와 수정 페이지를 아래와 같이 수정한다.

[src/main/resources]-[templates]-[user] read.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center mb-5">사용자정보</h1>
        <div class="card p-5">
            <div class="mb-3">사용자 아이디: <span th:text="${user.uid}"></span></div>
            <div class="mb-3">사용자 성명: <span th:text="${user.uname}"></span></div>
            <div class="mb-3">사용자 전화: <span th:text="${user.phone}"></span></div>
            <div class="mb-3">사용자 주소: <span th:text="${user.address}"></span></div>
            <div th:if="${user.photo!=null && user.photo!=''}">
                <img th:src="|/display?fileName=${user.photo}|" width=200>
            </div>
            <div th:unless="${user.photo!=null && user.photo!=''}">
                <img src="https://via.placeholder.com/200x250" width=200>
            </div>
        </div>
        <div class="text-center mt-3">
            <a th:href="|/user/update?id=${user.uid}|">
                <button class="btn btn-primary px-5">정보수정</button>
            </a>
        </div>
    </div>
</div>
```

- **사용자 정보와 이미지 수정하기**

1) UserMapper의 수정sql문을 아래와 같이 수정한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<mapper namespace="com.example.mapper.UserMapper">
    <update id="update">
        update users
        set uname=#{uname}, phone=#{phone}, address=#{address}, photo=#{photo}
        where uid=#{uid}
    </update>
</mapper>
```

2) 수정 페이지에 이미지 출력 부분을 아래와 같이 수정한다.

[src/main/resources]-[templates]-[user] update.html

```
<form class="card p-5" method="post" name="frm" enctype="multipart/form-data">
    ...
    <input name="photo" th:value="${user.photo}" style="display:none;">
    <div th:if="${user.photo!=null && user.photo!=''}">
        <img th:src="|/display?fileName=${user.photo}|" width=200 id="photo">
    </div>
    <div th:unless="${user.photo!=null && user.photo!=''}">
        <img src="https://via.placeholder.com/200x250" width=200 id="photo">
    </div>
    <input name="file" type="file" class="form-control mt-2">
    <div class="text-center mt-3">
        <button class="btn btn-primary px-5 me-3">수정</button>
        <button class="btn btn-secondary px-5" type="reset">취소</button>
    </div>
</form>
<script>
    ...
    $(frm.file).on("change", function(e) {
        $("#photo").attr("src", URL.createObjectURL(e.target.files[0]));
    });
</script>
```

3) UserController의 수정(update) 매핑을 아래와 같이 수정한다.

[src/main/java]-[com.example.controller] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    ...
    @RequestMapping(value="/update/{uid}", method=RequestMethod.POST)
    public String update(@PathVariable("uid") String id, UserVO vo, MultipartHttpServletRequest multi)throws Exception {
        MultipartFile file=multi.getFile("file");
        if(!file.isEmpty()) {
            String filePath="/upload/photo/";
            String fileName=System.currentTimeMillis() + ".jpg";
            file.transferTo(new File("c:" + filePath + fileName));
            vo.setPhoto(filePath + fileName);
        }
        dao.update(vo);
        return "redirect:/user/list";
    }
}
```

- **사용자 정보 읽기횟수 누적 프로그램 작성 (트랜잭션 처리)**

트랜잭션은 여러 작업이 연속해서 발생할 경우 한 작업에서 오류가 발생하면 모든 작업 취소를 위해 트랜잭션 롤백 처리를 해주어야 한다.

1) 읽기횟수를 누적하기 위해 viewcnt 필드를 tbl_user 테이블에 추가한 후 UserVO에 viewcnt 필드를 추가한다.

tbl_user 테이블 수정

```
alter table users add column viewcnt int default 0
```

[src/main/java]-[com.example.domain] UserVO.java

```
private int viewcnt;
```

2) 트랜잭션 처리를 위해 DatabaseConfig 파일에 아래 내용을 추가한다.

[src/main/java]-[com.example] MysqlConfig.java

```
import org.springframework.jdbc.datasource.DataSourceTransactionManager;
...
public class MysqlConfig {
    ..
    @Bean
    public DataSourceTransactionManager txManager(DataSource dataSource) {
        return new DataSourceTransactionManager(dataSource);
    }
}
```

3) UserDAO 인터페이스와 UserMapper.xml 파일에 아래 메서드를 추가한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<mapper namespace="com.example.mapper.UserMapper">
    ...
    <update id="update_viewcnt">
        update users set viewcnt=viewcnt+1
        where uid=#{uid}
    </update>
</mapper>
```

[scr/main/java]-[com.example.dao] UserDAO.java

```
public interface UserDAO {
    public void insert(UserVO vo);
    public List<UserVO> list();
    public UserVO read(String uid);
    public void update(UserVO vo);
    public void updateViewcnt(String uid);
}
```

[scr/main/java]-[com.example.dao] UserDAOImpl.java

```
@Repository
public class UserDAOImpl implements UserDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.UserMapper";
    ...
    @Override
    public void updateViewcnt(String uid) {
        session.update(namespace + ".update_viewcnt", uid);
    }
}
```

4) [com.example.service] 패키지 생성 후 UserService 인터페이스와 UserServiceImpl 클래스를 생성한다.

[src/main/java]-[com.example.service] UserService.java

```
public interface UserService {
    public UserVO read(String uid);
}
```

[src/main/java]-[com.example.service] UserServiceImpl.java

```
@Service
public class UserServiceImpl implements UserService{
    @Autowired
    UserDAO dao;

    @Transactional
    @Override
    public UserVO read(String uid) {
        dao.updateViewcnt(uid);
        return dao.read(uid);
    }
}
```

5) 목록페이지에 사용자 조회 출력횟수를 추가한다.

[com.example.resources]-[templates]-[user] list.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center">사용자목록</h1>
        <table class="table table-striped mt-5">
            <tr th:each="user:${users}">
                <td><a th:text="${user.uid}" th:href="|/user/read?id=${user.uid}|"></a></td>
                <td th:text="${user.uname}"></td>
                <td th:text="${user.phone}"></td>
                <td th:text="${user.address}"></td>
                <td th:text="${user.viewcnt}"></td>
            </tr>
        </table>
    </div>
</div>
```

6) UserController에서 읽기(read) 매핑을 아래와 같이 수정한다.

[src/main/java]-[com.example.controller] UserController.java

```
public class UserController {
    @Autowired
    UserDAO dao;

    @Autowired
    UserService service;

    @RequestMapping("/read")
    public String read(String id, Model model) {
        //model.addAttribute("user", dao.read(id));
        model.addAttribute("user", service.read(id));
        model.addAttribute("pageName", "/user/read");
        return "home";
    }
}
```

- **로그인/로그아웃 처리**

1) UserController에 로그인(login)/로그아웃(logout) 매핑을 추가한다.

[src/main/java]-[com.example.controller] UserController.java

```
@Controller
@RequestMapping("/user")
public class UserController {
    ...
    @RequestMapping("/login")
    public String login(Model model) {
        model.addAttribute("pageName", "/user/login");
        return "home";
    }

    @ResponseBody
    @RequestMapping(value="/login", method=RequestMethod.POST)
    public int login(UserVO vo) {
        int result=0;
        UserVO user=dao.read(vo.getUid());
        if(user != null) {
            if(user.getUpass().equals(vo.getUpass())) {
                result=1;
            }else {
                result=2;
            }
        }
        return result;
    }
}
```

2) 헤더 페이지에 아래 내용을 추가한다.

[src/main/resources]-[templates] header.html

```
<div class="row mb-5 justify-content-center" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <div class="text-center"><img src="https://via.placeholder.com/960x150" width="100%"></div>
        <div class="my-3">
            ...
            <span class="pe-3" style="float:right;" id="login"><a href="/user/login">로그인</a></span>
            <span class="pe-3" style="float:right;display:none" id="logout"><a href="#">로그아웃</a></span>
            <span class="pe-3" style="float:right;display:none" id="uid"><a href="#">아이디</a></span>
        </div>
    </div>
</div>
<script>
    if(sessionStorage.getItem("uid")) {
        $("#login").toggle();
        $("#logout").toggle();
        $("#uid").toggle();
        $("#uid a").html(sessionStorage.getItem("uid"));
    }

    $("#logout a").on("click", function(e) {
        e.preventDefault();
        sessionStorage.clear();
        location.href="/";
    });

    $("#uid").on("click", function(e) {
        e.preventDefault();
        location.href="/user/read/" + sessionStorage.getItem("uid");
    });
</script>
```

3) 로그인 페이지를 작성하고 header 페이지를 수정한다.

[src/main/resources]-[templates]-[user] login.html

```
<div class="row justify-content-center">
    <div class="col-md-6 my-5">
        <h1 class="text-center">로그인</h1>
        <form name="frm" class="card p-3" method="post">
            <input name="uid" class="form-control my-2">
            <input name="upass" type="password" class="form-control my-2">
            <div class="mt-2"><input name="save" type="checkbox" class="me-2">로그인상태유지</div>
            <div class="my-2"><button class="btn btn-primary w-100">로그인</button></div>
        </form>
    </div>
</div>
<script>
    $(frm).on("submit", function(e) {
        e.preventDefault();
        let uid=$(frm.uid).val();
        let upass=$(frm.upass).val();
        $.ajax({
            type: "post",
            url: "/user/login",
            data: { uid:uid, upass:upass },
            success: function(data){
                if(data==0) {
                    alert("아이디가 존재하지 않습니다!");
                }else if(data==2) {
                    alert("비밀번호가 일치하지 않습니다!");
                }else if(data==1) {
                    sessionStorage.setItem("uid", uid);
                    if($(frm.save).is(":checked")) {  //쿠키저장(개발자도구-애플리케이션-쿠키에서확인)
                        let date=new Date();
                        date.setTime(date.getTime() + 1*24*60*60*1000)
                        document.cookie=`uid=${uid};path=/;expires=${date.toUTCString()}`;
                    }
                    location.href="/";
                }
           }
        });
    });
</script>
```

[src/main/resources]-[templates] header.html

```
<script>
    //쿠키읽기
    let cookieName="uid"
    var value = document.cookie.match('(^|;) ?' + cookieName + '=([^;]*)(;|$)');
    if (value != null) sessionStorage.setItem("uid", value[2]);  //쿠키등록

    if(sessionStorage.getItem("uid")) {
        $("#login").toggle();
        $("#logout").toggle();
        $("#uid").toggle();
        $("#uid a").html(sessionStorage.getItem("uid"));
    }

    $("#logout a").on("click", function(e) {
        e.preventDefault();
        sessionStorage.clear();  //세션삭제
        document.cookie = "uid='';path=/;expires=Thu, 01 Jan 1999 00:00:10 GMT;"  //쿠키삭제
        location.href="/"
    });

    $("#uid").on("click", function(e) {
        e.preventDefault();
        location.href="/user/read/" + sessionStorage.getItem("uid");
    });

</script>
```

- **REST API를 이용한 게시글 목록 출력하기**

1) 게시글 테이블과 VO를 정의한다.

posts 테이블 생성

```
create table posts(
    id int auto_increment primary key,
    title varchar(500) not null,
    body text,
    regdate datetime default now(),
    writer varchar(20),
    foreign key(writer) references users(uid)
);
```

posts 테이블에 샘플 데이터 입력

```
insert into posts(title, body, writer)
values('스프링 정의','엔터프라이즈용 Java 애플리케이션 개발을 편하게 할 수 있게 해주는 오픈소스 경량급 애플리케이션 프레임워크이다','red');

insert into posts(title, body, writer)
values('애플리케이션 프레임워크','애플리케이션 프레임워크가 무엇인지 이해하기 위해서는 프레임워크에 대해서 알아야 합니다.','red');

insert into posts(title, body, writer)
values('POJO 프로그래밍을 지향','스프링의 가장 큰 특징은 POJO 프로그래밍을 지향한다는 것입니다.','blue');

insert into posts(title, body, writer)
values('스프링(Spring)이란?','Spring은 무엇일까요? 스프링은 자바 기반의 웹 어플리케이션을 만들 수 있는 프레임워크입니다.','blue');

insert into posts(title, body, writer)
values('스프링의 특징','Spring은 자바 객체와 라이브러리들을 관리해주며, 톰캣과 같은 WAS 가 내장되어 있어 자바 웹 어플리케이션을 구동할 수 있습니다.','blue');
```

[src/main/java]-[com.example.domain] PostVO.java

```
public class PostVO extends UserVO {
    private int id;
    private String title;
    private String body;
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss", timezone="Asia/Seoul")
    private Date regdate;
    private String writer;
    //getter()/setter()/toString()함수 추가
}
```

2) 게시글 Mapper와 DAO 인터페이스를 정의한다.

[src/main/resources]-[mapper] PostMapper.xml

```
<mapper namespace="com.example.mapper.PostMapper">
    <select id="list" resultType="com.example.domain.PostVO">
        select * from posts
        order by id desc
        limit #{start}, #{size}
    </select>
</mapper>
```

[src/main/java]-[com.example.dao] PostDAO.java

```
public interface PostDAO {
    public List<PostVO> list(int page, int size);
}
```

[src/main/java]-[com.example.dao] PostDAOImpl.java

```
@Repository
public class PostDAOImpl implements PostDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.PostMapper";

    @Override
    public List<PostVO> list(int page, int size) {
        Map<String, Object> map=new HashMap<String, Object>();
        map.put("start", (page-1) * size);
        map.put("size", size);
        return session.selectList(namespace + ".list", map);
    }
}
```

3) 게시글 출력 Controller와 페이지를 작성한다. 테스트를 위해 [chrome 웹 스토어]-[검색] 'JSON Viewer' 설치한다.

[src/main/java]-[com.example.controller] PostRESTController.java

```
@RestController
@RequestMapping("/api/post")
public class PostRESTController {
    @Autowired
    PostDAO dao;

    @RequestMapping("/list")
    public List<PostVO> list(int page, int size) {
        return dao.list(page, size);
    }
}
```

[src/main/java]-[com.example.controller] PostController.java

```
@Controller
@RequestMapping("/post")
public class PostController {

    @RequestMapping("/list")
    public String list(Model model) {
        model.addAttribute("pageName", "/post/list");
        return "home";
    }
}
```

[src/main/resources]-[templates] heaer.html

```
<div class="row mb-5 justify-content-center">
    <div class="col">
        <div class="text-center"><img src="https://via.placeholder.com/960x150" width="100%"></div>
        <div class="my-3">
            <span class="pe-3"><a href="/">Home</a></span>
            <span class="pe-3"><a href="/user/list">사용자목록</a></span>
            <span class="pe-3"><a href="/post/list">게시글</a></span>
            <span class="pe-3" style="float:right;" id="login"><a href="/user/login">로그인</a></span>
            <span class="pe-3" style="float:right;display:none" id="logout"><a href="#">로그아웃</a></span>
            <span class="pe-3" style="float:right;display:none" id="uid"><a href="#">아이디</a></span>
        </div>
    </div>
</div>
```

[src/main/resources]-[templates] home.html

```
<head>
    ...
    <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.1/handlebars.js"></script>
</head>
```

[src/main/resources]-[templates]-[post] list.html

```
<div class="row my-5">
    <div class="col">
        <h1 class="text-center">게시글목록</h1>
        <div id="posts"></div>
    </div>
</div>
<!-- 게시글목록 템플릿 -->
<script id="temp" type="x-handlebars-template">
    <table class="table table-striped mt-5">
        {{#each .}}
        <tr>
            <td>{{id}}</td>
            <td>{{title}}</td>
            <td>{{writer}}</td>
            <td>{{regdate}}</td>
        </tr>
        {{/each}}
    </table>
</script>
<script>
    let page=1;
    let size=3;

    getList();
    function getList() {
        $.ajax({
            type: "get",
            url: "/api/post/list",
            dataType: "json",
            data: { page:page, size:size },
            success: function(data) {
                let temp=Handlebars.compile($("#temp").html());
                $("#posts").html(temp(data));
            }
        });
    }
</script>
```

4) 목록 페이지에 페이징 처리를 추가한다.

[src/main/resources]-[mapper] PostMapper.xml

```
<select id="total" resultType="int">
    select count(*) from posts
</select>
```

[src/main/java]-[com.example.dao] PostDAO.java

```
public interface PostDAO {
    public List<Map<String, Object>> list(int page, int size);
    public int total();
}
```

[src/main/java]-[com.example.dao] PostDAOImpl.java

```
@Repository
public class PostDAOImpl implements PostDAO{
    ...
    @Override
    public int total() {
        return session.selectOne(namespace + ".total");
    }
}
```

[src/main/java]-[com.example.controller] PostRESTController.java

```
@RestController
@RequestMapping("/api/post")
public class PostRESTController {
    ...
    @RequestMapping("/total")
    public int total() {
        return dao.total();
    }
}
```

[src/main/resources]-[templates] home.html

```
<head>
    ...
   <script src="https://cdnjs.cloudflare.com/ajax/libs/twbs-pagination/1.4.2/jquery.twbsPagination.min.js"></script>
</head>
```

[src/main/resources]-[templates]-[post] list.html

```
<div class="row my-5">
    <div class="col">
        <h1 class="text-center">게시글목록</h1>
        <div id="posts" class="mb-5"></div>
        <div id="pagination" class="pagination justify-content-center mt-5"></div>
    </div>
</div>
<script>
    let page=1;
    let size=3;

    function getList() {
        ...
    }

    getTotal();
    function getTotal() {
        $.ajax({
            type:"get",
            url:"/api/post/total",
            success:function(data){
            const totalPages=Math.ceil(data/3);
                $("#pagination").twbsPagination("changeTotalPages", totalPages, page);
            }
        });
    }

    $('#pagination').twbsPagination({
        totalPages:1,   //총 페이지 수
        visiblePages: 5,  // 하단에서 한 번에 보이는 페이지 수
        startPage : 1,  // 시작 시 표시되는 현재페이지
        initiateStartPageClick: false,  // 플러그인이 시작 시 페이지 버튼 클릭여부(default : true)
        first : '<<',  // 페이지네이션 버튼 중 처음으로 돌아가는 버튼에 쓰여 있는 텍스트
        prev : '<',  // 이전 페이지 버튼에 쓰여 있는 텍스트
        next : '>',  // 다음 페이지 버튼에 쓰여 있는 텍스트
        last : '>>',  // 페이지네이션 버튼 중 마지막으로 가는 버튼에 쓰여 있는 텍스트
        onPageClick: function (event, clickPage) {
            page=clickPage;
            getList();
        }
    });
</script>
```

- **REST API 이용한 게시글 등록 페이지**

1) REST API 테스틀 위해 [chrome 웹 스토어]-[검색] 'Yet Another Rest Client' 프로그램을 Chrome에 추가한다.

2) 게시글 등록 Mapper DAO 인터페이스를 작성한다.

[src/main/resources]-[mappre] PostMapper.xml

```
<mapper namespace="com.example.mapper.PostMapper">
    ...
    <insert id="insert">
        insert into posts(title, body, writer)
        values(#{title}, #{body}, #{writer})
    </insert>
</mapper>
```

[src/main/java]-[com.example.dao] PostDAO.java

```
public interface PostDAO {
    public List<Map<String, Object>> list(int page, int size);
    public int total();
    public void insert(PostVO vo);
}
```

[src/main/java]-[com.example.dao] PostDAOImpl.java

```
@Repository
public class PostDAOImpl implements PostDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.PostMapper";
    ...
    @Override
    public void insert(PostVO vo) {
        session.insert(namespace + ".insert", vo);
    }
}
```

3) 게시글 등록 페이지로 이동하기위해 목록 페이지를 수정한다.

[src/main/resources]-[templates]-[post] list.html

```
<div class="row my-5">
    <div class="col">
        <h1 class="text-center">게시글목록</h1>
        <div class="text-end my-1" id="btn-group" style="display:none">
            <a href="/post/insert">
                <button class="btn btn-primary">글쓰기</button>
            </a>
        </div>
        <div id="posts"></div>
        <ul class="pagination justify-content-center" id="pagination"></ul>
    </div>
</div>
<script>
    let page=1;
    let size=3;

    if(sessionStorage.getItem("uid")) $("#btn-group").toggle();
    ...
</script>
```

4) 게시글 등록 Controller와 게시글 등록 페이지를 작성한다.

[src/main/java]-[com.example.controller] PostController.java

```
@Controller
@RequestMapping("/post")
public class PostController {
    ...
    @RequestMapping("/insert")
    public String insert(Model model) {
        model.addAttribute("pageName", "/post/insert");
        return "home";
    }
}
```

[src/main/java]-[com.example.controller] PostRESTController.java

```
@RestController
@RequestMapping("/api/post")
public class PostRESTController {
    ...
    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public void insert(PostVO vo) {  //REST Client프로그램이나 React에서 사용할 경우 insert(@Requestbody PostVO vo)
        dao.insert(vo);
    }
}
```

[src/main/resources]-[templates]-[post] insert.html

```
<div class="row">
    <div class="col">
        <h1 class="text-center">글쓰기</h1>
        <form name="frm">
            <input name="title" class="form-control my-2" placeholder="제목을 입력하세요.">
            <textarea name="body" class="form-control my-2" rows=10 placeholder="내용을 입력하세요."></textarea>
            <div class="text-center">
                <button class="btn btn-primary">글등록</button>
                <button class="btn btn-secondary">등록취소</button>
            </div>
        </form>
    </div>
</div>
<script>
    $(frm).on("submit", function(e){
        e.preventDefault();
        let title=$(frm.title).val();
        let body=$(frm.body).val();
        let writer=sessionStorage.getItem("uid");
        if(title=="" || body=="") {
            alert("글제목이나 내용을 입력하세요!");
        }else {
            $.ajax({
                type: "post",
                url: "/api/post/insert",
                data: { title: title, body: body, writer: writer },
                success: function() {
                    alert("새로운 글이 등록되었습니다.")
                    location.href="/post/list";
                }
            });
        }
    });
</script>
```

- **REST API 이용한 게시글 정보 페이지**

1) 게시글 정보 Mapper와 DAO 인터페이스를 작성한다.

[src/main/resources]-[mapper] PostMapper.xml

```
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.mapper.PostMapper">
    ...
    <select id="read" resultType="com.example.domain.PostVO">
        select *
        from tbl_posts
        where id=#{id}
    </select>
</mapper>
```

[src/main/java]-[com.example.dao] PostDAO.java

```
public interface PostDAO {
    public List<Map<String, Object>> list(int page, int size);
    public int total();
    public void insert(PostVO vo);
    public PostVO read(int id);
}
```

[src/main/java]-[com.example.dao] PostDAOImpl.java

```
@Repository
public class PostDAOImpl implements PostDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.PostMapper";
    ...
    @Override
    public PostVO read(int id) {
        return session.selectOne(namespace + ".read", id);
    }
}
```

2) 게시글 정보 페이지로 이동하기 위해 목록 페이지를 수정한다.

[src/main/resources]-[templates]-[post] list.html

```
<div class="row my-5">
    ...
</div>

<!-- 게시글목록 템플릿 -->
<script id="temp" type="x-handlebars-template">
    <table class="table table-striped mt-5">
        {{#each .}}
        <tr>
            <td>{{id}}</td>
            <td><a href="/post/read/{{id}}">{{title}}</a></td>
            <td>{{writer}}</td>
            <td>{{fmtdate}}</td>
        </tr>
        {{/each}}
    </table>
</script>
```

3) 게시글 정보 Controller와 사용자 정보 페이지를 작성한다.

[src/main/java]-[com.example.controller] PostController.java

```
public class PostController {
    ...
    @RequestMapping("/read/{id}")
    public String read(@PathVariable("id") int id, Model model) {
        model.addAttribute("id", id);
        model.addAttribute("pageName", "/post/read");
        return "home";
    }
}
```

[src/main/java]-[com.example.controller] PostRESTController.java

```
public class PostRESTController {
    ...
    @RequestMapping("/read/{id}")
    public PostVO read(@PathVariable("id") int id) {
        return dao.read(id);
    }
}
```

[src/main/resources]-[templates]-[post] read.html

```
<div class="row">
    <div class="col">
        <h1 class="text-center mb-5">게시글 정보</h1>
        <div id="post"></div>
        <div class="text-center my-3" id="btn-group" style="display:none;">
            <button class="btn btn-primary">정보수정</button>
        </div>
    </div>
</div>
<!-- 게시글 정보 템플릿 -->
<script id="temp" type="x-handlebars-template">
    <div class="card">
        <div class="card-body">
            [{{id}}] {{title}}
            <hr>
            {{body}}
        </div>
        <div class="card-footer">Posted on {{regdate}} by {{writer}}</div>
    </div>
</script>
<script>
    let id="[[${id}]]";
    $.ajax({
        type: "get",
        url: "/api/post/read/" + id,
        dataType: "json",
        success: function(data){
            let temp=Handlebars.compile($("#temp").html());
            $("#post").html(temp(data));
            if(sessionStorage.getItem("uid")==data.writer) $("#btn-group").toggle();
        }
    });
</script>
```

- **REST API 이용한 게시글 정보 수정 페이지**

1) 게시글 정보수정 Mapper와 정보수정 DAO 인터페이스를 작성한다.

[src/main/resources]-[mapper] PostMapper.xml

```
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.mapper.PostMapper">
    ...
    <update id="update">
       update posts
       set title=#{title}, body=#{body}
       where id=#{id}
    </update>
</mapper>
```

[src/main/java]-[com.example.dao] PostDAO.java

```
public interface PostDAO {
    public List<Map<String, Object>> list(int page, int size);
    public int total();
    public void insert(PostVO vo);
    public Map<String, Object> read(int id);
    public void update(PostVO vo);
}
```

[src/main/java]-[com.example.dao] PostDAOImpl.java

```
@Repository
public class PostDAOImpl implements PostDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.PostMapper";
    ...
    @Override
    public void update(PostVO vo) {
        session.update(namespace + ".update", vo);
    }
}
```

2) 게시글 정보수정 페이지와 연결하기 위해 정보 페이지를 아래와 같이 수정한다.

[src/main/resources]-[templates]-[post] read.html

```
<div class="row">
    <div class="col">
        <h1 class="text-center mb-5">게시글 정보</h1>
            <div id="post"></div>
            <div class="text-center my-3" id="btn-group" style="display:none;">
                <button class="btn btn-primary">정보수정</button>
            </div>
    </div>
</div>
<script>
    let id="[[${id}]]";  #model에 저장한 값을 가져올 경우
    $.ajax({ ... });
    //정보수정 버튼을 클릭한 경우
    $("#btn-group").on("click", ".btn", function(){
        location.href="/post/update/" + id;
    });
</script>
```

3) 게시글 정보수정 Controller와 정보 수정 페이지를 작성한다.

[src/main/java]-[com.example.controller] PostController.java

```
public class PostController {
    @Autowired
    PostDAO dao;
    ...
    @RequestMapping("/update/{id}")
    public String update(@PathVariable("id") int id, Model model) {
        model.addAttribute("post", dao.read(id));
        model.addAttribute("pageName", "/post/update");
        return "home";
    }
}
```

[src/main/resources]-[templates]-[post] update.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center mb-5">게시글 정보수정</h1>
        <form name="frm" method="post">
            <input name="id" th:value="${post.id}" type="hidden">
            <input name="title" th:value="${post.title}" class="form-control my-2">
            <textarea name="body" th:text="${post.body}" rows="10" class="form-control my-2"></textarea>
            <div class="text-center my-3">
                <button class="btn btn-primary">수정</button>
                <button class="btn btn-secondary" type="reset">취소</button>
            </div>
        </form>
    </div>
</div>
<script>
    $(frm).on("submit", function(e) {
        e.preventDefault();
        let id=$(frm.id).val();
        let title=$(frm.title).val();
        let body=$(frm.body).val();
        if(!confirm("변경된 내용으로 수정하실래요?")) return;
        $.ajax({
            type: "post",
            url: "/api/post/update",
            data: { id: id, title: title, body: body },
            success: function() {
                alert("수정이 완료되었습니다.");
                location.href="/post/read/" + id;
            }
        });
    });
</script>
```

[src/main/java]-[com.example.controller] PostRESTController.java

```
public class PostRESTController {
    ...
    @RequestMapping(value="/update", method=RequestMethod.POST)
    public void update(PostVO vo) {
        dao.update(vo);
    }
}
```

- **REST API 이용한 첨부파일 등록**

1) 첨부파일들을 저장할 테이블을 생성한다.

post_attach 파일 생성

```
create table post_attach(
    id int auto_increment primary key,
    postid int not null,
    image varchar(200) not null
);
```

2) 첨부파일 저장 Mapper와 첨부파일 저장 DAO 인터페이스를 생성한다.

[src/main/resources]-[mapper] PostMapper.xml

```
<mapper namespace="com.example.mapper.PostMapper">
    ...
    <insert id="add_attach">
        insert post_attach(postid, image) value(#{postid}, #{image})
    </insert>
    <select id="get_attach" resultType="hashmap">
        select * from post_attach where postid=#{postid}
        order by id desc
    </select>
    <delete id="del_attach">
        delete from post_attach where id=#{id}
    </delete>
</mapper>
```

[src/main/java]-[com.exmaple.dao] PostDAO.java

```
public interface PostDAO {
    ...
    public void addAttach(int postid, String image);
    public List<Map<String, Object>> getAttach(int postid);
    public void delAttach(int id);
}
```

[src/main/java]-[com.exmaple.dao] PostDAOImpl.java

```
@Repository
public class PostDAOImpl implements PostDAO{
    ....
    @Override
    public void addAttach(int postid, String image) {
        Map<String, Object> map=new HashMap<String, Object>();
        map.put("postid", postid);
        map.put("image", image);
        session.insert(namespace + ".add_attach", map);
    }

    @Override
    public List<Map<String, Object>> getAttach(int postid) {
        return session.selectList(namespace + ".get_attach", postid);
    }

    @Override
    public void delAttach(int id) {
        session.delete(namespace + ".del_attach", id);
    }
}
```

3) 첨부 파일 추가를 위한 게시글 수정 Controller를 아래와 같이 수정한다.

[src/main/java]-[com.example.controller] PostRestController.java

```
public class PostRESTController {
    ...
    @RequestMapping(value="/update", method=RequestMethod.POST)
    public void update(PostVO vo, MultipartHttpServletRequest multi) {
        try {
            //데이터 업데이트
            dao.update(vo);
            //업로드 할 폴더생성
            List<MultipartFile> files=multi.getFiles("files");
            if(files.size() > 0) {
                String pathName = "/upload/attach/" + vo.getId() + "/";
                File folder = new File("c:" + pathName);
                if(!folder.exists()) folder.mkdir();
                //첨부파일들 업로드
                for(MultipartFile file:files) {
                    String fileName = System.currentTimeMillis() + ".png";
                    file.transferTo(new File("c:" + pathName + fileName));
                    dao.addAttach(vo.getId(), pathName + fileName);
                }
            }
        }catch(Exception e) {
            System.out.println("게시글 수정오류:" + e.toString());
        }
    }
}
```

4) 게시글 수정 페이지의 첨부이미지 미리보기 기능을 아래와 같이 추가한다.

[src/main/resources]-[templates]-[post] update.html

```
<form name="frm" method="post">
    ...
    <div id="attachFiles" class="card p-3 my-3">첨부할 이미지</div>
    <input name="files" type="file" multiple class="form-control">
    <div class="text-center my-3">
        <button class="btn btn-primary">수정</button>
        <button class="btn btn-secondary" type="reset">취소</button>
    </div>
</form>
<!-- 첨부할 파일목록 템플릿 -->
<script id="temp_attach" type="x-handlebars-template">
    <div class="row">
        {{#each .}}
            <div class="col-6 col-md-2"><img src="{{this}}" width="100%"></div>
        {{/each}}
    </div>
</script>
<script>
    //이미지들 미리보기
    $(frm.files).on("change", function(e) {
        let attachFiles=[];
        for(let i=0; i<e.target.files.length; i++) {
            attachFile=URL.createObjectURL(e.target.files[i]);
            attachFiles.push(attachFile);
        }
        let temp=Handlebars.compile($("#temp_attach").html());
        $("#attachFiles").html(temp(attachFiles));
    });
</script>
```

5) 첨부 이미지를 업로드하기위해 게시글 수정 페이지의 Submit 이벤트를 아래와 같이 수정한다.

[src/main/resources]-[templates]-[post] update.html

```
...
<form name="frm" method="post">
    <input name="id" th:value="${post.id}" type="hidden">
    <input name="title" th:value="${post.title}" class="form-control my-2">
    <textarea name="body" th:text="${post.body}" rows="10" class="form-control my-2"></textarea>
    <div id="attachFiles" class="card p-3 my-3">첨부할 이미지</div>
    <input name="files" type="file" multiple class="form-control">
    <div class="text-center my-3">
        <button class="btn btn-primary">수정</button>
        <button class="btn btn-secondary" type="reset">취소</button>
    </div>
</form>
...
<script>
    //이미지들 미리보기
    $(frm.files).on("change", function(e) {
        let attachFiles=[];
        for(let i=0; i<e.target.files.length; i++) {
            attachFile=URL.createObjectURL(e.target.files[i]);
            attachFiles.push(attachFile);
        }
        let temp=Handlebars.compile($("#temp_attach").html());
        $("#attachFiles").html(temp(attachFiles));
    });
    //수정버튼을 클릭한 경우
    $(frm).on("submit", function(e) {
        e.preventDefault();
        let id=$(frm.id).val();
        let title=$(frm.title).val();
        let body=$(frm.body).val();
        if(title=="" || body=="") {
            alert("제목과 내용을 입력하세요!");
            return;
        }
        if(!confirm("변경된 내용으로 수정하실래요?")) return;
        var formData = new FormData();
        formData.append("id", id);
        formData.append("title", title);
        formData.append("body", body);
        let files=$(frm.files)[0].files;
        for(let i=0; i<files.length; i++) {
            formData.append("files", files[i]);
        }
        $.ajax({
            type: "post",
            url: "/api/post/update",
            data: formData,
            processData: false,
            contentType: false,
            success: function() {
                alert("수정이 완료되었습니다.");
                location.href="/post/read/" + id;
            }
        });
    });
</script>
```

6) 첨부된 파일 목록을 출력하는 Controller를 추가하고 게시글 정보 페이지도 수정한다.

[src/main/resources]-[templates]-[post] read.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center mb-5">게시글 정보</h1>
        <div id="post"></div>
        <div id="attachedFiles" class="card p-3 my-3">첨부한 이미지 없음</div>
        <div class="text-center my-3" id="btn-group" style="display:none;">
            <button class="btn btn-primary">정보수정</button>
        </div>
    </div>
</div>
<!-- 게시글 정보 템플릿 -->
<script id="temp" type="x-handlebars-template">
    <div class="card">
        <div class="card-body">
            [{{id}}] {{title}}
            <hr>
            {{body}}
        </div>
        <div class="card-footer">
            Posted on {{fmtdate}} by {{writer}}
        </div>
    </div>
</script>
<!-- 첨부된 파일목록 템플릿 -->
<script id="temp_attached" type="x-handlebars-template">
    <div class="row">
        {{#each .}}
            <div class="col-6 col-md-2"><img src="/display?fileName={{image}}" width="100%"></div>
        {{/each}}
    </div>
</script>
<script>
    let id="[[${id}]]";
    $.ajax({
        type: "get",
        url: "/api/post/read/" + id,
        dataType: "json",
        success: function(data){
            let temp=Handlebars.compile($("#temp").html());
            $("#post").html(temp(data));
            if(sessionStorage.getItem("uid")==data.writer) $("#btn-group").toggle();
        }
    });
    //첨부된 파일목록 출력
    $.ajax({
        type: "get",
        url: "/api/post/getAttach",
        data: { postid: id },
        success: function(data) {
            if(data.length > 0) {
                let temp=Handlebars.compile($("#temp_attached").html());
                $("#attachedFiles").html(temp(data));
            }
        }
    });
    //정보수정 버튼을 클릭한 경우
    $("#btn-group").on("click", ".btn", function(){
        location.href="/post/update/" + id;
    });
</script>
```

7) 게시글 수정 페이지에 첨부된 파일을 출력한다.

[src/main/resources]-[templates]-[post] update.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center mb-5">게시글 정보수정</h1>
        <form name="frm" method="post">
            <input name="id" th:value="${post.id}" type="hidden">
            <input name="title" th:value="${post.title}" class="form-control my-2">
            <textarea name="body" th:text="${post.body}" rows="10" class="form-control my-2"></textarea>
            <div id="attachedFiles" class="card p-3 my-3">첨부한 이미지없음</div>
            <div id="attachFiles" class="card p-3 my-3">첨부할 이미지</div>
            <input name="files" type="file" multiple class="form-control">
            <div class="text-center my-3">
                <button class="btn btn-primary">수정</button>
                <button class="btn btn-secondary" type="reset">취소</button>
            </div>
        </form>
    </div>
</div>
<!-- 첨부된 파일목록 템플릿-->
<script id="temp_attached" type="x-handlebars-template">
    <div class="row">
        {{#each .}}
            <div class="col-6 col-md-2">
                <img src="/display?fileName={{image}}" width="100%">
            </div>
        {{/each}}
    </div>
</script>
<!-- 첨부할 파일목록 템플릿 -->
<script id="tempAttach" type="x-handlebars-template">
    <div class="row">
        {{#each .}}
            <div class="col-6 col-md-2"><img src="{{this}}" width="100%"></div>
        {{/each}}
    </div>
</script>
<script>
    //첨부된 파일목록 출력
    let id=$(frm.id).val();
    $.ajax({
        type: "get",
        url: "/api/post/getAttach",
        data: { postid: id },
        success: function(data) {
            if(data.length > 0) {
                let temp=Handlebars.compile($("#temp_attached").html());
                $("#attachedFiles").html(temp(data));
            }
        }
    });
    //이미지들 미리보기
    $(frm.files).on("change", function(e) {
         ...
    });
    //수정버튼을 클릭한 경우
    $(frm).on("submit", function(e) {
        ...
    });
</script>
```

8) 첨부파일을 삭제하는 Controller와 게시글 수정 페이지를 수정한다.

[src/main/java]-[com/example.controller] PostController.java

```
public class PostRESTController {
    ...
    @RequestMapping(value="/delAttach", method=RequestMethod.POST)
    public void delAttach(int id, String fileName) {
        try {
            File file=new File("c:" + fileName);
            if(file.exists()) {
                file.delete();
                dao.delAttach(id);
            }
        }catch(Exception e) {
            System.out.println("이미지 삭제오류: " + e.toString());
        }
    }
}
```

[src/main/resources]-[templates]-[post] update.html

```
<!-- 첨부된 파일목록 템플릿-->
<script id="temp_attached" type="x-handlebars-template">
    <div class="row">
        {{#each .}}
            <div class="col-6 col-md-2 ">
                <div class="position-relative m-3">
                    <img src="/display?fileName={{image}}" width="100%">
                    <span id="{{id}}" fileName="{{image}}" style="cursor:pointer"
                        class="position-absolute top-0 start-80 translate-middle badge rounded-pill bg-danger">x</span>
                </div>
            </div>
        {{/each}}
    </div>
</script>
<script>
    //이미지 삭제 버튼을 클릭한 경우
    $("#attachedFiles").on("click", ".bg-danger", function() {
        let id=$(this).attr("id");
        let fileName=$(this).attr("fileName");
        if(!confirm(fileName + "파일을 삭제하실래요?")) return;
            $.ajax({
                type: "post",
                url: "/api/post/delAttach",
                data: { id: id, fileName: fileName },
                success: function() {
                    getAttach();
                }
            });
    });
    //첨부된 파일목록 출력
    getAttach();
    function getAttach() {
        let id=$(frm.id).val();
        $.ajax({
            type: "get",
            url: "/api/post/getAttach",
            data: { postid: id },
            success: function(data) {
                ...
            }
        });
    }
</script>
```

- **회원 가입할 경우 비밀번호 암호화**

1) 시큐리티 라이브러리를 추가하고 설정파일을 작성한다.

pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

[src/main/java]-[com.example] SecurityConfig.java

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    public PasswordEncoder getPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.csrf().disable();
        return http.build();
    }
}
```

2) 'header' 페이지의 사용자등록 메뉴를 삭제하고 로그인 페이지에 추가한다.

[src/main/resources]-[templates]-[user] login.html

```
<form name="frm" class="card p-3" method="post">...</form>
<div class="mt-2 text-end"><a href="/user/insert">회원가입</a></div>
```

3) UserController의 사용자 등록(insert)과 로그인(login) 매핑을 아래와 같이 수정한다.

[src/main/java]-[com.example.controller] UserController.java

```
public class UserController {
    @Autowired
    PasswordEncoder passwordEncoder;

    @ResponseBody
    @RequestMapping(value="/login", method=RequestMethod.POST)
    public int login(UserVO vo) {
        int result=0;
        UserVO user=dao.read(vo.getUid());
        if(user != null) {
            if(passwordEncoder.matches(vo.getUpass(), user.getUpass())) {  //암호화된 비밀번호 비교
                result=1;
            }else {
                result=2;
            }
        }
        return result;
    }

    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public String insert(UserVO vo, MultipartHttpServletRequest multi)throws Exception {
        MultipartFile file = multi.getFile("file");
        //이미지가 존재하면 업로드
        if(!file.isEmpty()) { ... }
        vo.setUpass(passwordEncoder.encode(vo.getUpass()));  //비밀번호 암호화
        dao.insert(vo);
        return "redirect:/";
    }
}
```

- **비밀번호 변경하기**

1) 사용자 정보페이지를 로그인한 경우에만 정보와 비밀번호를 수정하도록 아래와 같이 변경한다.

[src/main/resources]-[templates]-[user] read.html

```
<div class="row" xmlns:th="http://www.thymeleaf.org">
    <div class="col">
        <h1 class="text-center mb-5">사용자정보</h1>
        <div class="card p-5">
            <div class="mb-3">
                사용자 아이디: <span th:text="${user.uid}"></span>
                <a href="/user/update/password" class="btn btn-danger btn-sm ms-3">비밀번호변경</a>
            </div>
            <div class="mb-3">
                사용자 성명: <span th:text="${user.uname}"></span>
            </div>
            <div class="mb-3">
                사용자 전화: <span th:text="${user.phone}"></span>
            </div>
            <div class="mb-3">
                사용자 주소: <span th:text="${user.address}"></span>
            </div>
            <div th:if="${user.photo!=null && user.photo!=''}">
                <img th:src="|/display?fileName=${user.photo}|" width=200>
            </div>
            <div th:unless="${user.photo!=null && user.photo!=''}">
                <img src="https://via.placeholder.com/200x250" width=200>
            </div>
        </div>
        <div class="text-center mt-3">
            <a th:href="|/user/update/${user.uid}|">
                <button class="btn btn-primary px-5">정보수정</button>
            </a>
        </div>
    </div>
</div>
```

2) 비밀번호 변경 Mapper와 DAO 인터페이스를 추가한다.

[src/main/resources]-[mapper] UserMapper.xml

```
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.mapper.UserMapper">
    ...
    <update id="update_password">
        update tbl_user set upass=#{upass}
        where uid=#{uid}
    </update>
</mapper>
```

[src/main/java]-[com.example.dao] UserDAO.java

```
public interface UserDAO {
    ...
    public void updatePassword(String uid, String upass);
}
```

[src/main/java]-[com.example.dao] UserDAOImp.java

```
@Repository
public class UserDAOImpl implements UserDAO {
    ...
    @Override
    public void updatePassword(String uid, String upass) {
        HashMap<String,String> map = new HashMap<String,String>();
        map.put("uid", uid);
        map.put("upass", upass);
        session.update(namespace + ".update_password", map);
    }
}
```

3) 비밀번호 변경 Controller와 변경 페이지를 작성한다.

[scr/main/java]-[com.example.controller] UserController.java

```
public class UserController {
    ...
    @RequestMapping("/update/password")
    public String change(Model model) {
        model.addAttribute("pageName", "/user/password");
        return "home";
    }

    @ResponseBody
    @RequestMapping(value="/update/password", method=RequestMethod.POST)
    public void change(String uid, String npass) {
        String upass=passwordEncoder.encode(npass);
        dao.updatePassword(uid, upass);
    }
}
```

[src/main/resources]-[templates]-[user] password.html ①

```
<style>
    form span { width: 150px; }
</style>
<div class="row justify-content-center" xmlns:th="http://www.thymeleaf.org">
    <div class="col-md-6">
        <h1 class="text-center">비밀번호변경</h1>
        <form name="frm" class="card p-5">
            <div class="input-group mb-2">
                <span class="input-group-text">현재비밀번호</span>
                <input name="upass" type="password" class="form-control">
            </div>
            <div class="input-group mb-2">
                <span class="input-group-text">새비밀번호</span>
                <input name="npass" type="password" class="form-control">
            </div>
            <div class="input-group mb-2">
                <span class="input-group-text">새비밀번호확인</span>
                <input name="cpass" type="password" class="form-control">
            </div>
            <div><button class="btn btn-success w-100 my-1">확인</button></div>
        </form>
    </div>
</div>
```

[src/main/resources]-[templates]-[user] password.html ②

```
<script>
    $(frm).on("submit", function(e){
        e.preventDefault();
        const uid = sessionStorage.getItem("uid");
        const upass=$(frm.upass).val();
        const npass=$(frm.npass).val();
        const cpass=$(frm.cpass).val();
        if(upass == "") {
            alert("현재비밀번호를 입력하세요!");
            $(frm.upass).focus();
        }else if(npass == "") {
            alert("새비밀번호를 입력하세요!");
            $(frm.npass).focus();
        }else if(cpass == "") {
            alert("새비밀번호확인을 입력하세요!");
            $(frm.cpass).focus();
        }else if(npass != cpass) {
            alert("새비밀번호와 새비밀번호확인이 일치하지 않습니다!");
            $(frm.npass).val("");
            $(frm.cpass).val("");
            $(frm.npass).focus();
        }else{
            $.ajax({
                type:"post",
                url:"/user/login",
                data:{uid:uid, upass:upass},
                success:function(data){
                    if(data != 1){
                        alert("현재비밀번호가 일치하지 않습니다!");
                        $(frm.upass).val("");
                        $(frm.upass).focus();
                    }else{
                        if(!confirm("새로운 비밀번호로 변경하실래요?")) return;
                        $.ajax({
                            type:"post",
                            url:"/user/update/password",
                            data:{uid:uid, npass:npass},
                            success:function() {
                                alert("비밀번호가 변경되었습니다!");
                                document.cookie = "uid='';path=/;expires=Thu, 01 Jan 1999 00:00:10 GMT;";  //쿠키삭제
                                sessionStorage.clear();  //세션삭제
                                location.href="/user/login";
                            }
                        });
                    }
                }
            });
        }
    });
</script>
```

- **오라클 데이터베이스 연결**

1) [Properties]-[Java Build Path]-[Libraries]-[Classpath]-[Add External JARs...] 'ojdbc.jar' 파일을 추가한다.

2) [com.example] 패키지에 MysqlConfig 설정 파일을 수정한다.

[src/main/java]-[com.example] MyselConfig.java

```
package com.example;

import javax.sql.DataSource;
import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.io.Resource;
import org.springframework.core.io.support.PathMatchingResourcePatternResolver;
import org.springframework.jdbc.datasource.DataSourceTransactionManager;
import org.mybatis.spring.SqlSessionFactoryBean;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;

@Configuration
@PropertySource("classpath:/application.properties")
public class MysqlConfig {
    @Bean
    @ConfigurationProperties(prefix="spring.datasource.hikari")
    public HikariConfig hikariConfig() {
        return new HikariConfig();
    }

    @Primary
    @Bean(name="dataSource1")
    public DataSource dataSource() throws Exception {
        DataSource dataSource = new HikariDataSource(hikariConfig());
        System.out.println("dataSource:" + dataSource.toString());
        return dataSource;
    }

    @Primary
    @Bean(name="factory1")
    public SqlSessionFactory sqlSessionFactory(@Qualifier("dataSource1") DataSource dataSource) throws Exception {
        SqlSessionFactoryBean sessionFactory=new SqlSessionFactoryBean();
        sessionFactory.setDataSource(dataSource);
        Resource[] resources=new PathMatchingResourcePatternResolver().getResources("classpath:/mapper/*Mapper.xml");
        sessionFactory.setMapperLocations(resources);
        return sessionFactory.getObject();
    }

    @Primary
    @Bean(name="session1")
    public SqlSessionTemplate sqlSessionTemplate(@Qualifier("factory1") SqlSessionFactory sqlSessionFactioy) throws Exception {
        return new SqlSessionTemplate(sqlSessionFactioy);
    }

    @Primary
    @Bean(name="transaction1")
    public DataSourceTransactionManager txManager(@Qualifier("dataSource1") DataSource dataSource) {
        return new DataSourceTransactionManager(dataSource);
    }
}
```

3) [com.example] 패키지에 OracleConfig 설정 파일을 생성한다.

[src/main/java]-[com.example] OracleConfig.java

```
package com.example;

import javax.sql.DataSource;
import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.Resource;
import org.springframework.core.io.support.PathMatchingResourcePatternResolver;
import org.springframework.jdbc.datasource.DataSourceTransactionManager;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
import org.mybatis.spring.SqlSessionFactoryBean;

@Configuration
public class OracleConfig {
    @Bean(name="dataSource2")
    public DataSource dataSource() throws Exception {
        HikariConfig hikariConfig=new HikariConfig();
        hikariConfig.setUsername("system");
        hikariConfig.setPassword("1234");
        hikariConfig.setDriverClassName("net.sf.log4jdbc.sql.jdbcapi.DriverSpy");
        hikariConfig.setJdbcUrl("jdbc:log4jdbc:oracle:thin:@localhost:1521/xe");
        DataSource dataSource = new HikariDataSource(hikariConfig);
        System.out.println("dataSource2:" + dataSource.toString());
        return dataSource;
    }

    @Bean(name="factory2")
    public SqlSessionFactory sqlSessionFactory(@Qualifier("dataSource2") DataSource dataSource) throws Exception {
        SqlSessionFactoryBean sessionFactory=new SqlSessionFactoryBean();
        sessionFactory.setDataSource(dataSource);
        Resource[] resources=new PathMatchingResourcePatternResolver().getResources("classpath:/mapper_oracle/*Mapper.xml");
        sessionFactory.setMapperLocations(resources);
        return sessionFactory.getObject();
    }

    @Bean(name="session2")
    public SqlSessionTemplate sqlSessionTemplate(@Qualifier("factory2") SqlSessionFactory sqlSessionFactioy) throws Exception {
        return new SqlSessionTemplate(sqlSessionFactioy);
    }

    @Bean(name="transaction2")
    public DataSourceTransactionManager txManager(@Qualifier("dataSource2") DataSource dataSource) {
        return new DataSourceTransactionManager(dataSource);
    }
}
```

4) application.properties 파일에 아래 내용을 추가한다.

[src/main/resources] application.properties

```
...
spring.datasource.oracle.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy
spring.datasource.oracle.jdbc-url=jdbc:log4jdbc:oracle:thin:@localhost:1521/xe
spring.datasource.oracle.username=system
spring.datasource.oracle.password=1234
```

5) [src/main/resources]에 [mapper_oracle] 폴더를 생성한 후 OracleMapper xml 파일을 작성한다.

[src/main/resources]-[mapper_oracle] OracleMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.mapper_oracle.OracleMapper">
    <select id="now" resultType="string">
        select sysdate from dual
    </select>
</mapper>
```

6) [com.example.dao] 패키지를 생성한 후 OracleDAO 인터페이스와 OracleDAOImpl 클래스를 작성한다.

[src/main/java]-[com.example.dao_oracle] OracleDAOImpl.java

```
public interface OracleDAO {
    public String now();
}
```

[src/main/java]-[com.example.dao_oracle] OracleDAOImpl.java

```
@Repository
public class OracleDAOImpl implements OracleDAO {
    @Autowired
    @Qualifier("session2")
    SqlSession session
    String namespace="com.example.mapper_oracle.OracleMapper";

    public String now() {
        session.selectOne(namespace + ".now");
    }
}
```

7) 오라클 DB에 접속하는지 작성한 OracleDAO 클래스를 테스트한다.

[src/test/java]-[com.example] ApplicationTests.java

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringBootTest
class OracleTests {
    @Autowired
    private OracleDAO dao;

    @Test
    public void nowOracle() {
        System.out.println(dao.now());
    };
}
```

10. 부 록

- **저장프로시저(MySql)**

예제1) 게시판의 bno가 200보다 큰 게시글 목록을 출력

```
call list_bbs(200)
```

```
DROP PROCEDURE IF EXISTS LiST_BBS;

DELIMITER //
CREATE PROCEDURE LIST_BBS(in i_bno int)
BEGIN
    select * from tbl_board where bno > i_bno;
END
//

DELIMITER;
```

```
mapper.xml
```

```
<select id="listBoard" statementType="CALLABLE" resultType="com.example.domain.BoardVO">
    { call list_board(#{bno}) }
</select>
```

예제2) 게시판의 bno가 200보다 큰 게시글의 개수를 출력

```
call cnt_bbs(200)
```

```
CREATE DEFINER=`web`@`localhost` PROCEDURE `CNT_BBS`(in i_bno int)
BEGIN
    select count(*) from tbl_board
    where bno > i_bno;
END
```

```
mapper.xml
```

```
<select id="listBoard" statementType="CALLABLE" resultType="int">
    { call cnt_board(#{bno}) }
</select>
```

예제3) email을 입력받아 id에 email이 존재하지 않으면 tbl_user 테이블에 insert

```
call add_user('test01@eamil.com');
```

```
CREATE DEFINER=`web`@`localhost` PROCEDURE `ADD_USER`(in i_email char(8))
BEGIN
    DECLARE v_count int;

    select count(*) into v_count
    FROM tbl_user
    where uid = i_email;

    IF(v_count = 0)THEN
        insert into tbl_user(uid) values(i_email);
    END IF;
END
```

```
mapper.xml
```

```
<insert id="naverinsert" statementType="CALLABLE">
    { call add_user(#{email}) }
</insert>
```

예제4) 댓글의 게시글 번호를 읽어 게시글의 viewcnt를 1씩 증가시킨다.

```
call update_cnt(1);
```

```
CREATE DEFINER=`web`@`localhost` PROCEDURE `UPDATE_CNT`(in i_bno int)
BEGIN
    update tbl_board
    set viewcnt = viewcnt + 1 where bno=i_bno;
END
```

---

mapper.xml

```
<update id="updatecnt" statementType="CALLABLE">
    { call update_cnt(#{bno}) }
</update>
```

```
call list_courses_students('221', @ccnt, @scnt); select @ccnt;  특정교수가 지도하는 학생과 담당과목출력
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `LIST_COURSES_STUDENTS`(
    in i_pcode char(3), out o_ccnt int, out o_scnt int)
BEGIN
    select count(*) into o_ccnt from courses where instructor=i_pcode;
    select count(*) into o_scnt from students where advisor=i_pcode;
    select * from courses where instructor=i_pcode;
    select * from students where advisor=i_pcode;
END
```

[src/main/resources]-[mapper] MysqlMapper.xml

```
<mapper namespace="com.example.mapper.MysqlMapper">
    <resultMap id="clist" type="hashmap"/>
    <resultMap id="slist" type="hashmap"/>
    <select id="procedure" statementType="CALLABLE"  resultMap="clist, slist">
        { call LIST_COURSES_STUDENTS(
            #{pcode},
            #{ccnt,  mode=OUT, jdbcType=INTEGER, javaType=int},
            #{scnt , mode=OUT, jdbcType=INTEGER, javaType=int})
        }
    </select>
</mapper>
```

[scr/main/java]-[com.example.mapper] MysqlDAO.java

```
public interface MysqlDAO {
    public List<Map<String, Object>> procedure(Map<String,Object> map);
}
```

[src/main/java]-[com.example.controller] MysqlController.java

```
@RestController
public class MySqlController {
    @Autowired
    MysqlDAO dao;

    @RequestMapping("/procedure")
    public Map<String,Object> plist(String pcode) {
        Map<String,Object> map=new HashMap<String, Object>();
        map.put("pcode", pcode);

        List<Map<String,Object>> list=dao.procedure(map);
        map.put("clist", list.get(0));
        map.put("slist", list.get(1));
        return map;
    }
}
```

- **저장프로시저(Oracle)**

```
variable cur refcursor; variable cnt number; exec list_product('동서가구', :cur, :cnt);
```

```
create or replace PROCEDURE LIST_PRODUCT(
    i_company product.company%TYPE,
    o_cursor out sys_refcursor,
    o_count out int
)
AS
BEGIN
    OPEN o_cursor FOR select * from product where company=i_company;
    select count(*) into o_count
    from product
    where company=i_company;
END LIST_PRODUCT;
```

[src/main/resources]-[mapper_oracle] OracleMapper.xml

```
<mapper namespace="com.example.mapper_oracle.OracleMapper">
    <select id="getTime" resultType="string">
        select sysdate from dual
    </select>
    <resultMap id="listMap" type="hashmap"/>
    <select id="listProduct" statementType="CALLABLE" parameterType="hashmap">
        {CALL list_product(
            #{company},
            #{list, mode=OUT, jdbcType=CURSOR, javaType=ResultSet, resultMap=listMap},
            #{count, mode=OUT, jdbcType=INTEGER, javaType=int})
        }
    </select>
</mapper>
```

[scr/main/java]-[com.example.mapper_oracle] OracleDAO.java

```
public interface OracleMapper {
    public void listProduct(HashMap<String,Object> paraMap);
}
```

[scr/main/java]-[com.example.mapper_oracle] OracleDAO.java

```
@Repository
public interface OracleMapper {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper_oracle.OracleMapper"

    public void listProduct(HashMap<String,Object> paraMap){
        session.selsecList(namespace + ".listProduct");
    }
}
```

[src/main/java]-[com.example.controller] OracleController.java

```
@RestController
public class OracleController {
    @Autowired
    OracleMapper mapper;

    @RequestMapping("/product/list")
    public HashMap<String,Object> list(String company){
        HashMap<String,Object> paraMap=new HashMap<String,Object>();
        paraMap.put("company", company);
        mapper.listProduct(paraMap);

        return paraMap;
    }
}
```

- **웹 소켓(WebSocket)**

- **웹 소켓이란?**

Web Socket은 웹서버와 웹브라우저가 지속적으로 연결된 TCP 라인을 통해 실시간으로 데이터를 주고받을 수 있도록 하는 HTML5의 새로운 사양이다. 따라서 WebSocket을 이용하면 양방향 통신이 가능하다. 이와 같은 특징으로 웹에서도 게시판 프로젝트에서 댓글 알림이나 실시간 채팅, 게임, 실시간 주식 차트와 같은 실시간이 요구되는 응용프로그램의 개발을 한층 효과적으로 구현할 수 있게 되었다.

- **웹소켓을 이용한 채팅 프로그램**

1) pom.xml 파일에 websocket 라이브러리를 추가한다. (* security 라이브러리 뒤에 추가한다.)

pom.xml

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-websocket</artifactId>
    <version>4.1.7.RELEASE</version>
</dependency>
```

2) com.example.controller 패키지에 EchoHandler 클래스 파일을 생성한다.

[src/main/java]-[com.example.controller] EchoHandler.java

```
public class EchoHandler extends TextWebSocketHandler{
    //세션 리스트
    private List<WebSocketSession> listSession = new ArrayList<WebSocketSession>();

    //클라이언트가 연결 되었을 때 실행
    @Override
    public void afterConnectionEstablished(WebSocketSession session) throws Exception {
        listSession.add(session);
        System.out.println("연결됨 : "+ session.getId());
    }

    //클라이언트가 웹소켓 서버로 메시지를 전송했을 때 실행
    @Override
    protected void handleTextMessage(WebSocketSession session, TextMessage message) throws Exception {
        String strMessage=message.getPayload();
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
        String strDate=sdf.format(new Date());
        strMessage = strMessage + "|" + strDate;
        System.out.println("id: " + session.getId()+ " message: " + strMessage);

        //모든 유저에게 메세지 출력
        for(WebSocketSession webSocketSession : listSession){
            webSocketSession.sendMessage(new TextMessage(strMessage));
        }
    }

    //클라이언트 연결을 끊었을 때 실행
    @Override
    public void afterConnectionClosed(WebSocketSession session, CloseStatus status) throws Exception {
        listSession.remove(session);
        System.out.println("연결 끊김: " + session.getId());
    }
}
```

3) sevlet-context.xml [Namespaces]탭에서 websocket을 체크한다.

[scr]-[main]-[webapp]-[WEB-INF]-[spring]-[appServlet] servlet-context.xml

```
<beans:bean id="echoHandler" class="com.example.controller.EchoHandler"/>
<websocket:handlers>
    <websocket:mapping handler="echoHandler" path="/sock_chat"/>
    <websocket:sockjs/>
</websocket:handlers>
```

4) 채팅 프로그램을 작성한다.

[src]-[main]-[webapp]-[resources] chat.css

```
*{
    margin: 0px; padding: 0px;
}

div.header {
   position: sticky;
   top: 0;
}

.chat_wrap .header {
    font-size: 20px;
    padding: 15px 0px;
    background: #F18C7E;
    color: white;
    text-align: center;
}

.chat_wrap #chat {
    padding-bottom: 80px;
    width: 100%;
    font-size: 12px;
}

.chat_wrap #chat .left {
    text-align: left;
}

.chat_wrap #chat .right {
    text-align: right;
}

.chat_wrap #chat .sender {
    margin: 10px 25px 0px 10px;
    font-weight: bold;
}

.chat_wrap #chat .message {
    display: inline-block;
    margin: 5px 20px 10px 10px;
    max-width: 75%;
    border: 1px solid gray;
    padding: 5px;
    border-radius: 5px;
    background-color: #FCFCFC;
    text-align: left;
}

.chat_wrap #chat .date {
    margin: 5px 20px 10px 10px;
    font-size: 10px;
}

.chat_wrap .input-div {
    position: fixed;
    bottom: 0px;
    width: 100%;
    background-color: #FFF;
    text-align: center;
    border-top: 1px solid #F18C7E;
}

#txtMessage {
    width: 100%; height: 80px;
    border: none; padding: 5px;
}
#txtMessage:focus {
    outline: none;
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views] chat.jsp

```
<head>
    <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.5/sockjs.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.1/handlebars.js"></script>
    <link rel="stylesheet" href="/resources/chat.css"/>
    <style>
        .sender .right { display: none; }
    </style>
</head>
<body>
    <div class="chat_wrap">
        <div class="header">채 팅 방</div>
        <div id="chat"></div>
        <script id="temp" type="text/x-handlebars-template">
            <div class="{{printLeftRight sender}}">
                <div class="sender {{printLeftRight sender}}">{{sender}}</div>
                <div class="message">{{message}}</div>
                <div class="date">{{date}}</div>
            </div>
        </script>

        <script>
            var uid = "${id}";
            Handlebars.registerHelper("printLeftRight", function(sender) {
                if(uid != sender) {
                    return "left";
                }else {
                    return "right";
                }
            });
        </script>
        <div class="input-div">
            <textarea id="txtMessage" placeholder="Press Enter for send message."></textarea>
        </div>
    </div>
</body>

<script type="text/javascript">
    var uid = "${uid}";

    $("#txtMessage").on("keydown", function(e) {
        if(e.keyCode == 13 && !e.shiftKey) {
            e.preventDefault();
            var message = $(this).val();
            if(message=="") {
                alert("내용을 입력하세요!");
                return;
            }
            $("#txtMessage").val("");
            sock.send(uid + "|" + message);  // 메시지 전송
        }
    });

    //소켓설정
    var sock = new SockJS("http://localhost:8080/sock_chat/");
    sock.onmessage = onMessage;

    //서버로부터 메시지를 받았을 때
    function onMessage(msg) {
        var items = msg.data.split("|");
        var sender=items[0];
        var message=items[1];
        var date=items[2];
        var data={"sender":sender, "message":message, "date":date};
        var template = Handlebars.compile($("#temp").html());
        $("#chat").append(template(data));
        window.scrollTo(0, $("#chat").prop("scrollHeight"));  //스크롤바 아래 고정
    }
</script>
```

[src]-[main]-[webapp]-[WEB-INF]-[views] home.jsp

```
<div id="menu">
    ...
    <span class="item"><a href="#" id="chat">채팅방</a></span>
</div>

<script>
    $("#chat").on("click", function(e){
        e.preventDefault();
        window.open("/chat", "chat", "width=500, height=700, top=200, left=900");
    });
</script>
```

[src/main/java]-[com.example.controller] HomeController.java

```
@Controller
public class HomeController {
    ...
    @RequestMapping("/chat")
    public String chat(){
        return "chat";  //chat.jsp 페이지 출력
    }
}
```

- 채팅 데이터 테이블에 저장

tbl_chat 테이블 생성

```
create table tbl_chat(
    id int auto_increment primary key,
    sender varchar(100),
    message varchar(1000),
    regdate datetime default now(),
    foreign key(sender) references tbl_user(uid)
);
```

[scr/main/java]-[com.example.domain] ChatVO.java

```
public class ChatVO extends UserVO{
    private int id;
    private String sender;
    private String message;
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss",timezone="Asia/Seoul")
    private Date regdate;
    //setter(), getter(), toString() 메서드
}
```

[src/main/resources]-[mapper] ChatMapper.xml

```
<mapper namespace="com.example.mapper.ChatMapper">
    <select id="list" resultType="com.example.domain.ChatVO">
        select c.*, photo from tbl_chat c, tbl_user
        where uid=sender order by id
    </select>
    <insert id="insert">
        insert into tbl_chat(sender, message)
        values(#{sender}, #{message})
    </insert>
    <delete id="delete">
        delete from tbl_chat
        where id = #{id}
    </delete>
</mapper>
```

[scr/main/java]-[com.example.mapper] ChatDAO.java

```
public interface ChatDAO {
    public List<ChatVO> list();
    public void insert(ChatVO vo);
    public void delete(int id);
}
```

[scr/main/java]-[com.example.mapper] ChatDAOImpl.java

```
@Repository
public class ChatDAOImpl implements ChatDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.ChatMapper";

    @Override
    public List<ChatVO> list() {
        return session.selectList(namespace + ".list");
    }

    @Override
    public void insert(ChatVO vo) {
        session.insert(namespace + ".insert", vo);
    }

    @Override
    public void delete(int id) {
        session.delete(namespace + ".delete", id);
    }
}
```

[src/main/java]-[com.example.controller] ChatController.java

```
@RestController
public class ChatController {
    @Resource(name="uploadPath")
    private String path;

    @Autowired
    ChatDAO dao;

    @RequestMapping("/chat.json")
    public List<ChatVO> list(){
        return dao.list();
    }

    @RequestMapping(value="/chat/insert", method=RequestMethod.POST)
    public void insert(ChatVO vo){
        dao.insert(vo);
    }

    @RequestMapping(value="/chat/delete", method=RequestMethod.POST)
    public void insert(int id){
        dao.delete(id);
    }

    //이미지파일 출력
    @RequestMapping("/display")
    public byte[] display(String file)throws Exception{
        FileInputStream in=new FileInputStream(path + "/" + file);
        byte[] image=IOUtils.toByteArray(in);
        in.close();
        return image;
    }
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views] chat.jsp

```
<body>
    ...
    <script id="temp" type="text/x-handlebars-template">
    {{#each .}}
        <div class="{{printLeftRight sender}}">
            <div class="sender {{printLeftRight sender}}">
                <img src="/display?file={{photo}}" style="width:50px;border-radius:50%"/>{{sender}}
            </div>
            <div class="message">
                {{message}}<a href="{{id}}" class="{{printLeftRight sender}}">X</a>
            </div>
        </div>
    {{/each}}
</body>

<script type="text/javascript">
    var uid = "${uid}";
    getList();

    function getList() {  //채팅 목록출력
        $.ajax({
           type : "get",
           url : "/chat.json",
           dataType : "json",
           success : function(data) {
              var template = Handlebars.compile($("#temp").html());
              $("#chat").html(template(data));
              window.scrollTo(0, $("#chat").prop("scrollHeight"));  //스크롤바 아래 고정
           }
        });
     }

    //채팅 데이터 삭제
    $("#chat").on("click", ".message a", function(e){
        e.preventDefault();
        var id=$(this).attr("href");
        if(!confirm(id + "을(를) 삭제하실래요?")) return;
        $.ajax({
            type:"post",
            url: "/chat/delete",
            data: {id:id},
            success: function(){
                sock.send("delete|" + id);
            }
        });
    });

    //채팅 데이터 입력
    $("#txtMessage").on("keydown", function(e) {
        if (e.keyCode == 13 && !e.shiftKey) {
            ...
            $.ajax({  //메시지 테이블에 등록
                type: "post",
                url: "/chat/insert",
                data: {sender: uid, message:message},
                success: function(){
                    sock.send("insert|" + uid + "|" + message);  //메시지 보내기
                }
            });
        }
    });

    //웹 소켓 생성
    var sock = new SockJS("http://localhost:8088/sock_chat/");
    sock.onmessage = onMessage;
    //서버로부터 메시지 받기
    function onMessage(msg) {
        getList();
    }
</script>
```

- **웹소켓을 이용한 푸시 알림 프로그램**

[src/main/java]-[com.example.controller] HomeController.java

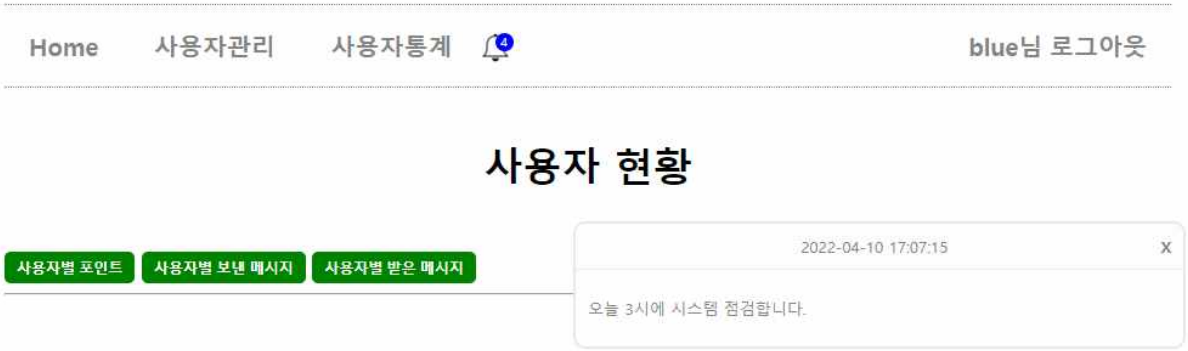
```
@Controller
public class HomeController {
    ...
    @RequestMapping(value = "/notice", method = RequestMethod.GET)
    public String notice(Locale locale, Model model) {
        model.addAttribute("pageName", "notice.jsp");
        return "home";
    }
}
```

[src/main/java]-[com.example.controller] NoticeHandler.java

```
public class NoticeHandler extends TextWebSocketHandler{
    List<WebSocketSession> listSession=new ArrayList<>();

    @Override
    public void afterConnectionEstablished(WebSocketSession session) throws Exception {
        System.out.println("Notice 연결됨:" + session.getId() );
        listSession.add(session);
        System.out.println("Notice 연결된갯수:" + listSession.size());
    }

    @Override
    public void afterConnectionClosed(WebSocketSession session, CloseStatus status) throws Exception {
        System.out.println("Notice 연결끊김:" + session.getId());
        listSession.remove(session);
        System.out.println("Notice 연결된갯수:" + listSession.size());
    }

    //클라이언트(브라우저)에서 서버로 메시지를 보냈을 경우
    @Override
    protected void handleTextMessage(WebSocketSession session, TextMessage message) throws Exception {
        String strMessage=message.getPayload();

        //연결된 세션들에게 메시지를 보낼 경우
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        String strDate = sdf.format(new Date());
        strMessage += "|" + strDate;
        System.out.println("메시지:" + strMessage);
        for(WebSocketSession webSocketSession:listSession){
            webSocketSession.sendMessage(new TextMessage(strMessage));
        }
    }
}
```

[scr]-[main]-[webapp]-[WEB-INF]-[spring]-[appServlet] servlet-context.xml

```
<beans:beans xmlns="http://www.springframework.org/schema/mvc" .../>
    ...
    <beans:bean id="noticeHandler" class="com.example.controller.NoticeHandler"/>
    <websocket:handlers>
        <websocket:mapping handler="noticeHandler" path="/sock_notice"/>
        <websocket:sockjs/>
    </websocket:handlers>
</beans:beans>
```



[src]-[main]-[webapp]-[resources] notice.css

```
@CHARSET "UTF-8";
#bell {
    display: inline-block;
    position: absolute;
    background-image: url('/resources/bell.png');
    background-repeat: no-repeat;
    background-size: 100%;
    width: 30px;
    height: 30px;
    cursor: pointer;
}
#count {
    display: inline-block;
    position: relative;
    top: 3px;
    left: 15px;
    z-index: 3;
    width: 15px;
    height: 15px;
    background: blue;
    border-radius: 15px;
    text-align: center;
    font-size: 10px;
    color: white;
}
#divNotice {
    position: fixed;
    width: 500px;
    border:2px solid #e2e2e2;
    border-radius:10px;
    background: white;
    color: gray;
    bottom: 10;
    right: 10;
    display: none;
}
#divHeader {
    background: #F6F6F6;
    padding:10px;
    text-align: center;
    border-radius:10px 10px 0px 0px;
    border-bottom: 1px solid #E2E2E2;
    font-size: 13px;
    color: gray;
}
#divMessage {
    padding: 10px;
    font-size: 13px;
}
#close {
    float: right;
    font-weight: bold;
    cursor: pointer;
    font-size: 13px;
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views] home.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html>
<head>
    <link href="/resources/home.css" rel="stylesheet">
    <link href="/resources/notice.css" rel="stylesheet">
    <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.5/sockjs.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.1/handlebars.js"></script>
    <script src="/resources/pagination.js"></script>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript" src="/resources/chart.js"></script>
</head>
<body>
    <div id="page">
        <div id="header">
            <img src="/resources/back.jpg" width=960 />
        </div>
        <div id="menu">
            <span><a href="/">Home</a></span>
            <span><a href="/user/list">사용자관리</a></span>
            <span><a href="/chart/user">사용자통계</a></span>
            <c:if test="${uid==null}">
                <span style="float: right;"><a href="/user/login">로그인</a></span>
            </c:if>
            <c:if test="${uid!=null}">
                <small id="bell" onClick="location.href='/notice'">
                    <small id="count">0</small>
                </small>
                <span style="float: right;">
                    <a href="/user/update?uid=${uid}">${uid}님</a>
                    <a href="/user/logout">로그아웃</a>
                </span>
            </c:if>
        </div>
        <div id="content">
            <div id="divNotice">
                <div id="divHeader">
                    <span id="date">2001-01-01</span>
                    <span id="close">X</span>
                </div>
                <div id="divMessage">
                    <p id="message">여기에 알림이 출력됩니다.</p>
                </div>
            </div>
            <jsp:include page="${pageName}"/>
        </div>
        <div id="footer">
            <h3>Copyright 2022 인천일보 아카데미. All Rights Reserved.</h3>
         </div>
    </div>
</body>
</html>
```

---

tbl_user 테이블 변경

```
alter table tbl_user
add notice_read_date datetime default '2000-01-01';
```

---

tbl_notce 테이블 생성 및 샘플 데이터 입력

```
create table tbl_notice(
    id int auto_increment primary key,
    content text,
    sender varchar(100),
    regdate datetime default now(),
    foreign key(sender) references tbl_user(uid)
);

insert into tbl_notice(content,sender) values('2021-05-01 오후 6시 이후 정전입니다', 'red');
insert into tbl_notice(content,sender) values('오늘은 시스템 점검일입니다.', 'red');
insert into tbl_notice(content,sender) values('오늘 오후3시에 삼겹살 폭탄세일 합니다.', 'red');
```

[scr/main/java]-[com.example.domain] NoticeVO.java

```
public class NoticeVO {
    private int id;
    private String content;
    private String sender;
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss",timezone="Asia/Seoul")
    private Date regdate;
    //setter(), getter(), toString() 메서드
}
```

[src/main/resources]-[mapper] NoticeMapper.xml

```
<mapper namespace="com.example.mapper.NoticeMapper">
    <select id="list" resultType="com.example.domain.NoticeVO">
        select n.*, uname from tbl_notice n, tbl_user
        where sender=uid
        order by id desc
    </select>
    <insert id="insert">
        insert into tbl_notice(sender, content)
        values(#{sender}, #{content})
    </insert>
    <delete id="delete">
        delete from tbl_notice
        where id=#{id}
    </delete>
    <select id="unreadCount" resultType="int">
        select count(*) from tbl_notice
        where regdate > (select notice_read_date from tbl_user where uid=#{uid});
    </select>
    <update id="updateReadDate">
        update tbl_user set notice_read_date = now()
        where uid=#{uid}
    </update>
</mapper>
```

[scr/main/java]-[com.example.mapper] NoticeDAO.java

```
public interface NoticeDAO {
    public List<NoticeVO> list();
    public void insert(NoticeVO vo);
    public void delete(int id);
    public int unreadCount(String uid);
    public void updateReadDate(String uid);
}
```

[scr/main/java]-[com.example.mapper] NoticeDAOImpl.java

```
@Repository
public class NoticeDAOImpl implements NoticeDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.NoticeMapper";

    @Override
    public List<NoticeVO> list() {
        return session.selectList(namespace + ".list");
    }

    @Override
    public void insert(NoticeVO vo) {
        session.insert(namespace + ".insert", vo);
    }

    @Override
    public void delete(int id) {
        session.delete(namespace + ".delete", id);
    }

    @Override
    public int unreadCount(String uid) {
        return session.selectOne(namespace + ".unreadDate", uid);
    }

    @Override
    public void updateReadDate(String uid) {
        session.update(namespace + ".updateReadDate", uid);
    }
}
```

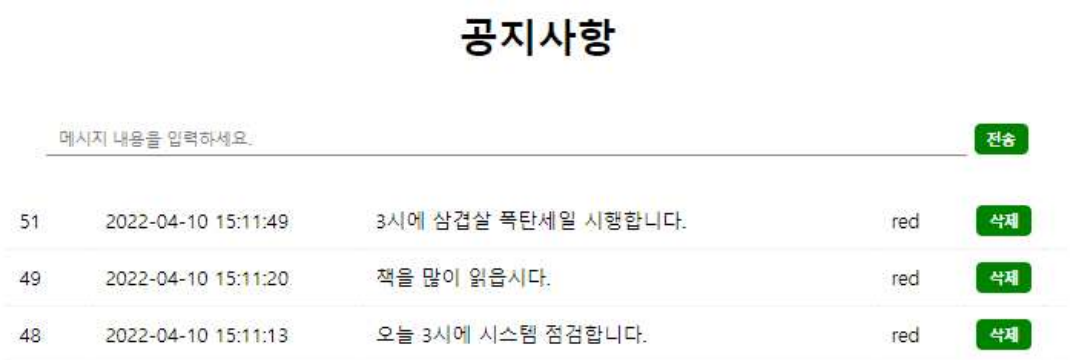
[src/main/java]-[com.example.controller] NoticeController.java

```
@RestController
@RequestMapping("/notice")
public class NoticeController {
    @Autowired
    NoticeDAO dao;

    @RequestMapping("/list.json")
    public List<NoticeVO> list(String uid){
        dao.updateReadDate(uid);
        return dao.list();
    }

    @RequestMapping(value="/insert", method=RequestMethod.POST)
    public void insert(NoticeVO vo){
        dao.insert(vo);
    }

    @RequestMapping(value="/delete", method=RequestMethod.POST)
    public void delete(int id){
        dao.delete(id);
    }

    @RequestMapping("unreadCount")
    public int noticeUnreadCount(String uid){
        return dao.unreadCount(uid);
    }
}
```

[src]-[main]-[webapp]-[WEB-INF]-[views] home.jsp

```
<script>
    //로그인한 경우에만 웹 소켓 생성
    var uid = "${uid}";
    if (uid !=null ) {
        var sock_notice=new SockJS("http://localhost:8088/sock_notice/");
        sock_notice.onmessage = onNoticeMessage;

        //읽지 않는 알림 개수를 출력해 count에 보여준다.
        $.ajax({
            type: "get",
            url: "/notice/unreadCount",
            data: {"uid": uid},
            success:function(data) {
                $("#count").html(data);
            }
        });
    }

    //서버로부터 메시지를 받은 경우
    function onNoticeMessage(msg) {
        var items = msg.data.split("|");
        var sender=items[0];
        var message=items[1];
        var date=items[2];

        var pageName="${pageName}";
        if(pageName != "notice.jsp") {
            $("#divNotice").show();
            $("#message").html(message);
            $("#date").html(date);

            //읽지 않는 알림 개수를 출력해 count에 보여준다.
            $.ajax({
                type: "get",
                url: "/notice/unreadCount",
                data: {"uid": uid},
                success:function(data) {
                    $("#count").html(data);
                }
            });
        }else if(sender != uid) {
            getNotice();  //notice.jsp 페이지이면서 보낸이가 로그인한 유저가 아닌 경우 목록 Refresh한다.
        }
    }

    $("#close").on("click", function(){
        $("#divNotice").hide();
    });
</script>
```

- 공지 사항 실행 결과



- **그래프 차트 출력**

1) ChartMapper와 CartDAO 인터페이스와 구현체를 작성한다.

[src/main/resources]-[mapper] ChartMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.mapper.ChartMapper">
    <select id="courses" resultType="hashmap">
        select e.lcode, lname, avg(grade)
        from enrollments e, courses c
        where  e.lcode=c.lcode
        group by lcode
    </select>
    <select id="students" resultType="hashmap">
        select e.scode, sname, avg(grade)
        from enrollments e, students s
        where e.scode=s.scode
        group by scode
    </select>
    <select id="department" resultType="hashmap">
        select dept, count(*)
        from students
        group by dept
    </select>
</mapper>
```

[src/main/java]-[com.example.dao] ChartDAO.java

```
public interface ChartDAO {
    public List<HashMap<String, Object>> courses();
    public List<HashMap<String, Object>> students();
    public List<HashMap<String, Object>> departments();
}
```

[src/main/java]-[com.example.dao] ChartDAOImpl.java

```
@Repository
public class ChartDAOImpl implements ChartDAO {
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.ChartMapper";

    @Override
    public List<HashMap<String, Object>> courses() {
        return session.selectList(namespace + ".courses");
    }

    @Override
    public List<HashMap<String, Object>> students() {
        return session.selectList(namespace + ".students");
    }

    @Override
    public List<HashMap<String, Object>> departments() {
        return session.selectList(namespace + ".department");
    }
}
```

2) ChartController를 작성한다.

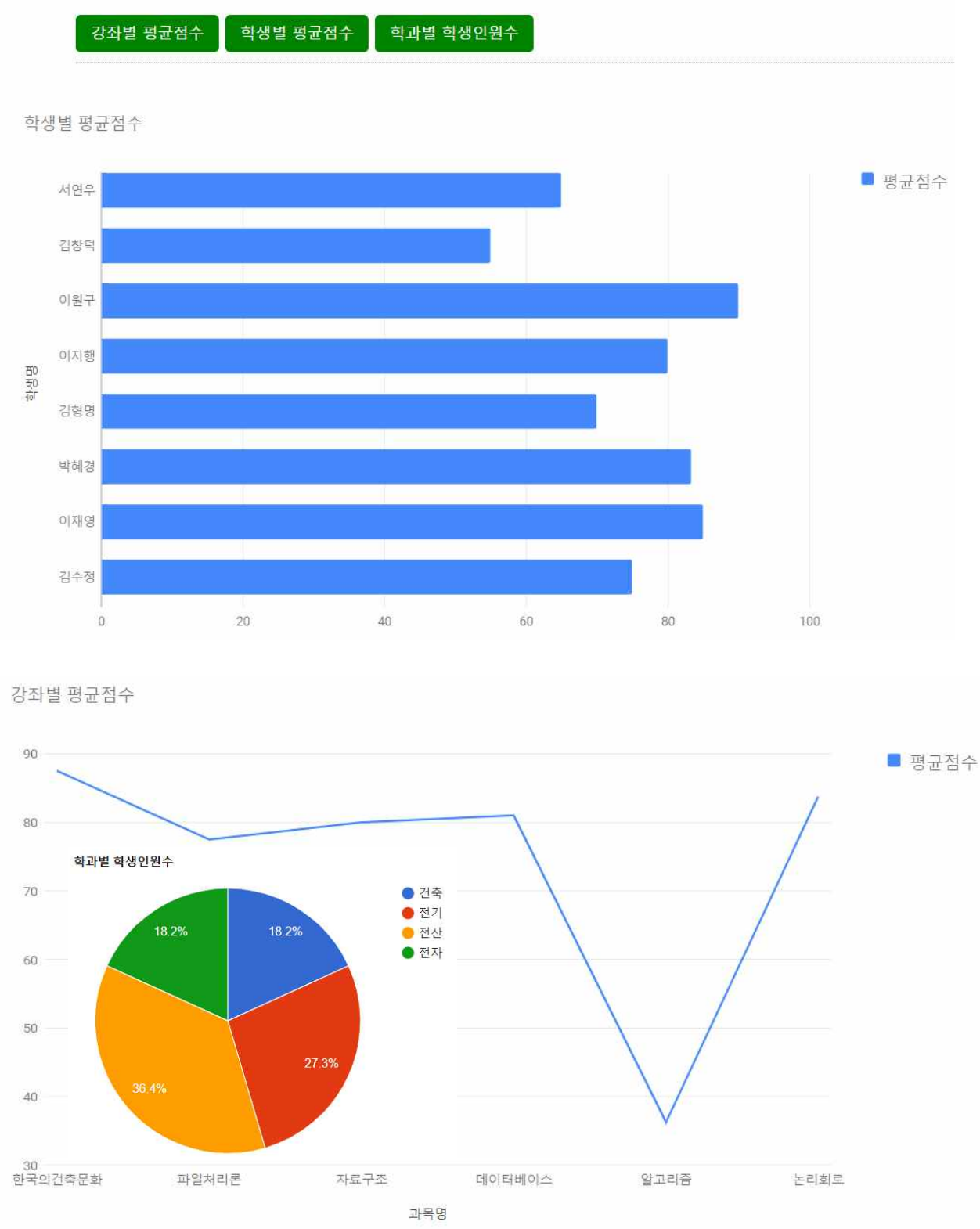
[scr/main/java]-[com.example.controller]-[ChartController.java]

```
@Controller
@RequestMapping("/chart")
public class CartController {
    @Autowired
    ChartDAO dao;

    @RequestMapping("/courses")
    @ResponseBody
    public List<Object> courses(){
        List<HashMap<String, Object>> list=dao.courses();

        List<Object> array=new ArrayList<>();
        List<Object> arr=new ArrayList<>();
        arr.add("과목명");
        arr.add("평균점수");
        array.add(arr);

        for(HashMap<String, Object> map: list){
            arr=new ArrayList<>();
            arr.add(map.get("lname"));
            arr.add(map.get("avg(grade)"));
            array.add(arr);
        }
        return array;
    }

    @RequestMapping("/students")
    @ResponseBody
    public List<Object> students(){
        List<HashMap<String, Object>> list=dao.students();

        List<Object> array=new ArrayList<>();
        List<Object> arr=new ArrayList<>();
        arr.add("학생명");
        arr.add("평균점수");
        array.add(arr);

        for(HashMap<String, Object> map: list){
            arr=new ArrayList<>();
            arr.add(map.get("sname"));
            arr.add(map.get("avg(grade)"));
            array.add(arr);
        }
        return array;
    }

    @RequestMapping("/departments")
    @ResponseBody
    public List<Object> departments(){
        List<HashMap<String, Object>> list=dao.departments();

        List<Object> array=new ArrayList<>();
        List<Object> arr=new ArrayList<>();
        arr.add("학과명");
        arr.add("학생수");
        array.add(arr);

        for(HashMap<String, Object> map: list){
            arr=new ArrayList<>();
            arr.add(map.get("dept"));
            arr.add(map.get("count(*)"));
            array.add(arr);
        }
        return array;
    }
}
```

3) Chart 출력을 위한 chart.jsp 파일을 작성한다.

아래 구글 차트 홈페이지로 이동하여 문서(Guides)의 각종 차트들을 참조한다.

```
https://developers.google.com/chart
```



[src]-[main]-[webapp]-[WEB-INF]-[views] chart.jsp

```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

<button id="btnCourses">강좌별 평균점수</button>
<button id="btnStudents">학생별 평균점수</button>
<button id="btnDepartments">학과별 평균점수</button>

<div id="chart" style="width: 900px; height: 500px;"></div>

<script type="text/javascript">
    $("#btnCourses").on("click", function(){
        $.ajax({
            type: "get",
            url: "/chart/courses",
            success: function(data){ var title="강좌별 평균점수"; lineChart(data, title); }
        });
    });

    $("#btnStudents").on("click", function(){
        $.ajax({
            type: "get",
            url: "/chart/students",
            success: function(data){ var title="학생별 평균점수"; barChart(data, title); }
        });
    });

    $("#btnDepartments").on("click", function(){
        $.ajax({
            type: "get",
            url: "/chart/departments",
            success: function(data){ var title="학과별 학생수"; pieChart(data, title); }
        });
    });

    function barChart(chartData, chartTitle) {
        google.charts.load('current', {'packages':['bar']});
        google.charts.setOnLoadCallback(drawChart);
        function drawChart() {
            var data = google.visualization.arrayToDataTable(chartData);
            var options = {
                chart: { title: chartTitle },
                bars: 'horizontal'  //vertical 세로그래프
            };
            var chart = new google.charts.Bar(document.getElementById('chart'));
            chart.draw(data, google.charts.Bar.convertOptions(options));
        }
    }

    function pieChart(chartData, chartTitle) {
        google.charts.load('current', {'packages':['corechart']});
        google.charts.setOnLoadCallback(drawChart);
        function drawChart() {
            var data = google.visualization.arrayToDataTable(chartData);
            var options = { title: chartTitle };
            var chart = new google.visualization.PieChart(document.getElementById('chart'));
            chart.draw(data, options);
        }
    }

    function lineChart(chartData, chartTitle) {
        google.charts.load('current', {'packages':['line']});
        google.charts.setOnLoadCallback(drawChart);
        function drawChart() {
            var data = google.visualization.arrayToDataTable(chartData);
            var options = { title: chartTitle };
            var chart = new google.charts.Line(document.getElementById('chart'));
            chart.draw(data, google.charts.Line.convertOptions(options));
        }
    }
</script>
```

- **Mybatis에서 배열처리**

- collection : 전달받은 인자. List or Array 형태만 가능
- item : 전달받은 인자 값을 alias 명으로 대체
- open : 구문이 시작될 때 삽입할 문자열
- close : 구문이 종료될 때 삽입할 문자열
- separator : 반복 되는 사이에 출력할 문자열
- index : 반복되는 구문 번호이다. 0부터 순차적으로 증가

[src/main/resources]-[mapper] NoticeMapper.xml 방법1

```
<select id="list" resultType="hashmap">
   select * from tbl_notice
   where sender in
      <foreach collection="users" item="user" open="(" close=")" separator=",">
         #{user}
      </foreach>
 </select>
```

[src/main/resources]-[mapper] NoticeMapper.xml 방법2

```
<select id="list" resultType="hashmap">
   select * from tbl_notice
   where
      <foreach collection="users" item="user" separator="or">
         sender = #{user}
      </foreach>
 </select>
```

[src/main/java]-[com.example.mapper] NoticeDAOImpl.java

```
@Repository
public class NoticeDAOImpl implements NoticeDAO{
    @Autowired
    SqlSession session;
    String namespace="com.example.mapper.NoticeMapper";
    ...
    @Override
    public List<NoticeVO> listArray(ArrayList<String> arrUser) {
        HashMap<String, Object> map=new HashMap<>();
        map.put("users", arrUser);
        return session.selectList(namespace + ".listArray", map);
    }
}
```

[src/main/java]-[com.example.controller] NoticeController.java

```
@Controller
public class NoticeController {
    @Autowired
    NoticeDAO dao;
    ...

    @RequestMapping("/notices.json")
    public List<NoticeVO> listArray(){
        ArrayList<String> users=new ArrayList<>();
        users.add("user01");
        users.add("user02");
        return dao.listArray(users);
    }
}
```

- CK Editor

[src]-[main]-[webapp]-[WEB-INF]-[views]-editor.jsp

```
<head>
    <script src="https://cdn.ckeditor.com/4.16.2/standard/ckeditor.js"></script>
</head>
<body>
    <form name="frm" method="post" action="insert">
        <textarea id="editor" name="content" cols=80 rows=15></textarea>
        <input type="submit" value="저장"/>
        <input type="reset" value="취소"/>
    </form>
</body>
<script>
    var ckeditor_config = {
        resize_enable : false,
        enterMode : CKEDITOR.ENTER_BR,
        shiftEnterMode : CKEDITOR.ENTER_P,
        filebrowserUploadUrl : "/ckupload",
        height: 300
    };
    CKEDITOR.replace('editor', ckeditor_config)
</script>
```

[src/main/java]-[com.example.controller]-FileController.java

```
@Controller
public class FileController {
    @Resource(name = "uploadPath")
    String path;

    // 이미지파일 브라우저에 출력
    @RequestMapping("/display")
    @ResponseBody
    public ResponseEntity<byte[]> display(String fileName) throws Exception {
        ResponseEntity<byte[]> result = null;
        // display fileName이 있는 경우
        if (!fileName.equals("")) {
            File file = new File(path + fileName);
            HttpHeaders header = new HttpHeaders();
            header.add("Content-Type", Files.probeContentType(file.toPath()));
            result = new ResponseEntity<>(FileCopyUtils.copyToByteArray(file), header, HttpStatus.OK);
       }
       return result;
    }

    //ckEditor file upload
    @RequestMapping(value="/ckupload", method=RequestMethod.POST)
    @ResponseBody
    public HashMap<String,Object> ckUpload(MultipartHttpServletRequest multi) throws Exception {
        HashMap<String, Object> map=new HashMap<>();
        MultipartFile file=multi.getFile("upload");
        //업로드 할 파일이 비어있지 않은 경우
        if(!file.isEmpty()) {
            String fileName=file.getOriginalFileName();
            String exe=fileName.substring(fileName.lastIndexOf("."));
            String name = UUID.randomUUID().toString() + exe;
            file.transferTo(new File("c:/" + path + name));
            map.put("uploaded", 1);
            map.put("url", path + name);
        }
        return map;
    }
}
```

- Slick Slider 사용법

- Slick Slider 홈페이지

```
https://kenwheeler.github.io/slick/
```

[src]-[main]-[webapp]-[WEB-INF]-[views] slider.jsp

```
<!-- Slick 불러오기 -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.9.0/slick.min.js"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.9.0/slick.min.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.9.0/slick-theme.min.css">

<!-- stlye 은 slick 영역 확인용 -->
<div style="padding:50px 50px; background-color:skyblue">
    <div id="slider-div">
        <div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div><div>6</div>
    </div>
</div>

<script>
   $.ajax({
       type:"get",
       url:"/product/best.json",
       dataType:"json",
       success:function(data){
           var template = Handlebars.compile($("#temp").html());
           $("#slider-div").html(template(data));
           applySlider();
       }
    });

    function applySlider() {
        $('#slider-div').slick({
            slide : 'div',  //슬라이드 되어야 할 태그 ex) div, li
            infinite : true,  //무한 반복 옵션
            slidesToShow :3,  // 한 화면에 보여질 컨텐츠 개수
            slidesToScroll : 1,  //스크롤 한번에 움직일 컨텐츠 개수
            speed : 100,  // 다음 버튼 누르고 다음 화면 뜨는데까지 걸리는 시간(ms)
            arrows : true,  // 옆으로 이동하는 화살표 표시 여부
            dots : true,  // 스크롤바 아래 점으로 페이지네이션 여부
            autoplay : true,  // 자동 스크롤 사용 여부
            autoplaySpeed : 10000,  // 자동 스크롤 시 다음으로 넘어가는데 걸리는 시간 (ms)
            pauseOnHover : true,  // 슬라이드 이동    시 마우스 호버하면 슬라이더 멈추게 설정
            vertical : false,  // 세로 방향 슬라이드 옵션
            prevArrow : "<button type='button' class='slick-prev'>Previous</button>",  // 이전 화살표 모양 설정
            nextArrow : "<button type='button' class='slick-next'>Next</button>",  // 다음 화살표 모양 설정
            dotsClass : "slick-dots",  //아래 나오는 페이지네이션(점) css class 지정
            draggable : true,  //드래그 가능 여부

            // 반응형 웹구현 옵션
            responsive : [
                {
                    breakpoint : 960,  //화면 사이즈 960px
                    settings : { slidesToShow : 3 }  //위에 옵션이 디폴트 , 여기에 추가하면 그걸로 변경
                },
                {
                    breakpoint : 768,  //화면 사이즈 768px
                    settings : { slidesToShow : 2 }  //위에 옵션이 디폴트 , 여기에 추가하면 그걸로 변경
                }
            ]
        });
    }
</script>
```