

Spring MVC

16 thymeleaf 알아보기

thymeleaf 알아보기

1. 타임리프(thymeleaf)란?

자바 웹개발에 이상적인 '**모던 서버 사이드 자바 템플릿 엔진**'

HTML과 유사해서 디자이너와 개발자간의 협업을 쉽게 해준다.

확장성이 뛰어나며, 커스터마이징이 쉽다.

다양한 도구와 확장 프로그램으로 구성된 에코 시스템 제공

<https://www.thymeleaf.org/ecosystem.html>



Thymeleaf

<https://www.thymeleaf.org/>

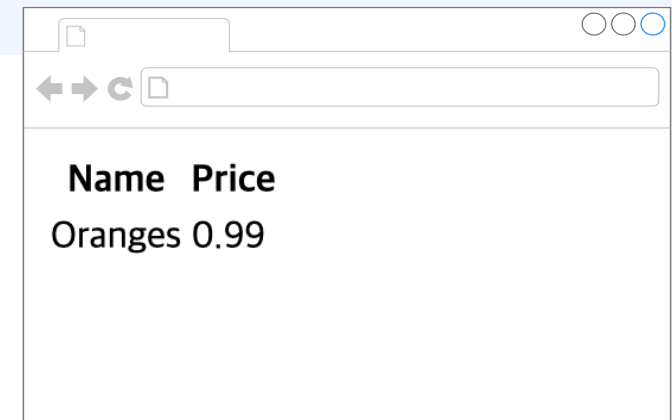
thymeleaf 알아보기

2. 타임리프 템플릿

타임리프 템플릿(*.html)은 HTML과 유사해서 편집 후 내용 확인이 쉽다.
th:* 속성은 타임리프 전용 속성이며, 브라우저는 이를 무시한다.

```
<table>
  <thead>
    <tr>
      <th th:text="#{msgs.headers.name}">Name</th>
      <th th:text="#{msgs.headers.price}">Price</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="prod: ${allProducts}">
      <td th:text="${prod.name}">Oranges</td>
      <td th:text="${#numbers.formatDecimal(prod.price, 1, 2)}">0.99</td>
    </tr>
  </tbody>
</table>
```

```
<th>${msgs.headers.name}</th>
<th>${msgs.headers.price}</th>
```



| Name | Price |
|---------|-------|
| Oranges | 0.99 |

thymeleaf 알아보기

3. th:text와 th:utext

th:text는 `${...}`을 해석해서 태그의 텍스트 노드로

```
<span th:text="${lastName}">Namkung</span>
```

```
<span>[[${lastName}]]</span>
```

문자열('...') 결합(+)과 리터럴 치환(|...|)

```
<span th:text="'My name is' + ${lastName} + ', ' + ${firstName}'></span>
```

```
<span th:text="' | My name is ${lastName}, ${firstName} |'></span>
```

th:utext는 태그의 `<`, `>`를 `<`, `>`로 바꾸지 않고 그대로

```
<span th:text="${'<i>Namkung, Seong</i>'}">Namkung, Seong</span>
```

```
<span th:utext="${'<i>Namkung, Seong</i>'}">Namkung, Seong</span>
```

```
<span>&lt;i&gt;Namkung, Seong&lt;/i&gt;</span>
<span><i>Namkung, Seong</i></span>
```

thymeleaf 알아보기

4. th:if, th:unless, th:switch로 조건부 처리

특정 조건에 맞는 경우만 처리

```
<tr th:if="${list.size}==0">
  <td>게시물이 없습니다.</td>
</tr>
```

```
<tr th:unless="${list.size}!=0">
  <td>게시물이 없습니다.</td>
</tr>
```

```
<tr th:class="${status.even} ? 'even' : 'odd'">
  ...
</tr>
```

```
<div th:switch="${user.grade}">
  <span th:case="A">특 급</span>
  <span th:case="B">고 급</span>
  <span th:case="C">중 급</span>
  <span th:case="*">기 타</span>
</div>
```

```
<th:block th:switch="${user.grade}">
  <span th:case="A">특 급</span>
  <span th:case="B">고 급</span>
  <span th:case="C">중 급</span>
  <span th:case="*">기 타</span>
</th:block>
```

thymeleaf 알아보기

5. th:each와 th:block을 이용한 반복(1)

Iterable의 반복 처리는 th:each 또는 th:block 사용. 향상된 for문과 유사

```
<select multiple>
```

```
  <option th:each="opt : ${list}" th:value="${opt}">[[${opt}]]</option>
</select>
```

th:each를 쓰기 어려운 경우, th:block으로 처리

```
<th:block th:each="opt : ${list}">
```

```
  <input type="checkbox" th:value="${opt}">[[${opt}]]<br/>
</th:block>
```

thymeleaf 알아보기

5. th:each와 th:block을 이용한 반복(2) – status변수

반복 관련 정보(index, count, size, odd, even, first, last, current) 제공

```
<select multiple>
```

```
  <option th:each="opt, status : ${list}" th:value="${opt}">
```

```
    [[${status.index}}]. [[${opt}]]
```

```
  </option>
```

```
</select>
```

status변수의 선언을 생략하면, '변수명+Stat'을 사용

```
<select multiple>
```

```
  <option th:each="opt : status : ${list}" th:value="${opt}" th:selected="${optStat.first}">
```

```
    [[${optStat.index}}]. [[${opt}]]
```

```
  </option>
```

```
</select>
```

thymeleaf 알아보기

6. th:attr와 th:attrappend, th:attrprepend로 속성 값 설정하기

th:attr은 속성의 값을 설정하는데 사용

```

```



```

```

대부분의 속성(attribute)은 th:속성이름도 가능

```

```

th:attrappend, th:attrprepend로 기존의 속성 값에 새로운 값을 추가 가능

```
<input type="button" value="Go" class="btn" th:attrappend="class='${ ' + style}'" />
```



```
<input type="button" value="Go" class="btn new-style"/>
```


thymeleaf 알아보기

7. URL 링크 - @{...}

@{...}를 경로로 변환. '/'로 시작할 때는 context root를 추가.

```
<a href="boardList.html" th:href="@{/board/list}">
```



```
<a href="/ch2/board/list">
```

Query parameter와 Path variable(여러 개일 때는 구분자 ';' 사용)

```
<a href="board.html" th:href="@{/board/read(bno=${bno}, type='L')}">
```



```
<a href="/ch2/board/read?bno=123&type=L">
```

```
<a href="board.html" th:href="@{/board/{bno}/read(bno=${bno})}">
```



```
<a href="/ch2/123/read">
```

thymeleaf 알아보기

8. 주석(comment)

<!-- --> : HTML 주석. 주석 내의 부분은 타임리프가 처리 안함.

<!--/* */--> : parser-level 주석. parser가 처리할 때 무시. 예러가 있어도 OK

<!--/*/ */--> : prototype-only 주석. html에서는 주석이지만, 처리되면 주석 아님

```
<!--      <span th:text="${list}"></span>      -->
```

```
<!--/*   <span th:text="${list}"></span>   */-->
```

```
<!--/*/  <span th:text="${list}"></span>  /*/-->
```

```
<!--/*/ <th:block th:each="opt : ${list}"> /*/-->
```

```
    <input type="checkbox" th:value="${opt}">[[${opt}]]<br/>
```

```
    <!--/*-->
```

```
    <input type="checkbox" value="sample1">sample1<br/>
```

```
    <input type="checkbox" value="sample2">sample2<br/>
```

```
    <!--*/-->
```

```
<!--/*/ </th:block> /*/-->
```

thymeleaf 알아보기

9. 자바스크립트 인라이닝

[[\${...}]]를 자바스크립트에 맞게 적절히 변환해주는 편리한 기능

```
<script>
  var firstName = [[${firstName}]]
  var lastName = /*[[${lastName}]]*/ "testName"
  var arr = [[${list}]]
  var userObj = [[${user}]]
</script>
```

```
<script>
  var firstName = Seong
  var lastName = /*Namkung*/ "testName"
  var arr = [aaa, bbb, ccc, ddd, eee, fff]
  var userObj = com.example.boot_ch2.User@6adf734d
</script>
```

```
<script th:inline="javascript">
  var firstName = [[${firstName}]]
  var lastName = /*[[${lastName}]]*/ "testName"
  var arr = [[${list}]]
  var userObj = [[${user}]]
</script>
```

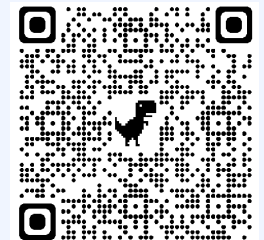
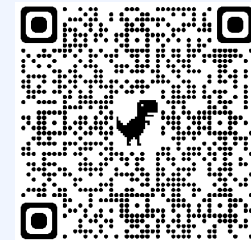
```
<script>
  var firstName = "Seong"
  var lastName = "Namkung"
  var arr = ["aaa", "bbb", "ccc", "ddd", "eee", "fff"]
  var userObj = {"name": "aaa", "age": 20}
</script>
```

thymeleaf 알아보기

10. 유틸리티 객체

유용한 메서드를 제공하는 객체들. 변환 & 형식화를 쉽게

- 문자열 & 숫자 : #strings, #numbers
- 날짜 & 시간 : #dates, #calendars, #temporals
- 배열 & 컬렉션 : #arrays, #lists, #sets, #maps
- 기타 : #uris – URI/URL의 escape/unescape 처리
#conversions – 스프링의 변환기능(ConversionService) 지원
#messages – 자바의 메시지 형식화 국제화 지원
#objects – null 확인 기능 제공
#bools – boolean연산(and, or) 기능 제공



[참고] <https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#appendix-b-expression-utility-objects>

[참고] <https://www.thymeleaf.org/apidocs/thymeleaf/3.0.0.RELEASE/index.html?overview-summary.html>

thymeleaf 알아보기

11. 기본 객체

서블릿의 기본 객체(request, session, application 등)에 접근방법을 제공

```
@GetMapping("/test")
```

```
public String test(Model model, HttpServletRequest request) {
```

```
    //...
```

```
    request.setAttribute(s: "year", o: 2022);
```

```
    HttpSession session = request.getSession();
```

```
    session.setAttribute(s: "id", o: "asdf");
```

```
    ServletContext application = session.getServletContext();
```

```
    application.setAttribute(s: "email", o: "service@fastcampus.com");
```

```
<h1 th:text="|year : ${year}|"></h1>
```

```
<h1 th:text="|id : ${session.id}|"></h1>
```

```
<h1 th:text="|email : ${application.email}|"></h1>
```

```
<h1 th:text="|year : ${#request.getAttribute('year')}|"></h1>
```

```
<h1 th:text="|id : ${#session.getAttribute('id')}|"></h1>
```

```
<h1 th:text="|email : ${#servletContext.getAttribute('email')}|"></h1>
```

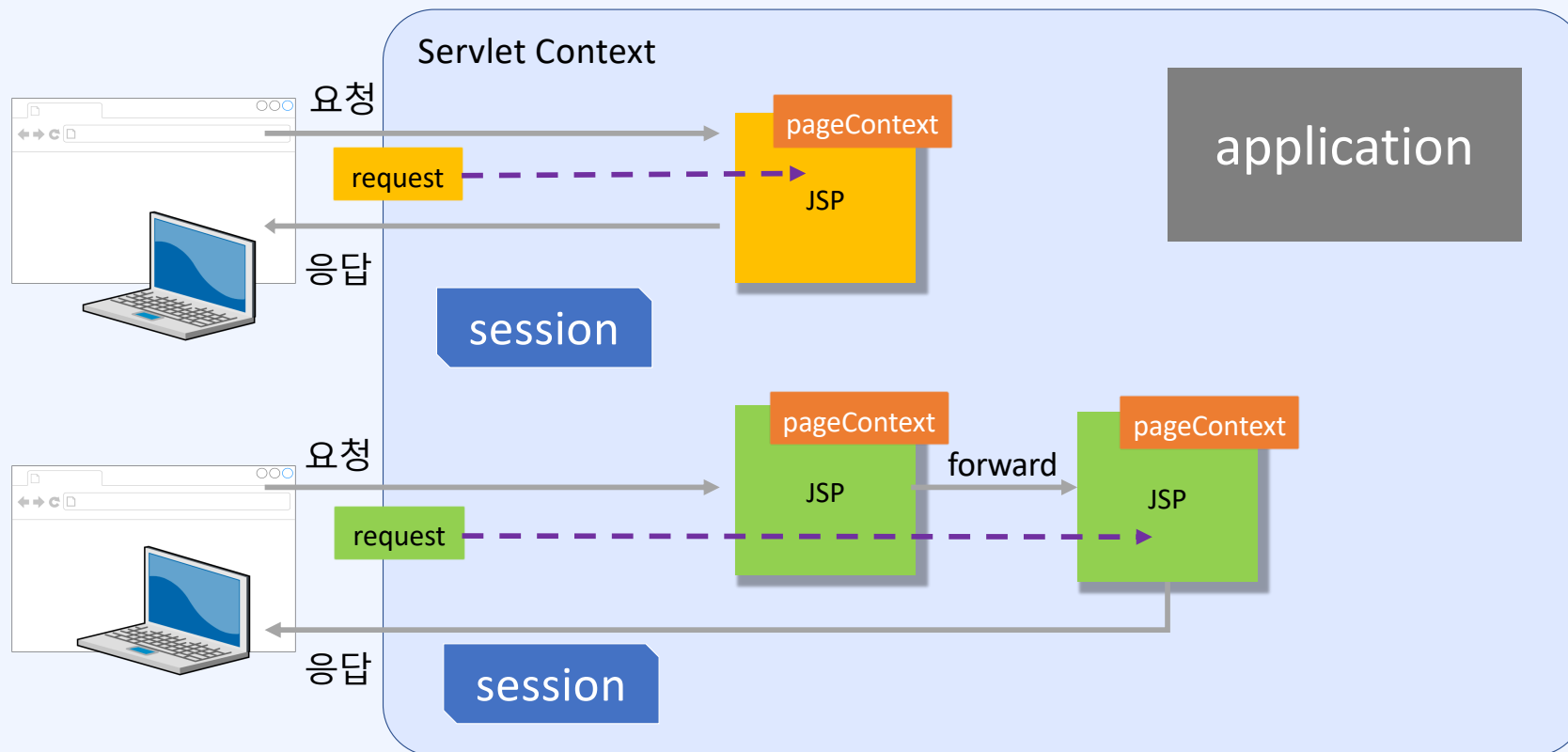
thymeleaf 알아보기

11. 기본 객체

| Key | Value |
|-----|-------|
| 속성1 | 속성값 |
| 속성2 | 속성값 |
| ... | ... |

2.

Spring MVC



thymeleaf 알아보기

11. 기본 객체

| 기본 객체 | 유효 범위 | 설명 |
|-------------|------------|--|
| pageContext | 1개 JSP페이지 | JSP페이지의 시작부터 끝까지. 해당 JSP 내부에서만 접근가능. 페이지당 1개 |
| request | 1+개 JSP페이지 | 요청의 시작부터 응답까지. 다른 JSP로 전달 가능. 요청마다 1개 |
| session | n개 JSP페이지 | session의 시작부터 종료까지(로그인 ~ 로그아웃). 클라이언트마다 1개 |
| application | context 전체 | Web Application의 시작부터 종료까지. context내부 어디서나 접근 가능 모든 클라이언트가 공유. context마다 1개 |

| 속성 관련 메서드 | 설명 |
|--|------------------------------------|
| void setAttribute (String name, Object value) | 지정된 값(value)을 지정된 속성 이름(name)으로 저장 |
| Object getAttribute (String name) | 지정된 이름(name)으로 저장된 속성의 값을 반환 |
| void removeAttribute (String name) | 지정된 이름(name)의 속성을 삭제 |
| Enumeration getAttributeNames () | 기본 객체에 저장된 모든 속성의 이름을 반환 |