└ Controller JUnit5 TEST (WAS 미구동)

생성일	@2022년 8월 25일 오전 4:47
태그	

1. WAS 없이, TEST할 수 있는 환경설정

1. 어노테이션 @WebAppconfiguration

    -. WebApplicationContext 존재를 이용하기 위해 사용

    -. Spring Beans Container

2. Spring Beans Conatiner 주입

    -. Spring에 주입 된 빈 객체를 이용하기 위해, BoardController가 아닌, WebApplicationContext를 주입받는

```
@Log4j2
@NoArgsConstructor

//Spring Beans Container & DI 수행시키는 어노테이션
@ExtendWith(SpringExtension.class)
@ContextConfiguration(locations = { "file:src/main/webapp/WEB-INF/**/*-context.xml" })

//-- Spring MVC Framework 구동시키는 어노테이션
//-- WebApplicationContext 존재를 이용하기 위해 사용
@WebAppConfiguration

@TestInstance(Lifecycle.PER_CLASS)
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class BoardControllerTests {

  //-- Springs Beans Container(type : WebApplicationContext) 객체를 주입
  @Setter(onMethod_ = {@Autowired})
  private WebApplicationContext ctx;
```

```
} // end class
```

2. TEST메소드 작성 (게시물 전체 조회)

1. MockMvc 객체를 생성하는 Builder 생성

-. WebApplicationContext를 가지고 있는 MockMvcBuilder를 생성해야하기 때문에, 앞에서 주입받은 Beans Container를 전달한다.

2. MockMvc 생성

-. 가짜 MVC로, 가짜URL과 파라미터를 브라우저에서 보내는 것처럼 Controller에 전달

3. Controller에 보낼 Request 작성 (전송파라미터가 없는 경우)

4. Controller에 요청 보내기

5. Controller에 요청하여, 발생한 결과물을 얻는다.

6. 발생한 결과물을 통해, Model과 View를 가지고 있는 객체 생성

7. Model과 View를 가지고 있는 객체로 부터, 2가지 정보 획득

-. ViewName 과 model

```
//  @Disabled
  @Test
  @Order(1)
  @DisplayName("1. BoardController.list")
  @Timeout(value=3, unit = TimeUnit.SECONDS)
  void testList() throws Exception {
    log.trace("testList() invoked.");

    //-- Step.1
    //-- MockMvc 객체를 생성하는 Builder 생성
    //-- 어떤 MockMvcBuilder냐? : 앞서 생성한 WebApplicationContext를 이용하는
    MockMvcBuilder mockMvcBuilder = MockMvcBuilders.webAppContextSetup(ctx);

    //-- Step.2 : MovcMvc 객체 생성
    MockMvc mockMvc = mockMvcBuilder.build();

    //-- Step.3 : Controller에 보낼 Request 생성
```

```
    //-- (1) : 전송파라미터가 없는 경우!
    RequestBuilder reqBuilder = MockMvcRequestBuilders.get("/board/list");

    //--(2) : 전송파라미터가 있는 경우! <예시>
//    MockHttpServletRequestBuilder reqBuilder = MockMvcRequestBuilders.get("/board/li
st");

//    reqBuilder.param("name", "Ryan");

    //-- Step.4 : Controller에 요청 보내기
    ResultActions resultActions = mockMvc.perform(reqBuilder);

    //-- Step.5 : Step.4에서 발생한 결과물을 얻는다.
    MvcResult mvcResult = resultActions.andReturn();

    //-- Step.6 : Step.5에서 얻어낸 결과물을 통해 ModelAndView 객체얻는다.
    //-- ModelAndView는 Model과 View이름을 가지고 있다.
    ModelAndView modelAndView = mvcResult.getModelAndView();

    //-- Step.7 : ModelAndView객체로부터, 아래의 2가지 정보를 획득한다.
    //-- (1) 반환된 뷰 이름
    String viewName = modelAndView.getViewName();

    //-- (2) 반환된 비지니스 데이터 (즉, Model 객체)
    ModelMap model = modelAndView.getModelMap();

    log.info("\t+ viewName : {}", viewName);
    log.info("\t+ model : {}", model);

  } // testList
```

3. TEST메소드 작성 (게시물 등록)

파라미터(BoardDTO) 를 전달해야하므로 Step.3에서 약간 변경된다.

-. register는 POST방식으로 보내야 한다.

게시물 등록에서 보내야할 파라미터는 다음과 같다.

-. title

-. content

-. writer

Fluent-API기반 Method Chaining을 이용하여, 코드 작성

```
@Test
@Order(2)
@DisplayName("2. BoardController.register")
@Timeout(value=3, unit = TimeUnit.SECONDS)
void testRegister() throws Exception {
  log.trace("testRegister() invoked.");

  //-- MockMvc 객체 생성
  MockMvc mockMvc = MockMvcBuilders
              .webAppContextSetup(ctx)
              .build();

  //-- 주의 : register는 POST 방식으로 전송되어야 한다.
  MockHttpServletRequestBuilder reqBuilder = MockMvcRequestBuilders.post("/board/regis
ter");

  //-- 전송파라미터 생성
  reqBuilder.param("title", "NEW_TITLE");
  reqBuilder.param("content", "NEW_CONTENT");
  reqBuilder.param("writer", "NEW_WRITER");

  //-- Controller로 요청(Request) 보내면서, ModelAndView 객체 생성
  ModelAndView modelAndView = mockMvc
                  .perform(reqBuilder)
                  .andReturn()
                  .getModelAndView();

  log.info("\t+ modelAndView : {}", modelAndView);


} // testRegister
```

4. TEST메소드 작성 (게시물 수정)

```
@Test
@Order(3)
@DisplayName("3. BoardController.modify")
@Timeout(value=3, unit = TimeUnit.SECONDS)
void testModify() throws Exception {
  log.trace("testModify() invoked.");

  //-- MockMvc객체 생성
  MockMvc mockMvc = MockMvcBuilders
              .webAppContextSetup(ctx)
              .build();
```

```
  //-- modify에 Request할 객체 생성
  MockHttpServletRequestBuilder reqBuilder = MockMvcRequestBuilders.post("/board/modif
y");

  //-- 전송파라미터 생성
  reqBuilder.param("bno", "70");
  reqBuilder.param("title", "UPDATE_TITLE");
  reqBuilder.param("content", "UPDATE_CONTENT");
  reqBuilder.param("writer", "UPDATE_WRITER");

  //-- ModelAndView 객체 생성
  ModelAndView modelAndView = mockMvc
                  .perform(reqBuilder)
                  .andReturn()
                  .getModelAndView();

  log.info("\t+ modelAndView : {}", modelAndView);

} // testModify
```

5. TEST메소드 작성 (게시물 삭제)

```
@Test
@Order(4)
@DisplayName("4. BoardController.remove")
@Timeout(value=3, unit = TimeUnit.SECONDS)
void testRemove() throws Exception {
  log.trace("testRemove() invoked.");

  //-- MockMvc객체 생성
  MockMvc mockMvc = MockMvcBuilders
                .webAppContextSetup(ctx)
                .build();

  //-- remove에 Request할 객체 생성 (POST)
  MockHttpServletRequestBuilder reqBuilder = MockMvcRequestBuilders.post("/board/remov
e");

  //-- 전송파라미터 작성
  reqBuilder.param("bno", "51");

  //-- ModelAndView 객체 생성
  ModelAndView modelAndView = mockMvc
                  .perform(reqBuilder)
                  .andReturn()
                  .getModelAndView();

  log.info("\t+ modelAndView : {}", modelAndView);
```

```
} //testRemove
```

6. TEST메소드 작성 (게시물 상세조회)

```
@Test
@Order(5)
@DisplayName("5. BoardController.get")
@Timeout(value=3, unit=TimeUnit.SECONDS)
void testGet() throws Exception {
  log.trace("testGet() invoked.");


  //-- MockMvc객체 생성
  MockMvc mockMvc = MockMvcBuilders
              .webAppContextSetup(ctx)
              .build();

  //-- get에 Request할 객체 생성 (GET)
  MockHttpServletRequestBuilder reqBuilder = MockMvcRequestBuilders.get("/board/get");

  //-- 전송파라미터 작성
  reqBuilder.param("bno", "53");

  //-- ModelAndView 객체 생성
  ModelAndView modelAndView = mockMvc
                  .perform(reqBuilder)
                  .andReturn()
                  .getModelAndView();

  log.info("\t+ modelAndView : {}", modelAndView);

} // testGet
```