

## 5장. 화면 개발 - thymeleaf



*thymeleaf*



Spring Boot

- 웹 애플리케이션 화면 개발

스프링 부트는 템플릿 엔진을 이용한 화면 처리를 지원한다.

스프링 부트가 지원하는 템플릿 엔진은 타임리프(Thymeleaf)를 비롯하여 Freemaker, Mustache, Groovy Templates이 있다.

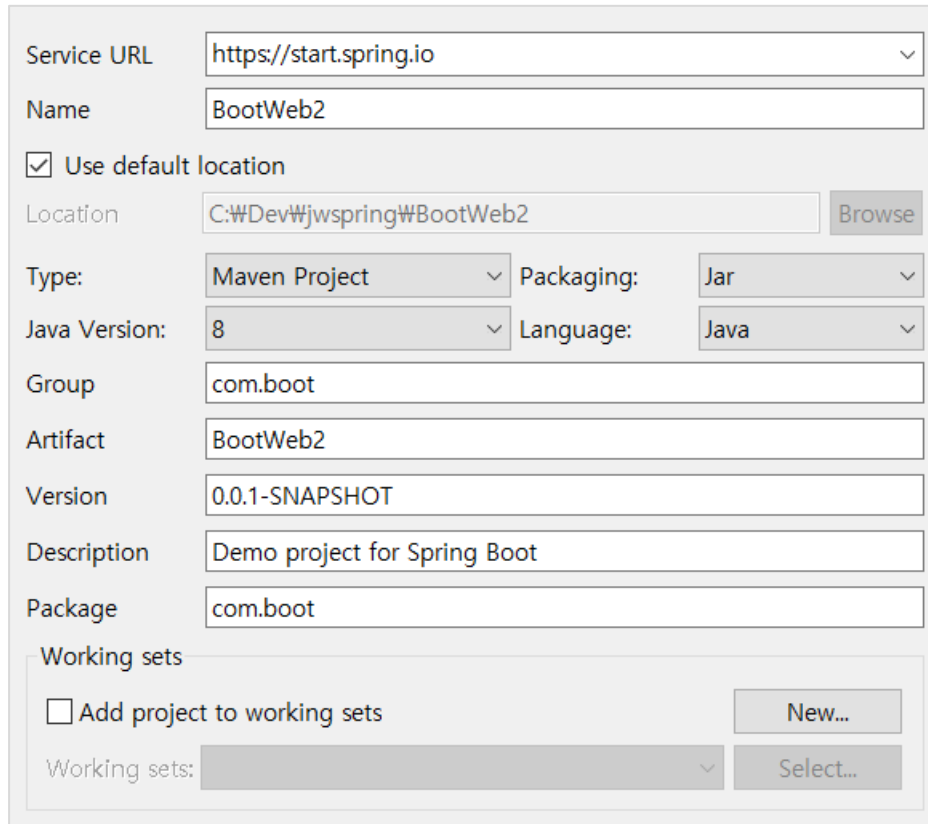
템플릿 엔진을 이용하면 데이터와 거의 완벽하게 분리된 화면을 개발 할 수 있기 때문에 순수하게 HTML 파일 만을 이용한 화면 개발이 가능하고, 운영 과정에서 쉽게 화면을 변경할 수도 있다.

최종적으로 스프링 부트가 제공하는 템플릿 엔진 중에서 타임리프를 적용하는데, 먼저 JSP 기반의 화면을 만들고 타임리프를 적용한다.

JSP는 자바의 표준이며, JSP로 여전히 화면을 제공하는 시스템들이 존재하기 때문이다.



## ▪ BootWeb2 프로젝트 생성 및 환경 설정



The screenshot shows the 'New Project' dialog for a Spring Boot application. The 'Service URL' is set to 'https://start.spring.io'. The 'Name' is 'BootWeb2'. The 'Use default location' checkbox is checked, and the 'Location' is 'C:\Dev\jwspring\BootWeb2'. The 'Type' is 'Maven Project', 'Packaging' is 'Jar', 'Java Version' is '8', and 'Language' is 'Java'. The 'Group' is 'com.boot', 'Artifact' is 'BootWeb2', 'Version' is '0.0.1-SNAPSHOT', 'Description' is 'Demo project for Spring Boot', and 'Package' is 'com.boot'. The 'Working sets' section has 'Add project to working sets' unchecked, with 'New...' and 'Select...' buttons.

Service URL	https://start.spring.io
Name	BootWeb2
<input checked="" type="checkbox"/> Use default location	
Location	C:\Dev\jwspring\BootWeb2 <span>Browse</span>
Type:	Maven Project
Packaging:	Jar
Java Version:	8
Language:	Java
Group	com.boot
Artifact	BootWeb2
Version	0.0.1-SNAPSHOT
Description	Demo project for Spring Boot
Package	com.boot
Working sets	
<input type="checkbox"/> Add project to working sets	<span>New...</span>
Working sets:	<span>Select...</span>

Packing – jar

Java Version: 8

- 모듈 추가

Lombok, Spring Web, DevTools

Spring Data JPA, H2 Database

Thymeleaf



- BootWeb2 main 실행 전 H2 DB 연결

English ▼ 설정 도구 도움말

로그인

저장한 설정: Generic H2 (Server) ▼

설정 이름: Generic H2 (Server) 저장 삭제

드라이버 클래스: org.h2.Driver

JDBC URL: jdbc:h2:tcp://localhost/~/test

사용자명: sa

비밀번호:

연결 연결 시험

```
@SpringBootApplication
public class BootWeb2Application {

    public static void main(String[] args) {
        SpringApplication.run(BootWeb2Application.class, args);
    }
}
```



## ▪ BootWeb2 프로젝트 생성 및 환경 설정

```
# WebApplication Type Setting
#spring.main.web-application-type=none
spring.main.web-application-type=servlet

## Server Setting(포트번호 변경)
server.port = 8181

# DataSource Setting
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.url=jdbc:h2:tcp://localhost/~ /test
spring.datasource.username=sa
spring.datasource.password=

# JPA Setting
spring.jpa.hibernate.ddl-auto=update
spring.jpa.generate-ddl=false
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.properties.hibernate.format_sql=true

# Thymeleaf Cache Setting
spring.thymeleaf.cache=false

# static resource
spring.mvc.static-path-pattern=/static/**
```



- JPA와 DB 연동하기

- (1) 엔티티 클래스로 변환 - Board.java

```
@Getter
@Setter
@Entity
public class Board {
    @Id @GeneratedValue
    private Long seq;

    private String title;

    @Column(updatable=false)
    private String writer;
    private String content;

    @Column(insertable=false, updatable=false,
            columnDefinition = "timestamp default current_timestamp")
    private Date createDate;

    @Column(insertable=false, updatable=false,
            columnDefinition = "bigint default 0")
    private Long cnt;
}
```



## ▪ JPA와 DB 연동하기

### (2) 테이블 생성 설정

```
# JPA Setting
spring.jpa.hibernate.ddl-auto=create
spring.jpa.generate-ddl=false
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.properties.hibernate.format_sql=true
```

### (3) 리포지터리 인터페이스

```
package com.boot.persistence;

import org.springframework.data.repository.CrudRepository;

public interface BoardRepository extends CrudRepository<Board, Long>{

}
```



- 게시글 등록 테스트 – Junit

```
@Slf4j
@SpringBootTest
public class BoardRepositoryTest {

    @Autowired
    private BoardRepository boardRepo;

    //글 등록
    @Test
    public void testInsertBoard() {
        Board board = new Board();
        board.setTitle("두 번째 게시글");
        board.setWriter("테스터");
        board.setContent("등록이 잘 되네요..");
        board.setCreateDate(new Date());
        board.setCnt(0L);

        boardRepo.save(board); //save() 메서드로 DB에 저장함
    }
}
```





- 게시글 등록 테스트

```
SELECT * FROM BOARD;
```

SEQ	CNT	CONTENT	CREATE_DATE	TITLE	WRITER
1	0	등록이 잘 되네요..	2022-09-22 20:20:22.125234	첫 번째 게시글	테스터

(1 row, 1 ms)



ddl-auto 설정을 update로 변경함

```
# JPA Setting
spring.jpa.hibernate.ddl-auto=update
spring.jpa.generate-ddl=false
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.properties.hibernate.format_sql=true
```



- 비즈니스 컴포넌트 만들기

(1) Service 인터페이스

```
public interface BoardService {  
  
    List<Board> getBoardList();  
  
    void insertBoard(Board board);  
  
    Board getBoard(long seq);  
  
    void updateBoard(Board board);  
  
    void deleteBoard(Board board);  
}
```



- 비즈니스 컴포넌트 만들기

(2) Service 구현 클래스

```
@Service
public class BoardServiceImpl implements BoardService {

    @Autowired
    private BoardRepository boardRepo;

    @Override
    public List<Board> getBoardList() {
        return (List<Board>) boardRepo.findAll();
    }

    @Override
    public void insertBoard(Board board) {
        boardRepo.save(board);
    }
}
```



- 비즈니스 컴포넌트 만들기

(2) Service 구현 클래스

```
@Override
public Board getBoard(long seq) {
    return boardRepo.findById(seq).get();
}

@Override
public void updateBoard(Board board) {
    Board findBoard = boardRepo.findById(board.getSeq()).get();
    findBoard.setTitle(board.getTitle());
    findBoard.setContent(board.getContent());
    boardRepo.save(findBoard);
}

@Override
public void deleteBoard(Board board) {
    boardRepo.delete(board);
}
```



- 비즈니스 컴포넌트 사용하기 - BoardController

```
@Controller
public class BoardController {

    @Autowired
    private BoardService service;

    //게시글 목록
    @GetMapping("/getBoardList")
    public String getBoardList(Model model) {
        List<Board> boardList = service.getBoardList();

        model.addAttribute("boardList", boardList);

        return "getBoardList";
    }

    //게시글 등록 폼 요청
    @GetMapping("/insertBoard")
    public String insertBoard() {
        return "insertBoard";
    }
}
```



- 비즈니스 컴포넌트 사용하기 - BoardController

```
//게시글 등록 처리
@PostMapping("/insertBoard")
public String insertBoard(Board board) {
    service.insertBoard(board);
    return "redirect:getBoardList";
}

//게시글 상세 조회
@GetMapping("/getBoard")
public String getBoard(Long seq, Model model) {
    Board board = service.getBoard(seq);
    model.addAttribute(board);
    return "getBoard";
}
```



- 비즈니스 컴포넌트 사용하기 - BoardController

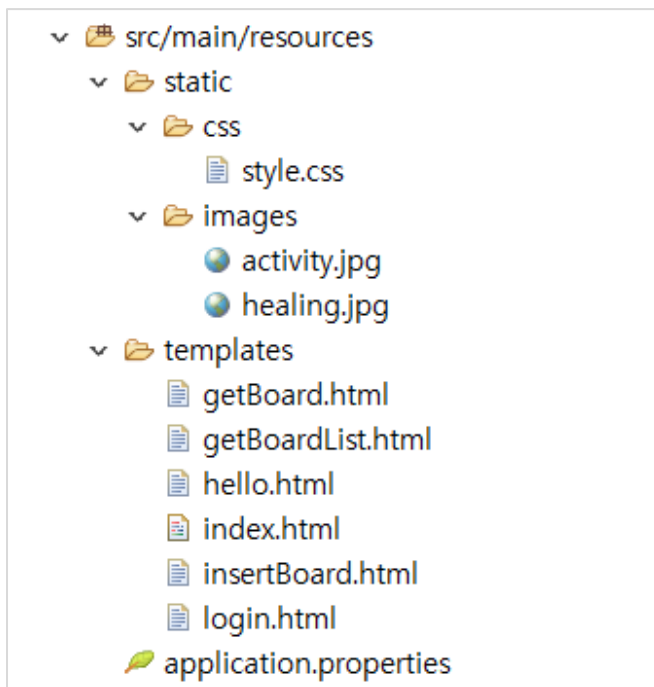
```
//게시글 삭제
@GetMapping("/deleteBoard")
public String deleteBoard(Board board) {
    service.deleteBoard(board);
    return "redirect:getBoardList";
}

//게시글 수정
@PostMapping("/updateBoard")
public String updateBoard(Board board) {
    service.updateBoard(board);
    return "redirect:getBoardList";
}
```





- 템플릿, CSS 계층 구조도



- 게시글 목록 화면

## 게시글 목록

번호	제목	작성자	등록일	조회수
1	가을	한가을	2022/09/18 07:22:25	0

[새글 등록](#)



- 게시글 목록 – getBoardList.html

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8">
<title>Hello~</title>
<link rel="stylesheet" href="/static/css/style.css">
</head>
<body>
  <div id="container">
    <h2>게시글 목록</h2>
    <table class="tbl_list">
      <thead>
        <tr>
          <th>번호</th><th>제목</th><th>작성자</th><th>등록일</th><th>조회수</th>
        </tr>
      </thead>
    </table>
  </div>
</body>
</html>
```



## ▪ 게시글 목록 – getBoardList.html

JSTL의 <c:forEach> 반복 처리 기능 – th:each 속성 사용

<c:out> 출력기능 – th:text 속성 사용

날짜 포맷 변경 - #dates.format()

```
<tbody>
  <tr th:each="board : ${boardList}">
    <td th:text="${board.seq}"></td>
    <td><a th:href="@{/getBoard(seq=${board.seq})}"
          th:text="${board.title}"></a></td>
    <td th:text="${board.writer}"></td>
    <td th:text="${#dates.format(board.createDate, 'yyyy-MM-dd hh:mm:ss')}"></td>
    <td th:text="${board.cnt}"></td>
  </tr>
</tbody>
</table>
<p><a th:href="@{/insertBoard}">새글 등록</a> </p>
```



- CSS 스타일 – style.css

```
@charset "UTF-8";

/* common 스타일 */
#container{width: 1000px; margin: 20px auto; text-align: center}
a{text-decoration: none;}
h2{padding-top: 20px;}

/* getBoardList 스타일 */
.tbl_list{width: 700px; margin: 0 auto; text-align: center;}
.tbl_list, th, td{border: 1px solid #ccc; border-collapse: collapse;}
.tbl_list th, td{padding: 10px;}
.tbl_list thead{background: #eee;}

/* insertBoard 스타일 */
.tbl_reg{width: 700px; margin: 0 auto;}
.tbl_reg, td{border: 1px solid #ccc; border-collapse: collapse;}
.tbl_reg td{padding: 10px;}
.tbl_reg input{width: 580px; height: 25px;}
.tbl_reg textarea{width: 580px;}
```



- CSS 스타일 – style.css

```
/* getBoard 스타일*/
.tbl_view{width: 700px; margin: 0 auto;}
.tbl_view, td{border: 1px solid #ccc; border-collapse: collapse;}
.tbl_view td{padding: 10px;}
.tbl_view input{width: 580px; height: 25px;}
.tbl_view textarea{width: 580px;}

/* login 스타일*/
.tbl_login{width: 380px; margin: 0 auto;}
.tbl_login, td{border: 1px solid #ccc; border-collapse: collapse;}
.tbl_login td{padding: 10px;}
.tbl_login input{width: 250px; height: 30px}
```



## ▪ 게시물 쓰기

글쓰기

제목	<input type="text" value="가을"/>
작성자	<input type="text" value="한가을"/>
내용	<div><div>가을이 오고 있어요.. 바람이 서늘해요.</div><div></div></div>
<div>새글 등록</div>	

## ▪ 게시물 쓰기 – insertBoard.html

```
<div id="container">
  <h2>글쓰기</h2>
  <form th:action="@{insertBoard}" method="post">
    <table class="tbl_reg">
      <tr>
        <td>제목</td>
        <td><input type="text" name="title"></td>
      </tr>
      <tr>
        <td>작성자</td>
        <td><input type="text" name="writer"></td>
      </tr>
      <tr>
        <td>내용</td>
        <td>
          <textarea name="content" rows="10" cols="50"></textarea>
        </td>
      </tr>
      <tr>
        <td colspan="2" align="center">
          <button type="submit">새글 등록</button>
        </td>
      </tr>
    </table>
  </form>
</div>
```





## ▪ 게시물 상세 조회

글 상세 보기	
제목	<input type="text" value="가을"/>
작성자	<input type="text" value="한가을"/>
내용	<div><div>가을</div><div></div></div>
작성일	2022/09/18 07:44:18
조회수	<input type="text" value="0"/>
<div><div>글수정</div><div>글삭제</div><div>글목록</div></div>	



## ▪ 게시물 수정 및 삭제

localhost:8181 내용:  
정말로 삭제하시겠습니까?

확인취소

제목	<input type="text"/>
작성자	<input type="text" value="장그래"/>
내용	<div>스프링 부트</div>
작성일	2022/09/18 08:17:30
조회수	<input type="text" value="0"/>
<div>글수정글삭제글목록</div>	

- 게시글 상세 조회 - getBoard.html

```
<div id="container">
  <h2>글 상세 보기</h2>
  <form th:action="@{updateBoard}" method="post">
    <!-- 글 수정시에 반드시 기본키를 컨트롤러에 보내야 함 -->
    <input type="hidden" name="seq" th:value="${board.seq}">
    <table class="tbl_view">
      <tr>
        <td>제목</td>
        <td><input type="text" name="title" th:value="${board.title}"></td>
      </tr>
      <tr>
        <td>작성자</td>
        <td th:text="${board.writer}"></td>
      </tr>
      <tr>
        <td>내용</td>
        <td>
          <textarea name="content" rows="10" cols="50"
            th:text="${board.content}"></textarea>
        </td>
      </tr>
    </table>
  </form>
</div>
```



- 게시글 상세 조회 - getBoard.html

```
<tr>
  <td>작성일</td>
  <td th:text="${#dates.format(board.createDate, 'yyyy-MM-dd HH:mm:ss')}">
</tr>
<tr>
  <td>조회수</td>
  <td th:text=="${board.cnt}"></td>
</tr>
<tr>
  <td colspan="2" align="center">
    <button type="submit">글수정</button>
    <a th:href="@{/deleteBoard(seq=${board.seq})}"
      onclick="return confirm('정말로 삭제하시겠습니까?')">
      <button type="button">글삭제</button>
    </a>
    <a th:href="@{/getBoardList}">
      <button type="button">글목록</button>
    </a>
  </td>
</tr>
</table>
</form>
```



- 인덱스 페이지

게시판 프로그램입니다.



[글 목록](#) [로그인](#)



- 로그인 화면 – login.html

```
<div id="container">
  <h2>로그인</h2>
  <form th:action="@{login}" method="post">
    <table class="tbl_login">
      <tr>
        <td>아이디</td>
        <td><input type="text" name="id"></td>
      </tr>
      <tr>
        <td>비밀번호</td>
        <td><input type="password" name="password"></td>
      </tr>
      <tr>
        <td colspan="2" align="center">
          <button type="submit">로그인</button>
        </td>
      </tr>
    </table>
  </form>
</div>
```



- 로그인 컨트롤러 – LoginController

```
@Controller
public class LoginController {

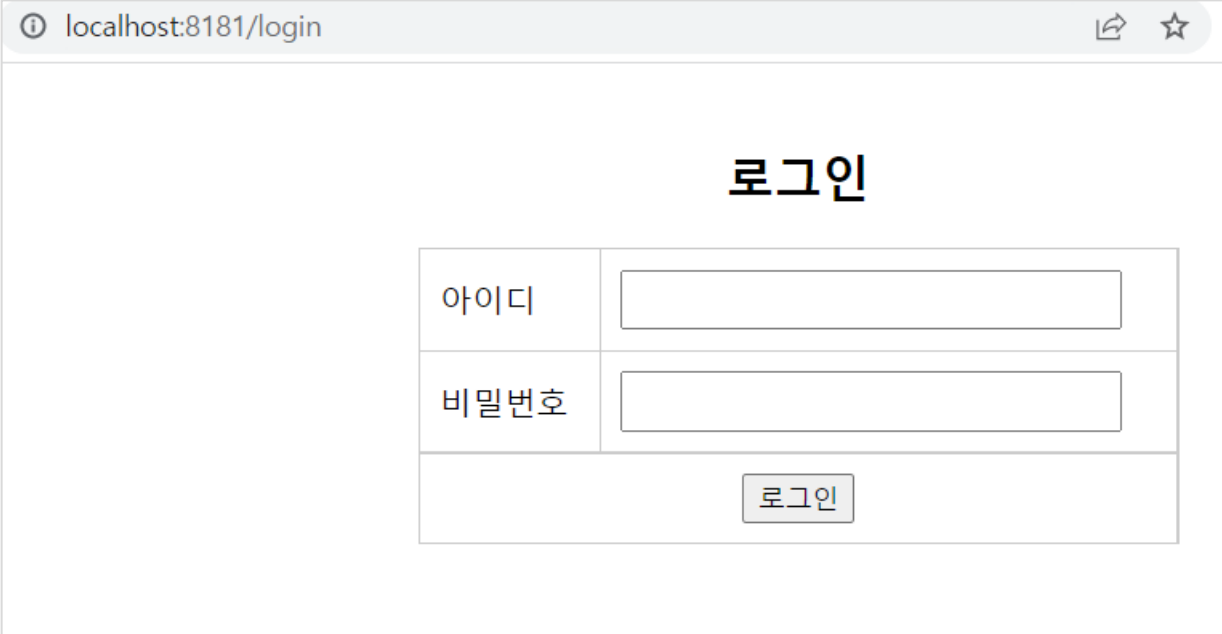
    @GetMapping("/login")
    public void loginView() {

    }
}
```





- 로그인 화면



The screenshot shows a web browser window with the address bar displaying 'localhost:8181/login'. The page content is centered and features the title '로그인' (Login) in a large, bold font. Below the title is a form with two input fields: '아이디' (ID) and '비밀번호' (Password). A '로그인' (Login) button is positioned below the password field.

아이디	<input type="text"/>
비밀번호	<input type="password"/>
<input type="button" value="로그인"/>	

# 로그인 컴포넌트 개발

- 엔티티 클래스 – Member.java

```
@Getter
@Setter
@ToString
@Entity
public class Member {

    @Id
    private String id;

    private String password;
    private String name;
    private String role;
}
```



# 로그인 컴포넌트 개발

- 리포지터리 – MemberRepository.java

```
import com.boot.domain.Member;  
  
public interface MemberRepository extends JpaRepository<Member, String>{  
  
}
```



# 로그인 컴포넌트 개발

- 테스트 데이터 초기화

```
@SpringBootTest
public class DataInitializeTest {

    @Autowired
    private MemberRepository memberRepo;

    @Autowired
    private BoardRepository boardRepo;

    @Test
    public void testDataInsert() {
        Member member1 = new Member();
        member1.setId("member1");
        member1.setPassword("member111");
        member1.setName("뽀로로");
        member1.setRole("ROLE_USER");
        memberRepo.save(member1);

        Member member2 = new Member();
        member2.setId("member2");
        member2.setPassword("member222");
        member2.setName("아기상어");
        member2.setRole("ROLE_ADMIN");
        memberRepo.save(member2);
    }
}
```



# 로그인 컴포넌트 개발

- 테스트 데이터 초기화

```
for(int i=1; i<=3; i++) {  
    Board board = new Board();  
    board.setWriter("뽀로로");  
    board.setTitle("뽀로로가 등록한 게시글 " + i);  
    board.setContent("뽀로로가 등록한 게시글 내용 " + i);  
    boardRepo.save(board);  
}  
  
for(int i=1; i<=3; i++) {  
    Board board = new Board();  
    board.setWriter("아기상어");  
    board.setTitle("아기상어가 등록한 게시글 " + i);  
    board.setContent("아기상어가 등록한 게시글 내용 " + i);  
    boardRepo.save(board);  
}  
}
```



# 로그인 컴포넌트 개발

- 테스트 데이터 초기화

SELECT \* FROM Member;

ID	NAME	PASSWORD	ROLE
member1	뽀로로	member111	ROLE_USER
member2	아기상어	member222	ROLE_ADMIN

(2 행, 2 ms)

SELECT \* FROM BOARD;

SEQ	CNT	CONTENT	CREATE_DATE	TITLE	WRITER
1	0	뽀로로가 등록한 게시글 내용 1	2022-09-23 05:56:51.107493	뽀로로가 등록한 게시글 1	뽀로로
2	0	뽀로로가 등록한 게시글 내용 2	2022-09-23 05:56:51.110576	뽀로로가 등록한 게시글 2	뽀로로
3	0	뽀로로가 등록한 게시글 내용 3	2022-09-23 05:56:51.113476	뽀로로가 등록한 게시글 3	뽀로로
4	0	아기상어가 등록한 게시글 내용 1	2022-09-23 05:56:51.11677	아기상어가 등록한 게시글 1	아기상어
5	0	아기상어가 등록한 게시글 내용 2	2022-09-23 05:56:51.118462	아기상어가 등록한 게시글 2	아기상어
6	0	아기상어가 등록한 게시글 내용 3	2022-09-23 05:56:51.121455	아기상어가 등록한 게시글 3	아기상어

(6 rows, 2 ms)



# 로그인 컴포넌트 개발

- Service 인터페이스와 구현 클래스

```
public interface MemberService {  
  
    Member getMember(Member member); //로그인  
}
```

```
@Service  
public class MemberServiceImpl implements MemberService{  
  
    @Autowired  
    private MemberRepository memberRepo;  
  
    @Override  
    public Member getMember(Member member) {  
        //findById의 반환형이 Optional<T> 임  
        //Optional은 주로 null 에러를 방지하기 위한 클래스임  
        Optional<Member> findMember = memberRepo.findById(member.getId());  
        if(findMember.isPresent()) {  
            return findMember.get();  
        }else {  
            return null;  
        }  
    }  
}
```



# 로그인 컴포넌트 개발

## ▪ Controller에서 Service 사용하기

```
//Model 객체 - 'member'를 자동으로 세션에 저장함
@SessionAttributes("member")
@Controller
public class LoginController {

    @Autowired
    private MemberService service;

    //로그인 폼 페이지 요청
    @GetMapping("/login")
    public void loginView() {
    }

    //로그인 인증 처리
    @PostMapping("/login")
    public String login(Member member, Model model){
        Member findMember = service.getMember(member);
        if(findMember != null
            && findMember.getPassword().equals(member.getPassword())) {
            model.addAttribute("member", findMember); //세션 발급
            return "redirect:getBoardList";
        }else {
            return "redirect:login";
        }
    }
}
```





# 로그인 컴포넌트 개발

- 로그인 상태 정보 이용 – getBoardList.html

SpringEL - `${session['member'].name}`

```
<div id="container">
  <h2>게시글 목록</h2>
  <h3><font color="red" th:text="${session['member'].name}"></font>님
    게시판 입장을 환영합니다.</h3>
  <p><a th:href="@{/logout}">LOG_OUT</a></p>
  <table class="tbl_list">
```

## 게시글 목록

뽀로로님 게시판 입장을 환영합니다.

LOG\_OUT

번호	제목	작성자	등록일	조회수
1	뽀로로가 등록한 게시글 1	뽀로로	2022-09-23 06:16:36	0
2	뽀로로가 등록한 게시글 2	뽀로로	2022-09-23 06:16:36	0
3	뽀로로가 등록한 게시글 3	뽀로로	2022-09-23 06:16:36	0



# 로그인 컴포넌트 개발

- 관리자(ROLE\_ADMIN) 만 글 삭제 하기 – getBoard.html

```
<td colspan="2" align="center">
  <button type="submit">글수정</button>
  <a th:if="${session['member'].role} == 'ROLE_ADMIN'"
    th:href="@{/deleteBoard(seq=${board.seq})}"
    onclick="return confirm('정말로 삭제하시겠습니까?')">
    <button type="button">글삭제</button>
  </a>
  <a th:href="@{/getBoardList}">
    <button type="button">글목록</button>
  </a>
</td>
```

SpringEL 사용

**`${session['member'].role} == 'ROLE_ADMIN'`**



# 로그인 컴포넌트 개발

- 인증 상태 유지하기- 로그인한 작성자 보이기

```
<tr>
  <td>제목</td>
  <td><input type="text" name="title"></td>
</tr>
<tr>
  <td>작성자</td>
  <td><input type="text" name="writer" th:value="${session['member'].name}"></td>
</tr>
```

글쓰기	
제목	<input type="text"/>
작성자	뽀로로
내용	<input type="text"/>



# 로그인 컴포넌트 개발

- 인증 상태 유지하기- 로그인 한 사용자만 글쓰기 가능. BoardController

```
@SessionAttributes("member")
@Controller
public class BoardController {
```

```
    //인증 상태 유지하기
    @ModelAttribute("member")
    public Member setMember() {
        return new Member();
    }

    //게시글 등록 폼 요청
    @GetMapping("/insertBoard")
    public String insertBoard(@ModelAttribute("member") Member member) {
        //로그인 하지 않은 경우 - 로그인 페이지로 이동
        if(member.getId() == null) {
            return "redirect:login";
        }
        return "insertBoard";
    }
}
```



# 로그인 컴포넌트 개발

- 인증 상태 유지하기- 로그인 한 사용자만 글쓰기 가능. BoardController

👉 클래스 선언부에 @SessionAttributes("member")를 추가했기 때문에 자동으로 세션에 등록된다. 그리고 setMember() 메소드를 추가해서 Model 객체를 리턴했는데, 이때 Member 객체가 가장 먼저 세션에 등록된다.

또한 insertBoard()의 매개 변수로 @ModelAttribute("member")을 사용해서 세션에 등록된 "member"라는 이름의 객체를 member 변수에 바인딩 했다.

그러면 insertBoard() 안에서 member 변수에 바인딩된 객체의 아이디 존재 여부를 통해 로그인 여부를 판단할 수 있다.



# 로그인 컴포넌트 개발

- 로그아웃 처리

```
@GetMapping("/logout")  
public String logout(HttpSession session) {  
    session.invalidate();  
    return "redirect: "; //경로가 공백이면 '/' 경로와 같음  
}
```

