스프링 - 게시판 만들기 04

Controller

Controller 단위 테스트

```
@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
@ContextConfiguration({
	"file:src/main/webapp/WEB-INF/spring/root-context.xml",
	"file:src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml"
})
@Log4j
public class BoardControllerTests {
	@Autowired
	private WebApplicationContext webApplicationContext;

	private MockMvc mockMvc;

	@Before
	public void setUp() {
		this.mockMvc =
MockMvcBuilders.webAppContextSetup(webApplicationContext).build();

	}

//	@Test
//	public void listTest() throws Exception {
//
log.info(mockMvc.perform(MockMvcRequestBuilders.get("/board/list")).andRetu
rn().getModelAndView().getModelMap());
//
//	}

	@Test
	public void registerTest() throws Exception{


log.info(mockMvc.perform(MockMvcRequestBuilders.post("/board/register")
```

```
                        .param("title", "테스트 새 글 제목")
                        .param("content", "테스트 새글 내용")
                        .param("writer","진호")
                        ).andReturn().getFlashMap());

        }
}
```

Controller에서 **Redirect** 방법

```
@PostMapping("/register")
public String register(BoardVO boardVO, RedirectAttributes rttr) {
        log.info("/register" + boardVO);
        boardService.register(boardVO);

        //model.addAttribute("bon", boardVO.getBno());
        //Flash 영역은 Session에 생기고 redirect 로 전송할 때
        //redirect로 전송할 때 request영역이 초기화 되고
        //초기화 되기전에 Flas영역에 데이터를 저장해 놓고 , 초기화된 후에
        //Flash 영역에서 데이터를 가지고 온다. 데이터를 가져왔따면,Flash영역에
있던
        //데이터는 없어진다.
        rttr.addFlashAttribute("bno",boardVO.getBno());

        //redirect 를 전송하면 사용을 못한다. 다른방법사용
        //방법 1. Session 이용
        //Session 에 flash 영역에 담아둔다 .
        //방법 2 쿼리 스트링에 사용
        return "redirect:/board/list";
}
```

RedirectAttributes rttr
객체를 사용해서 Session flash 영역에 Attribute 할 수 있다.
-Model객체 대신.

페이징 처리

화면 페이징 처리

ORDER BY 보다 인덱스를 사용하면 정렬을 생략할 수 있다.

인덱스라는 존재가 이미 정렬된 구조이므로 이를 이용해서 별도로 정렬을 하지 않는다.

예를 들어 데이터베이스 테이블을 하나의 책이라고 가정하면, 인덱스는 각 페이지 번호를

의미한다. 이를 통해서 원하는 내용을 위에서부터 혹은 반대로 찾아나가는 것을

스캔(scan) 한다고 표현한다.

데이터베이스에 테이블을 만들 때 PK를 부여하면 PK 컬럼을 기준으로 인덱스가 생성되고

실제 테이블의 데이터가 어디에 저장되어 있는 지를 찾을 수 있는 KEY 값이다. 실제

테이블의 데이터가 저장된 각 행에는 ROWID 라는 것이 존재하고 데이터베이스 내의

주소값을 의미한다.

BNO ROWID => 테이블에 숨어있다.

1 AAAE7A -ROWID BNO TITLE CONTENT

2 AAAE7B AAAE7B 2 aaa aaaa 이런식으로

3 AAAE7C AAAE7E 1000 bbb bbbb

— ..... AAAE7C 3 sdaf dfasd

....

1000 AAAE7E

테이블엔 정렬은 안되어있다.

—----------

힌트(Hint) : /*+ */ [힌트 문법]

SELECT문에 실행하고 싶을 계획을 전달할 때 사용하는 문법.

잘못 작성되어도 실행할 때에는 무시되며 별도의 오류는 발생하지 않는다.

/*+ 로 시작되며 */로 마친다. 또한 뒤에 컬럼명을 작성할 때, 를 사용하지 않는다.

페이징 처리에서는 이미 정렬된 인덱스를 내림차순으로 가지고 오는 문법을 힌트 내에서 작성해준다. 별도의 정렬이 필요 없어지므로 문법의 가독성이 좋아진다.

SELECT /*+ INDEX_DESC(TBL_BOARD2, SYS_C007197) */

BNO, TITLE, CONTENT, WRITER, REGDATE, UPDATEDATE FROM TBL_BOARD2;

/*+ INDEX_DESC(TBL_BOARD2, SYS_C007197) */ ⇒ 컨스트레인트 이름