Spring MVC 10

[웹페이지 만들기 - 1]

스프링 MVC 1편 - 백엔드 웹 개발 핵심 기술

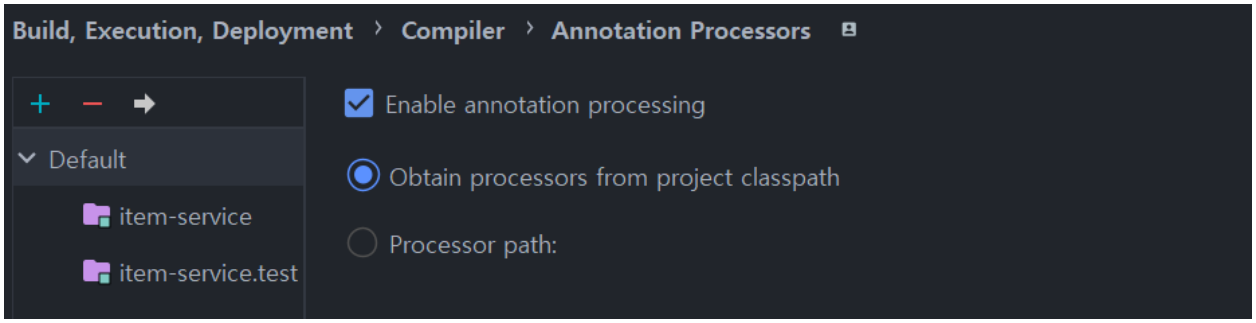
김영한

2022.08.28

Intellij Lombok 사용하기

File | Settings | Build, Execution, Deployment | Compiler | Annotation Processors

Enable annotation processing 체크!



정적 페이지

/resource/static 폴더에 HTML을 넣어두면 실제 서비스에서도 공개된다.

뷰 템플릿 Thymeleaf

Using path variable 사용

```
<tr th:each="item : ${items}">
   <td><a href="item.html" th:href="@{/basic/items/{itemId}(itemId=${item.id})}"
```

```
                                    → Using path variable 방법 1
th:text="${item.id}">회원id</a></td>
   <td><a href="item.html" th:href="@{'/basic/items/' + ${item.id}}"
                                    → Using path variable 방법 2
 th:text="${item.itemName}">상품명</a></td>
   <td th:text="${item.price}">10000</td>
   <td th:text="${item.quantity}">10</td>
</tr>
```

HTML문서로도 열리지만 동적 랜더링 되는 순간 th: 속성으로 들어간다.

값이 없다면 속성까지 전부 생성하게된다.

핵심

타임리프는 th: 태그가 붙으면 서버사이드에서 렌더링에서 되고 기존 값을 대체한다. 없으면 html속성이 그대로 사용되는 것 HTML 파일 보기를 유지하면서 템플릿 기능도 할 수 있다.

링크표현

th:href="@{/css/xxx/xxx..}" 등으로 사용

링크표현 **2**

th:href="@{/basic/items/{itemId}(itemId=${item.id}, query='test')}" 로 쿼리파라미터를 넣을 수 있다.

||리터럴 대체 문법

타임이프에서 문자와 표현식을 따로 더해서 사용해야 하지만 리터럴 대체문법을 사용하면 편리하게 사용가능하다.

url도 변경가능

`th:href="@{'/basic/items/' + ${item.id}}"`

를 변경하면

`th:href="@{|/basic/items/${item.id}|}"`

HTML을 유지하며 서버 사이드 렌더링이 있어도**,** 없어도 잘 동작하는 것을 네츄럴 템플릿이라고 한다**.**

```
<form action="item.html" th:action="@{/basic/items/add}" method="post">
```

```
<form action="item.html" th:action method="post">
```

값이 비워져 있으면 메소드 방식을 주고 같은 url로 이동한다.

주의**! URL**을 사용할 땐 **"@{}"** 사용

@ModelAttribute("xxx")

담기는 이름 까지 지정해준다. model.Attribute("xxx", 담을 내용)

받은 내용을 전달받은곳에서 사용할때 사용한다.

- 요청 파라미터를 처리
- Model에 지정된 객체를 바로 넣어줌

두가지 기능을 해준다.

이를 생략하면 클래스명 맨앞글자를 소문자로 바꾼 이름으로 담겨서 전달된다.

```
    @PostMapping("/add")
    public String addItemV1(@RequestParam String itemName,
                            @RequestParam int price,
                            @RequestParam Integer quantity,
                            Model model){

        Item item = new Item();
        item.setItemName(itemName);
        item.setPrice(price);
        item.setQuantity(quantity);

        itemRepository.save(item);

        model.addAttribute("item", item);
```

```
        return "basic/item";
    }
→ 그냥 사용
//    @PostMapping("/add")
    public String addItemV2(@ModelAttribute("item") Item item){

        itemRepository.save(item);

        return "basic/item";
    }
→ Model 생략 가능 , ModelAttribute 사용
    @PostMapping("/add")
    public String addItemV3(Item item){

        itemRepository.save(item);

        return "basic/item";
    }
 → ModelAttribute 까지 생략가능 명명법 Item ⇒ [“item”]
```

Redirect 문법

```
@PostMapping("/{itemId}/edit")
public String edit(@PathVariable Long itemId, @ModelAttribute Item item) {
    itemRepository.update(itemId, item);
    return "redirect:/basic/items/{itemId}";
}
```

리다이렉트를 사용하는 방법 PathVariable도 그대로 사용 가능하게 지원한다.

HTML Form 은 전송은 PUT PATCH를 지원하지 않는다. PUT, PATCH는 HTTP API 전송시에 사용한다.

PRG [POST REDIRECT GET] 패턴

상품 등록 등 post 등록후 바로 응답을 주면 포스트 요청된 url 이 남게된다. 마지막 요청한 것이

POST add 요청 → 새로고침하면 다시 Post add 요청이 되는 것이다. 데이터도 그래도 요청된다.

그러므로 redirect를 사용해서 이전 요청한 URL으로 부터 벗어나야 한다.

즉 → **POST** 요청후 **Redirect** → **GET** 요청 페이지 **PRG** 방식으로 이동시킨다**.**

재요청이 되는 것.

```
return "redirect:/basic/items/" + item.getId();
```

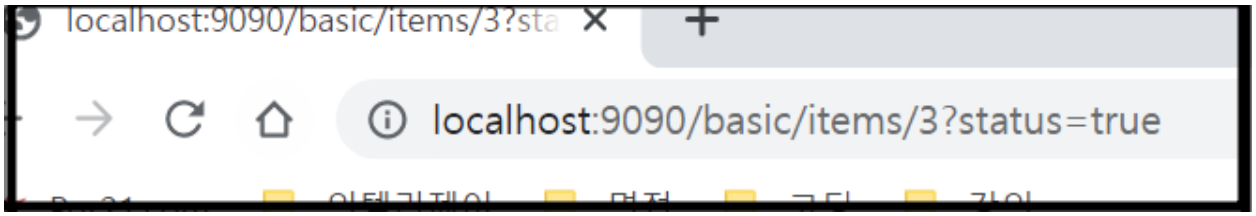
이렇게 보내게 되었을 때 item.getId() 같은 변수에 띄어쓰기나 한글이 안된다. 그러므로

리다이렉트에 여러 정보를 넣을 수있는 (인코딩 등) RedirectAttribute가 있다.

RedirectAttribute

```
@PostMapping("/add")
public String addItemV5(Item item, RedirectAttributes redirectAttributes){

   Item saveItem = itemRepository.save(item);
   redirectAttributes.addAttribute("itemId",saveItem.getId());
   redirectAttributes.addAttribute("status", true);
   return "redirect:/basic/items/{itemId}";
}
```




```
<h2 th:if="${param.status}" th:text="'저장 완료'"></h2>
```

타임 리프에서 지원하는 쿼리파라미터를 꺼내는 param.xxx 로 쿼리파라미터를 꺼낼 수 있다.