# Spring MVC2 01 [ 타임리프 - 기본 기능1 ]

스프링 MVC 2편 - 백엔드 웹 개발 활용 기술

김영한

2022.10.08

MVC2 강의 첫 시간 - 프로젝트 생성부터 기술 학습 시작.

**gradle** 포트 변경

src\main\resources\application.properties

server.port = 9595

## 타임리프 - 기본 기능

https://www.thymeleaf.org/

공식 사이트, 매뉴얼이 잘 되어있어서 찾아봐서 사용하기 좋음.

### 타임리프 특징

- 서버 사이드 **HTML** 렌더링 **( SSR )**

  타임리프는 백엔드 서버에서 HTML을 동적으로 렌더링 하는 용도로 사용된다.

- 네츄럴 템플릿

  타임리프는 순수 HTML을 유지하려는특징이 있음. 서버를 거치지 않으면 정적 HTML 페이지를 제공할 수 있다. 만약 JSP를 서버를 거치지 않고 열게되면 파일이 깨지게 되기 때문에 HTML 결과를 확인 할 수 없다. 네츄럴 템플릿 같은 경우 마크업 결과를 열어서 확인도 할 수 있다.

- 스프링 통합 지원

  타임리프 스프링은 통합 지원 기능을 많이 지원한다.

타임리프 기본 표현식

https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#standard-expression-syntax -공식문서

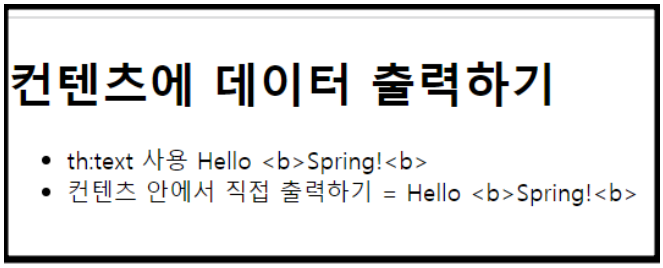
## 텍스트 - text, utext

기본적으로 속성안에 th: 를 통해 변수를 넣을수 있따.

**-data를 model에 담았을때.**

data = "hello Spring!

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <meta charset="UTF-8">
   <title>Title</title>
</head>
<body>
   <h1>컨텐츠에 데이터 출력하기</h1>
<ul>
   <li>th:text 사용 <span th:text="${data}"></span></li>
   <li>컨텐츠 안에서 직접 출력하기 = [[${data}]]</li>
</ul>
</body>
</html>
```



기본적으로 HTML 이스케이프 처리를 해주기 때문에 특수문자를 이스케이프 해서 HTML 엔티티로 나타내게 되어  를 그대로 출력한다.

만약 태그를 언이스케이프해서 사용하기 위해서는 **th:utext , [( )]** 를 사용한다.

```
<body>
<h1>text vs utext</h1>
<ul>
   <li>th:text = <span th:text="${data}"></span></li>
   <li>th:utext = <span th:utext="${data}"></span></li>
</ul>
<h1><span th:inline="none">[[...]] vs [(...)]</span></h1>
<ul>
   <li><span th:inline="none">[[...]] = </span>[[${data}]]</li>
   <li><span th:inline="none">[(...)] = </span>[(${data})]</li>
</ul>
</body>
```

→ 결과 출력

## text vs utext

- th:text = Hello Spring!
- th:utext = Hello **Spring!**

## [[...]] vs [(...)]

- [[...]] = Hello Spring!
- [(...)] = Hello **Spring!**

---

## 변수 - SpringEL

변수 표현식 : **${ ... }**

변수 표현식에는 스프링 EL 사용

```
<h1>SpringEL 표현식</h1>

<ul>Object          →model.addAttribute("user", userA)
   <li>${user.username} = <span th:text="${user.username}"></span></li>
   <li>${user['username']} = <span th:text="${user['username']}"></span></li>
   <li>${user.getUsername()} = <span th:text="${user.getUsername()}"></span></li>

→유저안에 프로퍼티 접근이 가능. 문자로도 접근가능
</ul>
<ul>List            →model.addAttribute("users", list)
   <li>${users[0].username} = <span th:text="${users[0].username}"></span></li>
   <li>${users[0]['username']} = <span
th:text="${users[0]['username']}"></span></li>
   <li>${users[0].getUsername()} = <span
th:text="${users[0].getUsername()}"></span></li>

→ List는 인덱스로 접근

</ul>
<ul>Map             →model.addAttribute("userMap", map)
   <li>${userMap['userA'].username} = <span
th:text="${userMap['userA'].username}"></span></li>
   <li>${userMap['userA']['username']} = <span
th:text="${userMap['userA']['username']}"></span></li>
   <li>${userMap['userA'].getUsername()} = <span
th:text="${userMap['userA'].getUsername()}"></span></li>

→ Map 의 키로 접근

</ul>
```

## 지역 변수로 선언해서 사용하는 것 가능

선언한 태그스코프안에서만 동작한다.

```
<h1>지역 변수 - (th:with)</h1>
<div th:with="first=${users[0]}">
   <p>처음 사람의 이름은 <span th:text="${first.username}"></span></p>
</div>
```

결과

```
처음 사람의 이름은 userA
```

## 기본 객체들

타임리프가 제공하는 기본 객체들

${#request}, ${#response}, ${#session}, ${#servletContext}, ${#locale}

편의 객체

```
<ul>
 <li>Request Parameter = <span th:text="${param.paramData}"></span></li>
 <li>session = <span th:text="${session.sessionData}"></span></li>
 <li>spring bean = <span
th:text="${@helloBean.hello('Spring!')}"></span></li>
</ul>
```

request.getParameter("data") 접근이 불편하기 때문에 param 객체를 지원한다.

스프링빈에 바로 접근하기 위해서는 ( helloBean이라는 스프링 빈을 등록했을시)
**@helloBean**를 사용하면 된다