Spring Boot Application

스프링 부트 응용

Thymeleaf를 이용한 화면 개발

한국기술교육대학교
온라인평생교육원

학습내용

- Thymeleaf 개요
- Thymeleaf를 이용한 화면 개발

학습목표

- Thymeleaf의 개요에 대해 설명할 수 있다.
- JSP로 만든 웹 애플리케이션 화면을 스프링 부트의 Thymeleaf로 변환할 수 있다.

Thymeleaf 개요

1 스프링 부트와 Thymeleaf

1 템플릿 엔진이란?

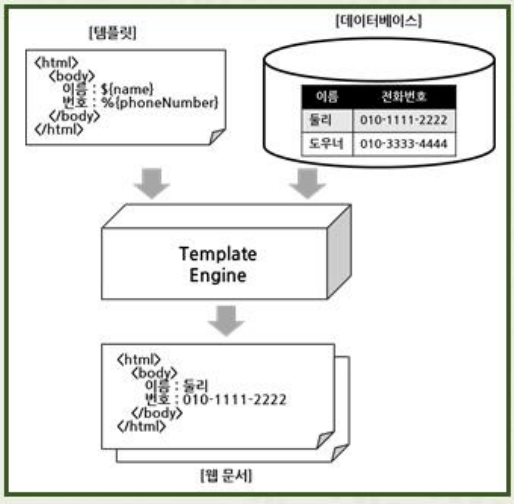
- 사용자에게 제공되는 화면과 데이터 처리 로직을 분리시켜서 개발 및 유지보수의 편의성을 증대 시킴
- EL과 JSTL이 결합된 JSP도 템플릿 엔진에 포함됨

1 스프링 부트와 Thymeleaf

1 템플릿 엔진이란?

스프링 부트가 지원하는 템플릿 엔진

- Thymeleaf
- Apache Freemarker
- Mustache
- Groovy Templates



Thymeleaf 개요

1 스프링 부트와 Thymeleaf

2 Thymeleaf 설정

- 스프링 부트 프로젝트에서 Thymeleaf를 적용하기 위해서는 pom.xml에 Thymeleaf 스타터를 추가해야 함

1 스프링 부트와 Thymeleaf

2 Thymeleaf 설정

- pom.xml 파일에 의존성이 추가된 것을 확인한 후에는 반드시 애플리케이션을 다시 실행함

```
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
</dependencies>
```

Thymeleaf 개요

2 Thymeleaf 기본

1 컨트롤러 작성

- BoardController에 hello() 메소드를 작성함

```
@GetMapping("/hello")
public void hello(Model model) {
    model.addAttribute("greeting", "HelloWorld!");
}
```

2 Thymeleaf 기본

1 컨트롤러 작성

- hello 메소드의 리턴 타입이 void이므로 요청 path("/hello")에 해당하는 hello.html 화면을 응답 화면으로 사용함

Thymeleaf 개요

2 Thymeleaf 기본

① 컨트롤러 작성

- 가독성을 위해 다음과 같이 뷰 이름을 리턴할 수 있음

```
@GetMapping("/hello")
public String hello(Model model) {
    model.addAttribute("greeting", "HelloWorld!");
    return "hello";
}
```

2 Thymeleaf 기본

② 템플릿 파일 작성

- Thymeleaf가 인지하는 템플릿 파일의 기본 위치
  - src/main/resources의 templates 폴더

Thymeleaf 개요

2 Thymeleaf 기본

⟨2⟩ 템플릿 파일 작성

- templates 폴더에 hello.html인 템플릿 파일을 작성함

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>타임리프 테스트</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body th:align="center">
<h1 th:text="${greeting}"></h1>
</body>
</html>
```

Thymeleaf 네임스페이스

Model에 "greeting"이라는 이름으로 등록된 메시지 출력

2 Thymeleaf 기본

⟨2⟩ 템플릿 파일 작성

- Thymeleaf 네임스페이스 선언으로 인해 HTML 태그에서 Thymeleaf가 제공하는 속성들을 사용할 수 있음
- th:text 속성은 화면에 문자열이나 특정 값을 출력할 때 사용함
- HTML 템플릿에서 사용할 데이터는 컨트롤러에서 설정함

Thymeleaf 개요

2 Thymeleaf 기본

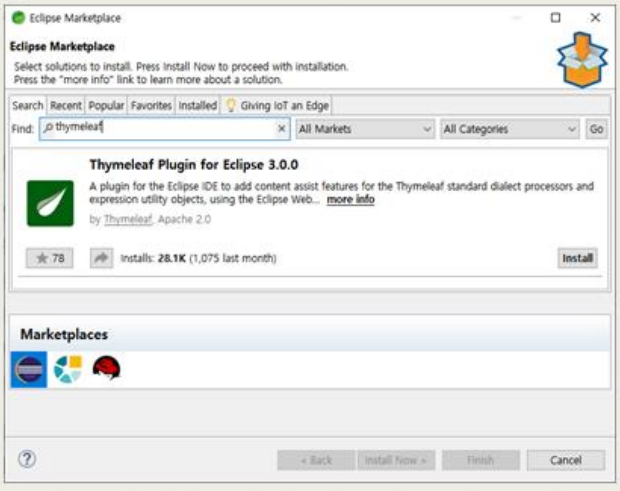
3 Thymeleaf 플러그인

- Thymeleaf가 제공하는 속성을 자동 완성(Ctrl + Space) 하려면 Thymeleaf 플러그인을 설치해야 함

2 Thymeleaf 기본

3 Thymeleaf 플러그인

- Eclipse의 마켓플레이스에서 Thymeleaf 플러그인을 검색한 결과



Thymeleaf 개요

3 Thymeleaf 문법

1 Thymeleaf 속성

- Thymeleaf를 적용하기 위해서는 HTML 태그 내에서 'th:' 접두사로 시작하는 속성을 사용하면 됨

3 Thymeleaf 문법

1 Thymeleaf 속성

- 자주 사용하는 Thymeleaf 속성과 사용 예

속성	의미	사용 예
th:text	단순 텍스트 출력	 greeting 값이 "환영합니다!"라면 환영합니다!
th:with	동적 변수 값 출력	<div th:with="address=${name} + '의 주소는 쌍문동'" th:text={address}> name 값이 "둘리"라면 둘리의 주소는 쌍문동

Thymeleaf 개요

3 Thymeleaf 문법

1 Thymeleaf 속성

- 자주 사용하는 Thymeleaf 속성과 사용 예

속성	의미	사용 예
th:if th:unless	조건문(if, else)	관리자 <span th:unless="${role} == 'Admin'">사용자 role 값이 "Admin"이면 관리자 나머지는 모두 사용자

3 Thymeleaf 문법

1 Thymeleaf 속성

- 자주 사용하는 Thymeleaf 속성과 사용 예

속성	의미	사용 예
th:switch th:case	다중 분기문 (switch~case)	 제목 내용 제목 + 내용 searchCondition 값이 "T"라면 제목 searchCondition 값이 "C"라면 내용 searchCondition 값이 "TC"라면 제목 + 내용

Thymeleaf 개요

3 Thymeleaf 문법

1 Thymeleaf 속성

- 자주 사용하는 Thymeleaf 속성과 사용 예

속성	의미	사용 예
th:each	반복문(for)	<tr th:each="board : ${boardList}"> 반복할 내용 </tr>

		boardList 컬렉션에 3개의 게시글 정보가 들어 있다면
		<tr>
		  반복할 내용
		  반복할 내용
		  반복할 내용
		</tr>

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

⟨1⟩ 글목록 기능 구현

1 HTML 파일 작성

- 게시글 목록 화면을 출력하기 위해서 templates 폴더에 board 폴더를 생성하고 getBoardList.html 파일을 작성함

1 Thymeleaf 기반의 게시판

⟨1⟩ 글목록 기능 구현

1 HTML 파일 작성

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>게시 글 목록</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body th:align="center">
<h1>게시글 목록(Thymeleaf)</h1>
<table th:align="center" border="1" th:cellpadding="0" th:cellspacing="0" th:width="700">
<tr>
    <th bgcolor="orange" width="100">번호</th>
    <th bgcolor="orange" width="200">제목</th>
    <th bgcolor="orange" width="150">작성자</th>
    <th bgcolor="orange" width="150">등록일</th>
    <th bgcolor="orange" width="100">조회수</th>
</tr>
<!-- 게시글 목록 컬렉션에 대한 반복 처리 -->
</table>
<br>
</body>
</html>
```

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

2 프로젝트 생성

1 글목록 데이터 출력

- Thymeleaf의 th:each 속성을 사용하면 배열이나 List, Map 같은 컬렉션에 저장된 데이터를 반복 처리할 수 있음

```
<tr th:each="board : ${boardList}">
    <td th:text="${board.seq}">
    <td th:text="${board.title}">
    <td th:text="${board.writer}">
    <td th:text="${board.createDate}">
    <td th:text="${board.cnt}">
</tr>
```

1 Thymeleaf 기반의 게시판

2 프로젝트 생성

1 글목록 데이터 출력

- th:each 속성은 기존의 JSTL의 <c:forEach> 태그와 결과가 동일함

```
<c:forEach var="board" items="${boardList }">
<tr>
<td>${board.seq }</td>
<td>${board.title }</td>
<td>${board.writer }</td>
<td>${board.createDate }</td>
<td>${board.cnt }</td>
</tr>
</c:forEach>
```

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

〈2〉 프로젝트 생성

2 상태 변수와 날짜 포맷 지정

- th:each 속성에서는 현재 컬렉션의 상태 정보를 저장하는 상태 변수를 선언할 수 있음

1 Thymeleaf 기반의 게시판

〈2〉 프로젝트 생성

2 상태 변수와 날짜 포맷 지정

- state라는 상태 변수를 추가하여 게시글의 일련번호(seq) 대신에 출력되는 순서대로 번호를 부여할 수 있음

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

〈2〉 프로젝트 생성

2 상태 변수와 날짜 포맷 지정

- Thymeleaf는 다양한 객체들을 지원함
- 이 중에서 dates는 java.util.Date 타입의 객체를 처리할 때 사용함

1 Thymeleaf 기반의 게시판

〈2〉 프로젝트 생성

2 상태 변수와 날짜 포맷 지정

- dates 객체의 format 함수를 이용하면 날짜 데이터에 대한 형식을 지정할 수 있음

```
<tr th:each="board, state : ${boardList}">
    <td th:text="${state.count}">
    <td><a th:href="@{getBoard(seq=${board.seq})}" th:text="${board.title}"></a></td>
    <td th:text="${board.writer}">
    <td th:text="${#dates.format(board.createDate, 'yyyy-MM-dd')}">
    <td th:text="${board.cnt}">
</tr>
```

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

〈2〉 프로젝트 생성

2 상태 변수와 날짜 포맷 지정

- Thymeleaf를 적용하는 순간 더 이상 JSP에서 사용하던 ViewResolver가 사용되지 않음
- 때문에 BoardController.getBoardList 메소드가 리턴하는 화면 정보를 수정해야 함

```
@RequestMapping("/board/getBoardList")
public String getBoardList(Model model) {
    model.addAttribute("boardList", boardService.getBoardList());
    return "board/getBoardList";
}
```

1 Thymeleaf 기반의 게시판

〈3〉 글상세 기능 구현

1 글상세 링크 및 파라미터 설정

- 글목록에서 제목 링크를 클릭했을 때 글상세 화면으로 이동하기 위해서 getBoardList.html 파일에 링크를 추가함

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

〈3〉 글상세 기능 구현

1 글상세 링크 및 파라미터 설정

- 하이퍼링크를 설정할 때는 '@{ 링크주소 }' 구문을 사용함

1 Thymeleaf 기반의 게시판

〈3〉 글상세 기능 구현

1 글상세 링크 및 파라미터 설정

- 링크와 함께 파라미터 정보를 전달하기 위한 조건

링크 주소 뒤에 괄호 추가

'키=값' 형태로 파라미터 전달

```
<tr th:each="board : ${boardList}">
    <td th:text="${board.seq}">
    <td><a th:href="@{getBoard(seq=${board.seq})}" th:text="${board.title}"></a></td>
    <td th:text="${board.writer}">
    <td th:text="${board.createDate}">
    <td th:text="${board.cnt}">
</tr>
```

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

〈3〉 글상세 기능 구현

1 글상세 링크 및 파라미터 설정

- 파라미터가 여러 개인 경우에는 콤마(,)를 이용하여 전달할 수 있음

```
<td><a th:href="@{getBoard(seq=${board.seq}, password=${board.password})}"
       th:text="${board.title}"></a></td>
```

1 Thymeleaf 기반의 게시판

〈3〉 글상세 기능 구현

2 HTML 파일 작성

- templates/board 폴더에 getBoard.html 파일을 작성함

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>게시 글 상세</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h1 th:align="center">게시글 상세</h1>

<!-- 상세 정보를 출력 -->

<p th:align="center">
<a th:href="@{getBoardList}">글목록</a>
</p>
</body>
</html>
```

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

〈3〉 글상세 기능 구현

3 글상세 데이터 출력

- getBoard.html 파일에 BoardController.getBoard 메소드에서 처리한 검색 결과를 출력함

1 Thymeleaf 기반의 게시판

〈3〉 글상세 기능 구현

3 글상세 데이터 출력

```
<h1 th:align="center">게시글 상세</h1>
<table th:align="center" border="1" th:cellpadding="0" th:cellspacing="0">
<tr>
<td bgcolor="orange" th:text="제목" width="80"></td>
<td><input name="title" type="text" th:value="${board.title}"></td>
</tr>
<tr>
<td bgcolor="orange" th:text="작성자"></td>
<td th:text="${board.writer}"></td>
</tr>
<tr>
<td bgcolor="orange" th:text="내용">
<td><textarea name="content" th:text="${board.content}" cols="40" rows="10"></textarea></td>
</tr>
<tr>
<td bgcolor="orange" th:text="등록일"></td>
<td th:text="${board.createDate}"></td>
</tr>
<tr>
<td bgcolor="orange" th:text="조회수"></td>
<td th:text="${board.cnt}"></td>
</tr>
</table>
```

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

〈3〉 글상세 기능 구현

4 컨트롤러 수정

- BoardController.getBoard 메소드가 리턴하는 화면 정보를 수정함

```
@GetMapping("/board/getBoard")
public String getBoard(Board board, Model model) {
    model.addAttribute("board", boardService.getBoard(board));
    return "board/getBoard";
}
```

1 Thymeleaf 기반의 게시판

〈4〉 글등록 기능 구현

1 글등록 링크 추가

- 글등록 화면으로 이동하기 위해서 getBoardList.html과 getBoard.html에 각각 링크를 추가함

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

④ 글등록 기능 구현

1 글등록 링크 추가

getBoard.html	 *<a th:href="@{insertBoard}">글등록 * <a th:href="@{getBoardList}">글목록 </body> </html>
getBoardList.html	 *<a th:href="@{insertBoard}">글등록* </body> </html>

1 Thymeleaf 기반의 게시판

④ 글등록 기능 구현

2 HTML 파일 작성

- templates/board 폴더에 insertBoard.html 파일을 작성함

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

⟨4⟩ 글등록 기능 구현

2 HTML 파일 작성

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>새 글 등록</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h1 th:align="center">게시글 등록</h1>

<!-- 글 등록 form을 제공한다. -->

<br>
<p th:align="center">
<a th:href="@{getBoardList}">글목록으로...</a>   
</p>
</body>
</html>
```

1 Thymeleaf 기반의 게시판

⟨4⟩ 글등록 기능 구현

3 글등록 form 작성

- insertBoard.html 파일에 글등록을 위한 ⟨form⟩을 제공함

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

④ 글등록 기능 구현

3 글등록 form 작성

```
<form th:action="insertBoard" method="post">
<table th:align="center" border="1" th:cellpadding="0" th:cellspacing="0">
<tr>
<td bgcolor="orange" th:text="제목" width="80"></td>
<td><input name="title" type="text" size="50"></td>
</tr>
<tr>
<td bgcolor="orange" th:text="작성자"></td>
<td><input name="writer" type="text" size="10"></td>
</tr>
<tr>
<td bgcolor="orange" th:text="내용">
<td><textarea name="content" cols="50" rows="10"></textarea></td>
</tr>
<tr>
<td colspan="2" align="center">
<input type="submit" value="게시글 등록">
</td>
</tr>
</table>
</form>
```

1 Thymeleaf 기반의 게시판

④ 글등록 기능 구현

4 다국어 적용

- Thymeleaf에서 다국어를 적용하려면 '#{메시지 키}'를 사용함

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

⟨4⟩ 글등록 기능 구현

4 다국어 적용

- JSP에서의 다국어 처리와는 달리 taglib 지시자를 선언할 필요가 없음

1 Thymeleaf 기반의 게시판

⟨4⟩ 글등록 기능 구현

4 다국어 적용

JSP

```
<%@page contentType="text/html; charset=UTF-8"%>
<%@taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>새글등록</title>
</head>
<body>
<center>
<h3><spring:message code="board.insertBoard.mainTitle"/></h3>
```

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

⟨4⟩ 글등록 기능 구현

4 다국어 적용

Thymeleaf

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>새 글 등록</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h1 th:align="center" th:text="#{board.insertBoard.mainTitle}"></h1>
```

1 Thymeleaf 기반의 게시판

⟨5⟩ 글수정 기능 구현

- 글상세 화면(getBoard.html)에 <form> 태그와 HIDDEN 타입의 seq 파라미터를 추가함

Thymeleaf를 이용한 화면 개발

1 Thymeleaf 기반의 게시판

⑤ 글수정 기능 구현

- 화면 하단에 컨트롤을 호출하기 위해 [게시글 수정] 버튼을 추가함

```
<form th:action="updateBoard" method="post">
<input name="seq" type="hidden" th:value="${board.seq}">
<table th:align="center" border="1" th:cellpadding="0" th:cellspacing="0">
<tr>
<td bgcolor="orange" th:text="제목" width="80"></td>
<td><input name="title" type="text" th:value="${board.title}"></td>
</tr>
~ 생략 ~

<tr>
<td bgcolor="orange" th:text="등록일"></td>
<td th:text="${board.createDate}"></td>
</tr>
<tr>
<td bgcolor="orange" th:text="조회수"></td>
<td th:text="${board.cnt}"></td>
</tr><tr><td colspan="2" align="center"><input type="submit" value="게시글 수정"></td></tr>
</table>
</form>
```

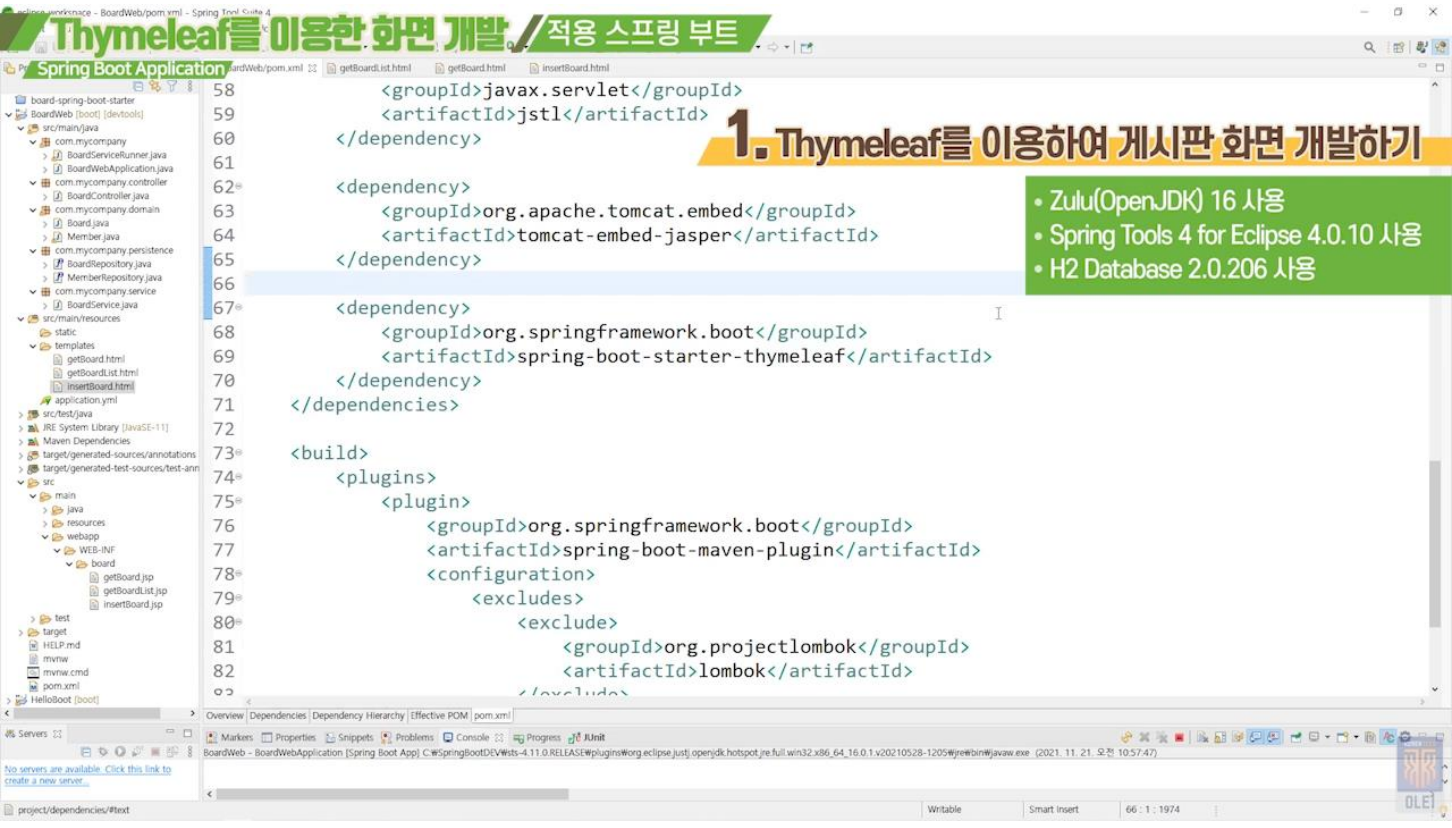
1 Thymeleaf 기반의 게시판

⑥ 글삭제 기능 구현

- 게시글을 삭제하기 위해서는 가장 먼저 상세 화면 (getBoard.html)에 [글삭제] 링크만 추가하면 됨

```
<p th:align="center">
<a th:href="@{insertBoard}">글등록</a>   
<a th:href="@{deleteBoard(seq=${board.seq})}">글삭제</a>   
<a th:href="@{getBoardList}">글목록</a>
</p>
</body>
</html>
```

적용 스프링 부트



실습단계
Thymeleaf를 이용하여 게시판 화면 개발
더 이상 JSP는 사용하지 않음
pom.xml 파일을 이용하여 thymeleaf 스타터 추가
화면만 JSP → Thymeleaf로 변환
templates → New → File
중요! HTML 루트 엘리먼트 Thymeleaf에 해당하는 namespace 선언
중요! th:each 속성을 이용
브라우저의 특정 위치에 텍스트 데이터를 출력하기 위해 th:text 속성 사용
'@'를 이용하여 링크 추가
BoardWebApplication 실행
다시 실행하는 이유는 새로운 라이브러리를 추가했기 때문
브라우저를 띄우고 localhost:8080/getBoardList 요청
예전에 등록했던 글목록
날짜 포맷 변경

적용 스프링 부트

실습단계
board.createDate 부분을 #dates.format으로 수정
상태 변수 추가
state 변수 추가하여 일련번호 출력, 비어있는 번호 없이 출력을 위해 state.count 작성
하이퍼링크 추가
타이틀 부분 변경
JSP 쿼리 스트링을 통해서 ? 뒤에 파라미터를 보내는 것과 개념은 같음
글등록, 글목록 링크
Tip! 화면을 잠깐 멈추고 소스를 타이핑하며 진행할 것
중요! 수정할 때엔 seq 정보를 딸려 보내야 함
table 태그 위에 form 태그 추가
table 태그 아래에 종료 form 추가
insertBoard.html 파일 작성, insertBoard.jsp와 유사함
JSP에서 JSTL 작성한 것과 거의 유사함
Thymeleaf 관련 네임 스페이스 등록, th 제공 속성을 이용해 화면을 구성한다는 것이 차이점