프리젠테이션 계층 (Controller)

생성일	@2022년 8월 24일 오후 3:05
태그	

BoardController 분석

항상 Controller를 작성하기 고려해야한다.

(1) Mapping URL을 어디로 할지

(2) HTTP Method방식

(3) Parameter는 무엇으로 받아야 할지

(4) 어떤 Form (JSP Page)을 필요로 하는지

(5) 요청 후, 어디로 이동을 할지

Task	URL	Method	Parameter	From	URL 이동
전체 목록 조회	/board/list	GET		list화면 필요	
게시물 등록	/board/register	POST	BoardDTO	register화면 필요	list로 이동
게시물 상세조회	/board/get	GET	BoardDTO	get화면 필요	
게시물 삭제	/board/remove	POST	BoardDTO		list로 이동
게시물 수정	/board/modify	POST	BoardDTO	modify 화면 필요	list로 이동

BoardController 작성

Controller Bean 등록

-. 어노테이션 @Controller를 이용하여, Bean으로 등록

-. servlet-context.xml에는 해당 package를 스캔해야 한다.)

Base URI 등록

-. 어노테이션 @RequestMapping을 이용하여, “/board”등록

Service 객체 의존성 주입

-. BoardController는 BoardService 타입의 객체가 필요

-. Setter를 통한 의존성 주입방법 / 생성자를 통한 의존성 주입방법 으로 의존성 주입을 한다.

```
@Log4j2

@RequestMapping("/board")
@Controller
public class BoardController {


  //-- 생성자를 이용한 BoardService 주입
  private final BoardService service;

  @Autowired
  public BoardController(BoardService service) {
    this.service = service;
  } // Constructor

  //어노테이션 사용 시, 아래와 같이 작성 가능하다.
  /*
   * @AllArgsConstuctor 선언하기
   *
   * 필드에 주입받고자 하는 객체를 1개만 적으면, Spring 4.3부터는 자동으로
   * 의존성을 주입해준다. (별도의 @Autowired 어노테이션 사용 X )
   * private final BoardService service
   *
   */



} // end class
```

1. 게시글 전체목록 조회 (”/board/list”)

View name = Request URI + Base URI + DetailURI

-. /board/ + /list = /board/list

-. Controller의 핸들러의 리턴 값이 DetailedURI 이다.

View Resolver = Prefix(/WEB-INF/views/) + View name + Suffix(.jsp)

```
@GetMapping("/list")
  public void list(Model model) throws ControllerException {
    log.trace("list() invoked.");

    try {
      List<BoardVO> list = new ArrayList<>();

      list = service.getList();
      list.forEach(log::info);

      model.addAttribute("list", list);

    } catch (Exception e) {
```

```
      throw new ControllerException(e);
    } // try-catch

  } // list
```

2. 게시글을 등록 (”/board/register”)

게시물을 등록하는 화면에서 등록버튼을 눌렀을 때, 발생되는 이벤트이다.

게시물을 등록하고나면, 게시판화면으로 이동(Redirect)하게 되며, 게시글이 잘 등록되었는지, 확인되는 문구가 뜨게 한다.

POST방식으로 전달이 필요하며, Redirect하게 되면 일반적으로 Request Scope에 저장된 Data는 모두 날라가게 된다. 그러기 때문에, RedirectAttributes라는 Redirect해도 공유객체가 전달 될 수 있도록 한다.

RedirectAttributes에는 2가지 저장방식이 있다.

-. (1) addflashAttribute( ) : (1회성)Request Scope에 바인딩하여 전달한다.

-. (2) addAttribute( ) : Get방식의 전송파라미터로, Query String 형식으로 보낸다.

```
@PostMapping("/register")
  public String register(BoardDTO dto, RedirectAttributes rttrs) throws ControllerException {
    log.trace("register() invoked.");

    try {

      boolean isRegister = this.service.register(dto);
      log.info("\t+ isRegister", isRegister);

      rttrs.addAttribute("result", isRegister ? "SUCCESS(" + dto.getBno() + ")" : "FAILURE");

      return "redirect:/board/list";

    } catch (Exception e) {
      throw new ControllerException(e);
    }

  } // register
```

3. 게시글을 수정 (”/board/register”)

게시글 수정도 마찬가지로, 수정하는 화면에서 수정하기버튼을 눌렀을 때, 발생되어야 한다.

수정을 하면, 게시글 리스트 화면으로 Redirect하게 된다.

수정하는 화면은 GET방식으로 접근하지만, 수정자체를 하는 작업은 POST방식으로 동작한다.

-. 게시글 등록 / 게시글 수정 / 게시글 삭제는 모두 POST 방식으로 동작한다.

-. 각각의 화면은 GET방식으로 접근한다.

```
@PostMapping("/modify")
  public String modify(BoardDTO dto, RedirectAttributes rttrs)
    throws ControllerException {
    log.trace("modify() invoked.");

    try {
      boolean isModify = service.modify(dto);

      rttrs.addFlashAttribute("result", isModify ? "SUCCESS(" + dto.getBno() + ")" : "FAILURE");

      return "redirect:/board/list";

    } catch (Exception e) {
      throw new ControllerException(e);
    } // try-catch

  } // modify
```

4. 게시글을 삭제 (”/board/remove”)

게시글에서 삭제버튼을 눌렀을 때, 발생한다.

삭제 후, 게시글 리스트로 Redirect 된다.

```
@PostMapping("/remove")
  public String remove(BoardDTO dto, RedirectAttributes rttrs)
    throws ControllerException {
    log.trace("remove() invoked.");

    try {
      boolean isRemove = service.remove(dto);

      rttrs.addAttribute("result", isRemove ? "SUCCESS(" + dto.getBno() + ")" : "FAILURE");

      return "redirect:/board/list";

    } catch (Exception e) {
      throw new ControllerException(e);
    } // try-catch

  } // remove
```

5. 게시글을 상세조회 (”/board/get”)

선택한 게시글 1개를 조회한다.

```
@GetMapping("/get")
  public void get(BoardDTO dto, Model model) throws ControllerException {
    log.trace("get() invoked.");

    try {
      BoardVO vo = service.get(dto);
      log.info("\t+ vo : {}", vo);

      model.addAttribute("vo", vo);

    } catch (Exception e) {
      throw new ControllerException(e);
    } // try-catch

  } // get
```