1. 스프링 부트에서는 뷰(View)페이지 개발에 기존의 jsp를 사용하는 것에서 벗어나 또 다른 템플릿 기반의 화면 처리가 지원된다.

1-1. 따라서 템플릿 엔진으로 Thymeleaf, Apache Freemarker, Mustache, Groovy Templates 을 제공한다. 그 중에서 Thymeleaf를 이용해서 확장자가 html인 뷰페이지 개발을 할 수가 있다.

1-2 . src/main/resources 폴더 하위의 있는 static은 js, css, html, 이미지 파일들을 추가하는 경로이고, templates 내부에는 ThymeLeaf를 이용한 템플릿을 넣게 된다. 여러 개의 템플릿 엔진을 포함하지 않은 이상, 스프링 부트의 자동 구성 기능이 작동하여 이미 ThymeLeaf에 대한 기본 설정은 완료된 상태가 된다.

1-3. ThymeLeaf는 기본적으로 '.html' 확장자를 사용하고, 작성된 하면은 서버의 내부에 보관되어 재처리 없이 빠르게 서비스할 수 있는 환경으로 반영된다.

1-4. ThymeLeaf의 기본 설정으로 개발 시 알아두어야 하는 내용으로는 다음과 같다.

가. 확장자는 '.html'을 기본으로 설정
나. 인코딩은 'UTF-8'방식으로 설정
다. Mime 타입은 'text/html' 으로 설정
라. 서버 내부의 cache는 'true' 로 설정

1-5. 위의 기본 설정으로 뷰페이지 개발시 ThymeLeaf 로 개발된 화면을 수정하면, 매번 프로젝트를 재시작하는 불편함이 있기 때문에 개발 시에는 작성한 화면을 서버 내부에 보관(cache)하지 않도록 설정해 주는 것이 좋다.

1-6. application.properties 설정파일에 다음과 같이 설정한다.

```
#No Cache
spring.thymeleaf.cache=false
```

2. net.daum.controller 패키지를 만들고 SampleController를 만든다.

```
package net.daum.controller;

import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import net.daum.vo.MemberVO;

@Controller
public class SampleController {

	@GetMapping("/start_thymeLeaf")
	public ModelAndView start_thymeLeaf() {

		ModelAndView tm=new ModelAndView();
		tm.addObject("greeting2","안녕하세요!! 방갑습니다.");
		tm.setViewName("./thymeLeaf/start_thymeLeaf");
		return tm;
	}
}
```

STS상에서 html 파일을 생성하면 자동으로 'src/main/webapp' 폴더로 들어가게 된다. 따라서 일반 File 메뉴를 통해서 start_thymeLeaf.html 파일을 생성한다. 사전에 src/main/resources 경로에 있는 templates에 thymeLeaf라는 폴더를 생성한다.

```
<html xmlns:th="http://www.thymeleaf.org">
 <head>
  <title>Thymeleaf</title>
  <meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
  <link rel="stylesheet" type="text/css" href="./css/th.css" >
 </head>
 <body>
  <h1>ThymeLeaf 연습 페이지</h1>
  <h2 class="thymeLeaf_css" th:text="${greeting2}"></h2>
  <hr>

  <h3>맛난 복숭아</h3>
  <img src="./images01/fruit01.jpg" width="150px" height="100" alt="복숭아">

  <h3>맛난 사과</h3>
```

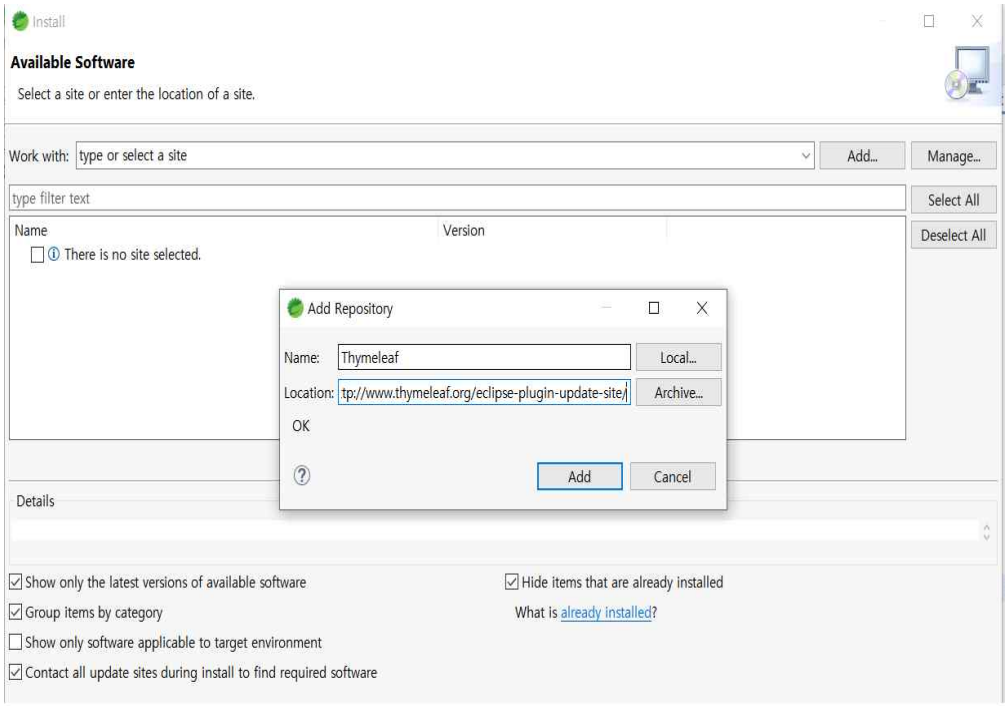
```
  <img src="./images02/apple.jpg" width="150px" height="100" alt="사과">
 </body>
</html>
```

스프링 부트를 이용해서 웹을 개발할 때에는 'DevTools'를 포함한 상태에서 개발하는 것을 권장한다. 이유는 컨트롤러 소스 코드를 수정하면 자동으로 스프링 부트를 재시작해서 반영해 주기 때문이다.

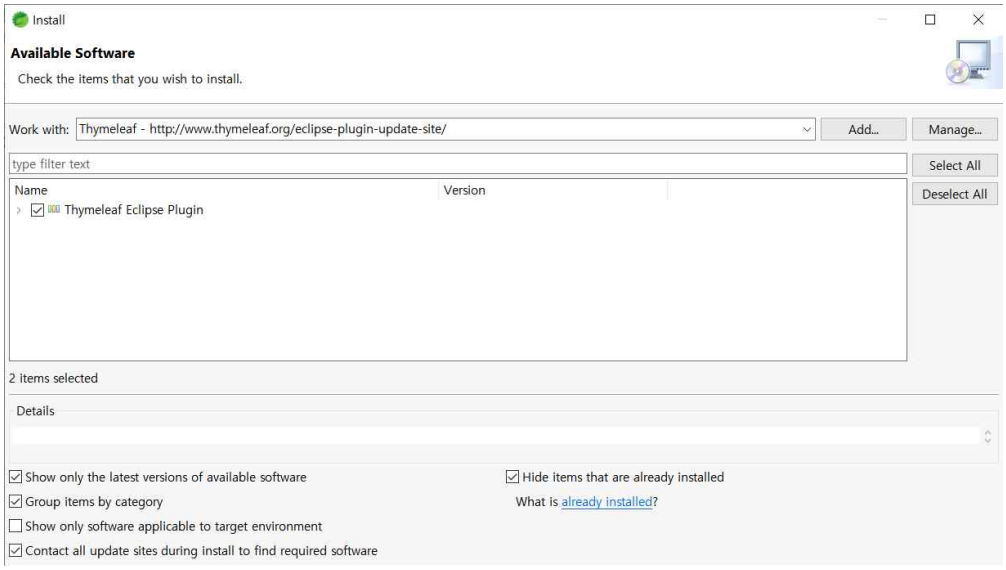
3. STS에 ThymeLeaf 플러그인을 설치한다.

STS 혹은 이클립스에는 타임리프 자동완성을 위한 플러그인이 기본으로 설치되어 있지 않다. 자동완성이 되지 않는다면 개발하기가 몹시 불편하므로 자동완성을 위한 플러그인을 설치해 본다.

3-1. ThymeLeaf 플러그인 STS Help -> Install New Softwares 메뉴를 이용한다.

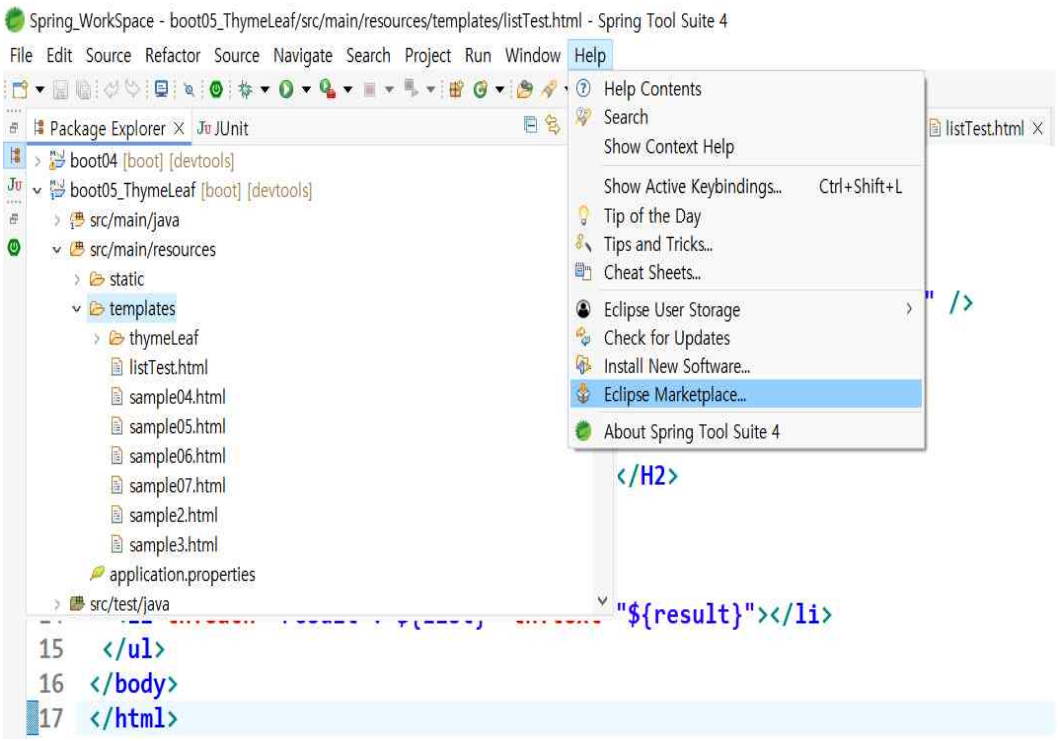


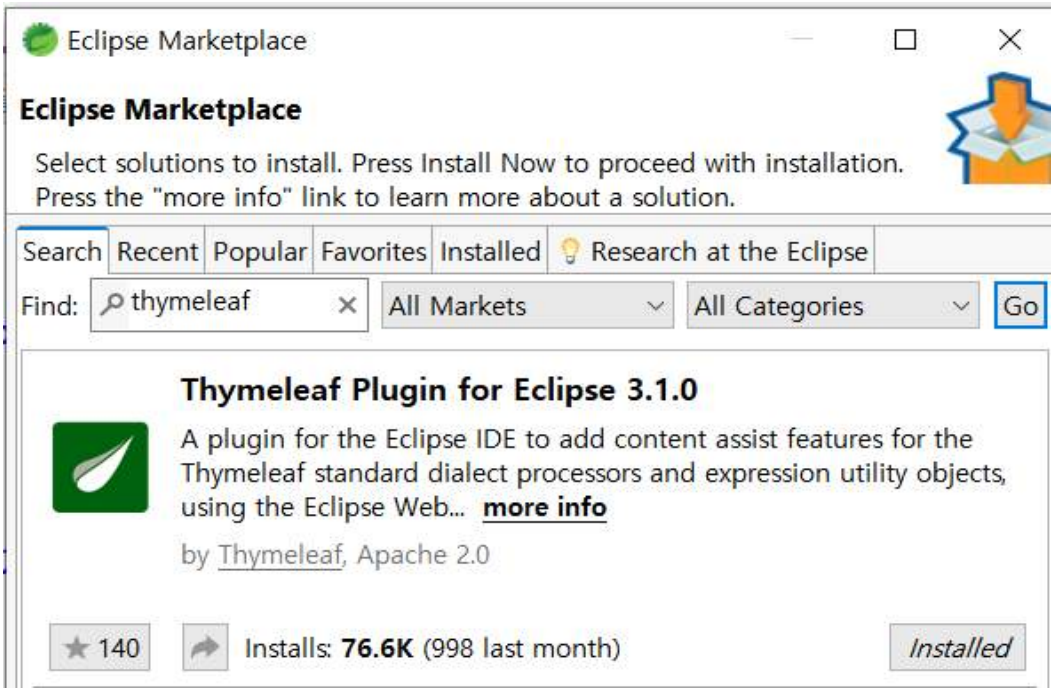
Name 입력란에 Thymeleaf를 입력하고 Location에는 http://www.thymeleaf.org/eclipse-plugin-update-site/를 입력한다.



플러그 인 설치가 완료되면 STS를 재 시작한다.

3-2. 또 다른 두 번째 방법으로 STS에 타임리프 플러그인을 설치해 본다.





Find 검색창에서 thymeleaf 를 입력하고 Go 버튼을 클릭한 다음 install 버튼을 클릭해서 사용권 계약 동의하고 설치한다. 설치가 다 끝나면 STS를 재시작 한다.
위 그림에서는 위쪽 방법으로 미리 설치를 해서 installed로 나온다.

```
    <title>Thymeleaf</title>
    <meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
  </head>
  <body>
    <h1>ThymeLeaf로 복수개의 리스트 목록 연습</h1>
    <hr>
    <H2 th:text="'Hello, ' + ${name} + '!'"></H2>
    <hr>

    <ul>
      <li th:e="result : ${list}" th:text="${result}"></li>
    </ul>
  </body>
  </html>
```

th:each
th:enctype
Press 'Ctrl+Space' to show HTML Template Proposals

STS에 타임리프 플러그인이 설치가 되어져서 위의 그림처럼 타임리프 자동완성이 나오게 된다. 자동완성 단축키는 ctrl+space 이다.

4-1. net.daum.controller 패키지의 SampleController.java에 다음의 코드를 추가한다.

```
   중략...
        @RequestMapping("/listTest")
         public void listTest(Model model) {

                  List<String> list=new ArrayList<>();

                  for(int i=1;i<=10;i++) {
                           list.add("Data : "+i);
                  }//for

                  model.addAttribute("name","타임리프 연습");
                  model.addAttribute("list",list);
         }
중략...
```

net.daum.vo 패키지를 만들고 MemberVO.java를 만든다.

4-2. net.daum.controller 패키지의 SampleController.java에 다음의 코드를 추가한다.

```
        @GetMapping("/sample2")
         public void sample2(Model model) {
                  MemberVO vo = new MemberVO(123, "u00", "p00", "홍길동",
new Timestamp(System.currentTimeMillis()));
                  model.addAttribute("vo", vo);
         }//sample2()
```

sample2.html 파일을 작성한다.

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
 <head>
  <title>Thymeleaf 3</title>
  <meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
 </head>
 <body>
  <h1 th:text="${vo}"></h1>
  <hr>
```

```
  <div th:utext="${'<h2>'+vo.mid+'</h2>'}"></div> <!-- th:utext속성은 html태그를 적용
-->
  <div th:text="${'<h2>'+vo.mname+'</h2>'}"></div> <!-- th:text속성은 <를 &lt; 특수
문자로 인식하고 >를  &gt; 로 각각 인식해서
   < >을 그대로 웹상에 출력되어서 html이 적용안됨.-->
 </body>
 </html>
```

4-3. net.daum.controller 패키지의 SampleController.java에 다음의 코드를 추가한다.

```
중략 ..
        @GetMapping("/sample3")
         public void sample3(Model model) {

                 List<MemberVO> list = new ArrayList<>();

                 for (int i = 0; i < 10; i++) {
                         list.add(new MemberVO(123+1, "u0" + i, "p0" + i, "홍길동
" + i, new Timestamp(System.currentTimeMillis())));
                 }
                 model.addAttribute("list", list);
         }//sample3
중략 ...
```

sample3.html 파일을 작성한다.

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Thymeleaf 3</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

        <table border="1">
                 <tr>
```

```
                        <td>MID</td>
                        <td>MNAME</td>
                        <td>REGDATE</td>
                </tr>

                <tr th:each="member : ${list}">
                        <td th:text="${member.mid}"></td>
                        <td th:text="${member.mname}"></td>
                        <td              th:text="${#dates.format(member.regdate,
'yyyy-MM-dd')}"></td>
                </tr>

        </table>

        <hr>

        <table border="1">
                <tr>
                        <td>COUNT</td>
                        <td>SIZE</td>
                        <td>ODD/EVEN</td>
                        <td>MID</td>
                        <td>MNAME</td>
                        <td>REGDATE</td>
                </tr>

                <tr th:each="member, count2 : ${list}">
                        <td th:text="${count2.count} "></td>
                        <!-- index는 0부터 시작하는 인덱스 번호, size 현재 목록 개수
(크기),odd는 홀수이면 true, even짝수이면 true, count 1부터 시작하는 번호-->
                        <td th:text="${count2.size} "></td>
                        <td th:text="${count2.odd +' ' + count2.even} "></td>
                        <td th:text="${member.mid} "></td>
                        <td th:text="${member.mname}"></td>
                        <td              th:text="${#dates.format(member.regdate,
'yyyy-MM-dd')}"></td>
                </tr>

        </table>
</body>
```

```
</html>
```

4-4. net.daum.controller 패키지의 SampleController.java에 다음의 코드를 추가한다.

```
..중략...
        @GetMapping("/sample04")
         public void sample04(Model model) {

                List<MemberVO> list = new ArrayList<>();

                for (int i = 0; i < 10; i++) {
                        list.add(new MemberVO(i+1, "u000" + i % 3, "p0000" + i
% 3, "홍길동" + (i+1),
                                        n                e                w
Timestamp(System.currentTimeMillis())));
                }
                model.addAttribute("list", list);
         }//sample4()
..중략...
```

sample04.html 파일을 작성한다.

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Thymeleaf3</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

  <table border="1" th:with="target='u0001'">
  <!-- th:with를 사용해서 특정한 범위내에서만 유효한 지역변수 선언할 수 있다. target이
라는 변수 선언함. 이변수는
  table 태그 내에서만 유효한 지역변수이다. -->
    <tr>
      <td>MID</td>
      <td>MNAME</td>
      <td>REGDATE</td>
```

```
    </tr>

    <tr th:each="member : ${list}">
       <td th:text="${member.mid == target ?'SECRET':member.mid }"></td>
                          <!-- 삼항 조건 연산자 사용 -->
       <td th:text="${member.mname}"></td>
       <td th:text="${#dates.format(member.regdate, 'yyyy-MM-dd')}"></td>
     </tr>

  </table>

   <hr>

        <table border="1" th:with="target2='u0001'">
                <tr>
                        <td>MID</td>
                        <td>MNAME</td>
                        <td>REGDATE</td>
                </tr>

                <tr th:each="member : ${list}">
                        <td th:if="${member.mid}">
                                <a    href="/modify"    th:if="${member.mid    ==
target2}">MODIFY</a> <!-- th:if~ th:unless 타임리프는 자바의 if~else와 기능이 같다.
-->
                                          <!-- 회원아이디가 u0001과 같다면 ->
true일때 실행-->
                          <p th:unless="${member.mid == target2}">VIEW</p>
                          <!-- 회원아이디가 u0001과 다르다면 -> false일때 실행 -->
                        </td>
                        <td th:text="${member.mname}"></td>
                        <td              th:text="${#dates.format(member.regdate,
'yyyy-MM-dd')}"></td>
                </tr>

        </table>

</body>
</html>
```

4-5. net.daum.controller 패키지의 SampleController.java에 다음의 코드를 추가한다.

```
..중략...
        @GetMapping("/sample05")
         public void sample05(Model model) {

                  String result = "SUCCESS";

                  model.addAttribute("result", result);
         }//sample05()
..중략...
```

sample05.html 파일을 작성한다.

```
<html xmlns:th="http://www.thymeleaf.org">

<head>
<title>Thymeleaf 05</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

<script th:inline="javascript">
/* th:inline="javascript" 속성을 사용하면 <script>태그내에 타임리프 문법을 사용할 수 있다.  */

var result = [[${result}]];
document.write("타임리프 가져온값 :"+ result + "<hr>");
</script>

<script>
/* 스프링 컨트롤러의 문자열 값 SUCCESS가 자바스크립트의 변수처럼 선언되는 것을 볼 수 있다. th:inline속성을 사용하지
 * 않아서 자바스크립트 코드 내에서 타임리프 문법 사용이 제한된다. */

var result = [[${result}]];
```

```
</script>

</body>
</html>
```

4-6. net.daum.controller 패키지의 SampleController.java에 다음의 코드를 추가한다.

```
..중략..
        @GetMapping("/sample06")
         public void sample06(Model model) {

                List<MemberVO> list = new ArrayList<>();

                for (int i = 0; i < 10; i++) {

                        list.add(new MemberVO(i, "u0" + i, "p0" + i, "홍길동" + i,
                                        n                    e                    w
Timestamp(System.currentTimeMillis())));
                }

                model.addAttribute("list", list);

                String result = "SUCCESS";

                model.addAttribute("result", result);
         }//sample06()
..중략...
```

sample06.html 파일을 작성한다.

```
<html xmlns:th="http://www.thymeleaf.org">

<head>
<title>Thymeleaf3</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body >
```

```
        <table border="1" th:inline="text"> <!-- th:inline="text"를 지정해서 일반 태그
에서도 사용가능하다. -->
                <tr>
                        <td>MID</td>
                        <td>MNAME</td>
                        <td>REGDATE</td>
                </tr>

                <tr th:each="member : ${list}">
                        <td>[[${member.mid}]]</td>
                        <td>[[${member.mname}]]</td>
                        <td>[[${member.regdate}]]</td>
                </tr>

        </table>

        <div>[[${result}]]</div>
        <div>[[${#vars.result}]]</div>  <!-- 표현식 기본객체에서 vars은 생략가능함. 옆의
2가지의 표현은 같은 의미이다. -->

        <script th:inline="javascript">

        var result = [[${result}]];
        alert(result);
        </script>


</body>
</html>
```

4-7. net.daum.controller 패키지의 SampleController.java에 다음의 코드를 추가한다.

```
..중략...
        @GetMapping("/sample07")
         public void sample07(Model model) {

                model.addAttribute("now", new Date());
```

```
                model.addAttribute("price", 123456789);
                model.addAttribute("title", "This is a just sample.");
                model.addAttribute("options",
Arrays.asList("AAAA","BBB","CCC","DDD"));//Arrays.asList()는 배열을 컬렉션 List로 변환
        }//sample07()
..중략...
```

sample07.html 파일을 작성한다.

```
<html xmlns:th="http://www.thymeleaf.org">

<head>
<title>Thymeleaf3</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

        <h1 th:text="${now}"></h1>

        <h2 th:text="${#dates.format(now, 'yyyy-MM-dd')}"></h2>

        <div th:with="timeValue=${#dates.createToday()}">
                <p>[[${timeValue}]]</p>
        </div>



        <h1 th:text="${price}"></h1>

        <h2 th:text="${#numbers.formatInteger(price,3,'COMMA')}"></h2>
         <!-- 세자리 수 콤마로 표현,  #numbers.formatInteger(표시해야 할 값, 최소 자
리수, COMMA는 구분자 ,를 의미)  -->

        <div th:with="priceValue=99.87654">
                <                                                       p
th:text="${#numbers.formatInteger(priceValue,3,'COMMA')}"></p>
                <!-- 세자리수 콤마로 표현, formatInteger는 정수 처리를 한다.결국 반올
림해서 세자리 수 정수 100으로 표현-->
                <                                                       p
th:text="${#numbers.formatDecimal(priceValue,5,10,'POINT')}"></p>
```

```
                <!-- formatDecimal(표시 값, 최소 정수 자릿수, 최소 소수 자릿수) ,
POINT는 구분자 .을 의미-->
        </div>



        <h1 th:text="${title}"></h1>

        <span th:utext="${#strings.replace(title,'s','<b>s</b>')}"></span>
        <!-- title에 저장된 영문자에서 s를 진하게 표현하게 변경 -->

        <ul>
          <li th:each="str:${#strings.listSplit(title,' ')}">[[${str}]]
          <!-- title에 있는 영문자에서 공백 문자를 기준으로 문자를 분리한다. -->
          </li>
        </ul>

        <h1 th:text="${options}"></h1>

</body>
</html>
```