Spring MVC : Controller 파라미터 수집

생성일	@2022년 8월 18일 오후 1:13
태그	

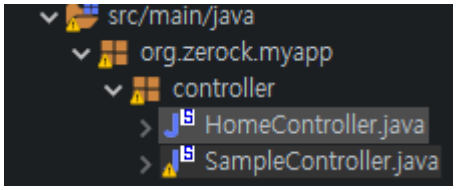
1. Controller 클래스 생성

(1) 어노테이션 @Controller 작성

-. 현재 클래스의 역할이 Controller라고 정의 하는 어노테이션이다.

(2) Servlet-context.xml에 스프링 빈 객체로 등록

```
<context:component-scan base-package="org.zerock.myapp" />
```



-. 빈 객체로 등록이 된다면 ‘s’가 붙는다. 이는 스프링에서 관리한다는 의미이다.

(2) Base URI를 지정하는 어노테이션 @RequestMapping 작성

-. RequestMapping 값으로 “/board/”로 지정 (클래스와 메소드에만 선언가능)

→ 모든 메소드들의 기본 경로가 된다. 즉! Client로 부터 요청이 들어오는 URL은 http://localhost:8080/board/*가 된다.

```
@Log4j2
@NoArgsConstructor
```

```
@RequestMapping("/board/")

@Controller
public class SampleController {

} // end class
```

2. 어노테이션 @RequestMapping

-. 컨트롤러의 선언 되는 메소드는 컨트롤러의 “핸들러(Handler)” 이다. 이 메소드들은 **Request을 처리하는 핸들러**라는 의미이다.

-. RequestMapping 어노테이션을 사용하여, **어떤 전송방식(HTTP Method)과 어떤 Request URI**를 가지고 들어온 Request를 처리하도록 한다.

-. method 속성을 생략하면 어떤 전송방식이든지 받는다는 의미이다.

-. RequestMapping의 path 속성은 /doA와 /doAA로 지정되어있다. 현재 BaseURI가 /board/이므로 요청이 들어오는 **RequestURI는 /board/doA 혹은 /board/doAA만 받겠다**는 의미이다.

-. method 속성으로는 Get방식과 POST 방식으로 설정되어, 이 두 전송방식(HTTP Method)만 받겠다는 의미이다.

```
@RequestMapping(
      path= {"/doA", "/doAA"},
      method= {RequestMethod.GET, RequestMethod.POST}
  )
  // method 속성을 제외시키면, 모든 전송방식을 받는다.
  public void doA() {
    log.trace("doA() invoked.");

  } // doA
```

1. @RequestMapping(””)

-. RequestMapping에서 “”의 의미는 * 이다. 그냥 @RequestMapping으로 축약해서 사용할 수 있다.

-. /board/* 로 들어오는 요청을 모두 받겠다는 의미이다.

```
//  @RequestMapping("")
  @RequestMapping
  public String basic() {
    log.trace("basic() invoked.");

    return "sample";
  } // basic
```

2. @GetMapping / @PostMapping

-. GET / POST방식으로만, 받을 수 있다.

```
@GetMapping("/basicOnlyGet")
  public String basicOnlyGet() {
    log.trace("basicOnlyGet() invoked.");

    return "sample";
  } // basicOnlyGet

@PostMapping("/basicOnlyPost")
  public String basicOnlyPost() {
    log.trace("basicOnlyPost() invoked.");

    return "sample";
  } // basicOnlyPost
```

3. Controller 의 메소드 파라미터 수집

-. Controller 메소드에서 파라미터를 수집할 때, DTO객체에 선언 된, 속성을 매개변수로 전달 할 수도 있지만, DTO 객체 자체를 수집할 수도 있다.

```
@GetMapping("ex01")
  public String ex01(String name, Integer age) {
```

```
    log.trace("ex01() invoked.");

    // servlet-context.xml에 지정 된, view-resolvers 의해서
    // prefix + viewname + suffix = /WEB-INF/views/ + ex01 + .jsp = /WEB-INF/views/ex0
1.jsp
    // MVC패턴에서, View의 이름
    return "ex01";
  } // ex01

  @GetMapping("ex02")
  public String ex02 (SampleDTO dto) {
    log.trace("ex02({}) invoked.", dto);

    return "ex02";
  } // ex02
```

http://localhost:8080/board/ex01?name=Ryan&age=23

-. parameter 변수명과 동일한 이름으로 전송되어야 한다.

http://localhost:8080/board/ex02?name=Ryan&age=23

-. parameter 전송 시, SampleDTO라는 객체안에, name과 age 속성이 수집되어있다.

```
o.z.m.c.SampleController.ex01:45 - ex01() invoked.
o.z.m.c.SampleController.ex01:47 - name : Ryan, age : 23
o.z.m.c.SampleController.ex02:57 - ex02(SampleDTO(name=Ryan, age=23)) invoked.
```

-. 어노테이션 **@RequestParam**을 사용하면, 매개변수명과 전송되는 파라미터의 이름이 달라도 수집이 가능하다.

```
@GetMapping("ex03")
  public String ex03 (
      @RequestParam("name")String name2,
      @RequestParam("age")Integer age2) {
    log.trace("ex03({}, {}) invoked.",name2, age2);
    return "ex03";
  } // ex03
```

2. 동일한 이름의 파라미터 전달 (ArrayList<>를 이용한 객체 수집방법)

@RequestParam과 ArrayList를 사용하여 출력

```
@GetMapping("/ex04")
  public String ex04 (
      @RequestParam("name") ArrayList<String> name
  ) {
    log.trace("ex04() invoked.");

    name.forEach(log::info);

    return "ex04";
  }
```

3. 날짜형식의 파라미터 수집방법 (@DateTimeFormat)

@DateTimeFormat 어노테이션을 이용한 데이터 형식 파라미터 수집

```
@GetMapping("/ex05")
  public String ex05 (
      @RequestParam("date")
      @DateTimeFormat(pattern = "yyyy/MM/dd")
      Date hireDate
  ) {
    log.trace("ex05() invoked.");

    log.info("hireDate : {}", hireDate);

    return "ex05";
  } // ex05
```

4. 데이터 전달자 MODEL

1. Model 객체란?

Model 객체 : JSP에 컨트롤러에서 생성 된 데이터를 담아서 전달하는 역할

(Servlet의 request.setAttribute()와 유사한 역할)

2. 왜? Model 객체를 사용해햐하는가?

-. 파라미터들에 대한 처리 후 결과를 전달

-. 기본형 타입의 파라미터들을 전달받아, View로 전달해야할 때

→ 기본형타입의 파라미터는 JSP로 전달되지 못한다. (Java Beans규칙에 맞는 객체만 전달)

3. 어노테이션 @ModelAttribute

-. 스프링MVC 컨트롤러는 Java Beans 규칙에 맞는 객체는 다시 화면으로 전달한다.

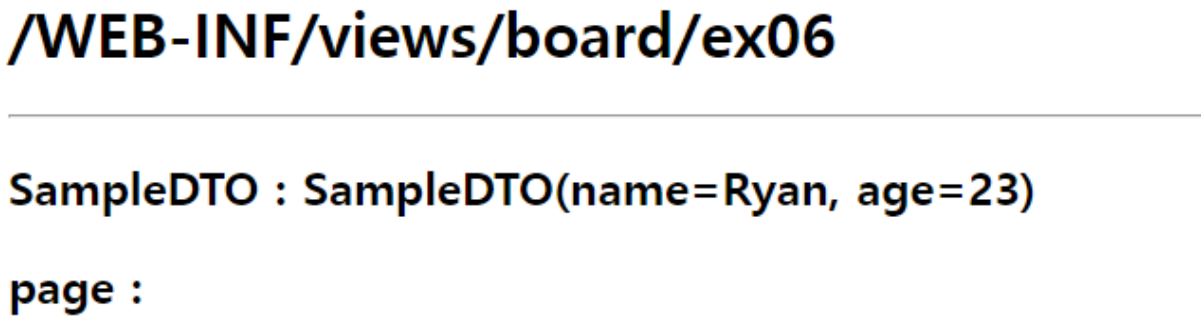
→ 전달할 때, 클래스명의 앞글자는 소문자로 처리된다.

-. Java Beans 규칙 : (1) Default 생성자, (2) getter/setter 메소드 존재하는 클래스의 객체

```
@GetMapping("/ex06")
  public String ex06 (
      SampleDTO dto, int page
  ) {
    log.trace("ex06({},{}) invoked.", dto, page);


    return "/board/ex06";
  } // ex06
```

-.http://localhost:8080/board/ex06?name=Ryan&age=23&page=10 전달 시, JSP로 전달 된 “page”값은 null 값으로 전달된다.



-. Model 객체를 파라미터로 전달하여, Model에 담아서 JSP로 전달

-. model.addAttribute( ) 를 이용하여 JSP로 전달한다.

```
@GetMapping("/ex07")
  public String ex07 (
      SampleDTO dto, int page, Model model
  ) {
    log.trace("ex06({},{}) invoked.", dto, page);

    model.addAttribute("page", page);

    return "/board/ex06";
  } // ex06
```

-. @ModelAttribute 어노테이션을 사용하여 JSP로 DATA 전달

```
@GetMapping("/ex08")
  public String ex08 (
      SampleDTO dto, @ModelAttribute("page")int page
  ) {
    log.trace("ex08({},{}) invoked.", dto, page);

    return "/board/ex06";
  } // ex08
```

5. 1회성 DATA 전달 : RedirectAttrubutes

-. /board/ex09 에서 받아서 1회성 DATA로 저장해서, board/ex10 으로 Redirect 한다.

-. /board/ex10에서는 매개변수로 받아, JSP 뷰를 리턴한다.

-. JSP뷰에서는 이미 /board/ex09에서 /board/ex10으로 1회성 Data로 전달되었기 때문에 DATA는 모두 사라졌다. 따라서, /board/ex10에서 뷰로 전달되는 데이터는 모두 Null값으로 나온다.

/WEB-INF/views/board/ex06

SampleDTO : SampleDTO(name=null, age=null)

page :

```
@GetMapping("/ex09")
  public String ex09 (SampleDTO dto, Integer page, RedirectAttributes rttrs) {
    log.trace("ex09() invoked.");

    rttrs.addFlashAttribute("name", dto.getName());
    rttrs.addFlashAttribute("age", dto.getAge());
    rttrs.addFlashAttribute("page", page);

    return"redirect:/board/ex10";
  }

  @GetMapping("/ex10")
  public String ex10 (SampleDTO dto, Integer page, RedirectAttributes rttrs) {
    log.trace("ex09() invoked.");


    return "/board/ex06";
  }
```

-. RedirecttAttributes의 addAttribute는 GET방식의 Query String 형태로 전달된다.

```
@GetMapping("/ex09")
  public String ex09 (SampleDTO dto, Integer page, RedirectAttributes rttrs) {
    log.trace("ex09() invoked.");

//    rttrs.addFlashAttribute("name", dto.getName());
//    rttrs.addFlashAttribute("age", dto.getAge());
//    rttrs.addFlashAttribute("page", page);

    rttrs.addAttribute("name", dto.getName());
    rttrs.addAttribute("age", dto.getAge());
    rttrs.addAttribute("page", page);

    return"redirect:/board/ex10";
  }

  @GetMapping("/ex10")
```

```
public String ex10 (SampleDTO dto, @ModelAttribute("page")Integer page) {
  log.trace("ex10() invoked.");


  return "/board/ex06";
}
```