**Software Project**

**Project #2**

**담당교수: 최강임 교수님**

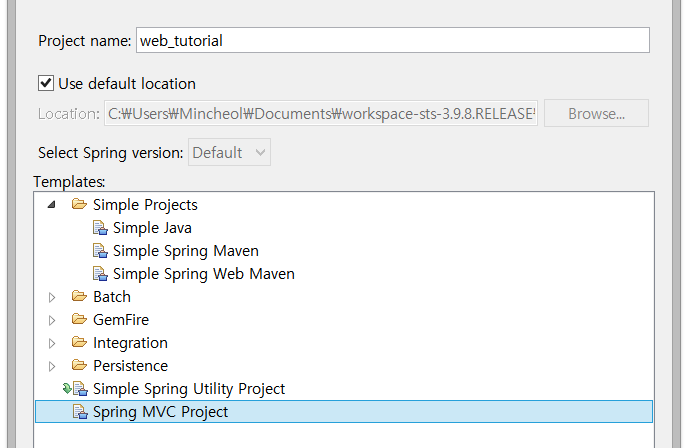
**제출일: 2019. 05. 26.**

**학과: 컴퓨터정보공학부**

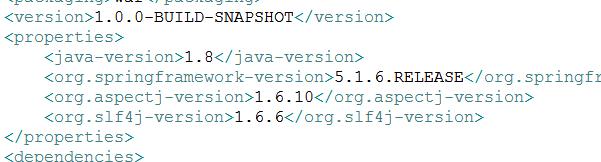
**학번 : 2015722087**

**이름 : 김민철**

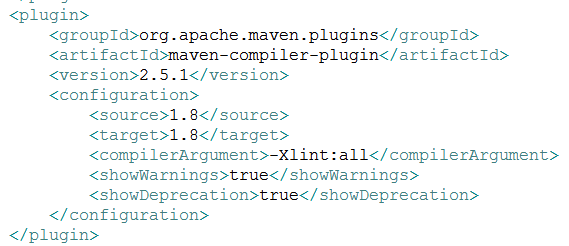
1. **튜토리얼 과정에 대한 설명 및 과정 캡쳐**



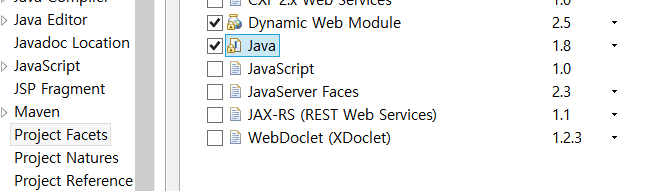
프로젝트를 생성하는 과정이다. 스프링 MVP 프로젝트로 생성하며, 프로젝트의 이름은 web\_tutorial으로 생성하였다.



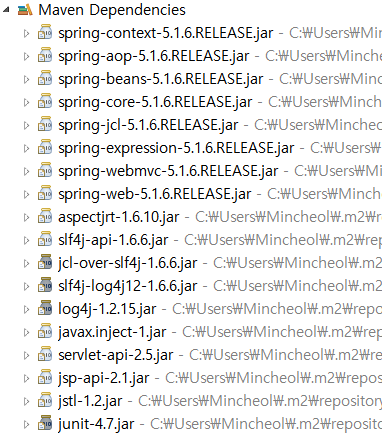
이번 프로젝트에서 사용하는 자바와 springframework의 버전을 변경해주었다. 각각 1.8과 5.1.6 버전으로 pom.xml 파일(설정 파일)에서 변경하였다.



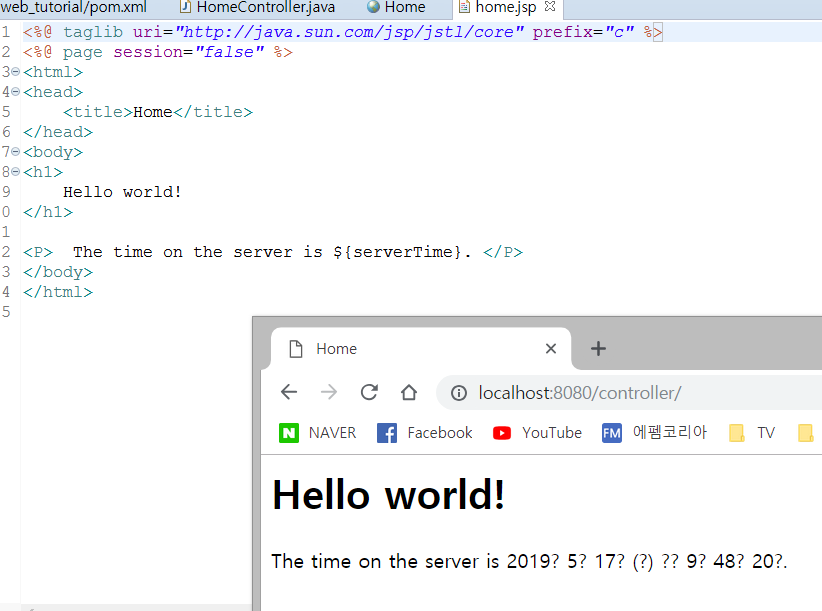
마찬가지로 사용할 자바의 버전을 변경해주었다.



STS의 프로젝트 설정에서 자바의 버전을 1.8로 변경해주었다.



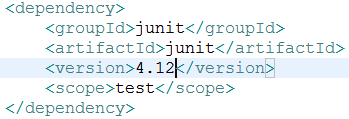
Maven으로 추가된 라이브러리들을 확인하였다. pom.xml 설정에서 변경한 5.1.6 RELEASE 버전의 라이브러리를 확인할 수 있다.



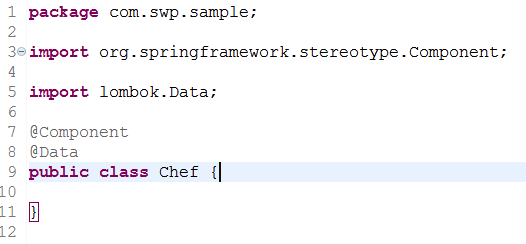
Run on server을 이용하여 기본적으로 생성되어 있는 코드의 결과를 출력하였다. home.jsp 파일에서 작성된 html의 출력을 확인할 수 있다.



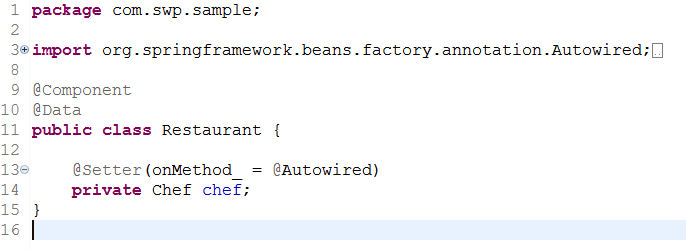
lombok을 설치하고 pom.xml에 추가하였다. 본인의 lombok은 1.18.8 버전이다.



Junit의 버전을 4.12로 변경하였다.



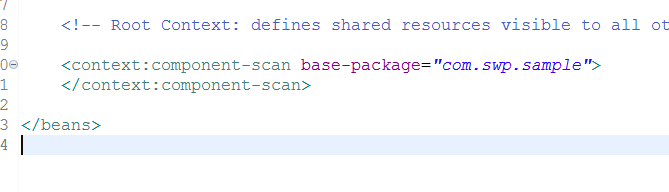
Chef 클래스를 작성하였다.(테스트용)



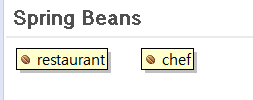
Restaurant 클래스를 작성하였다.(테스트용)



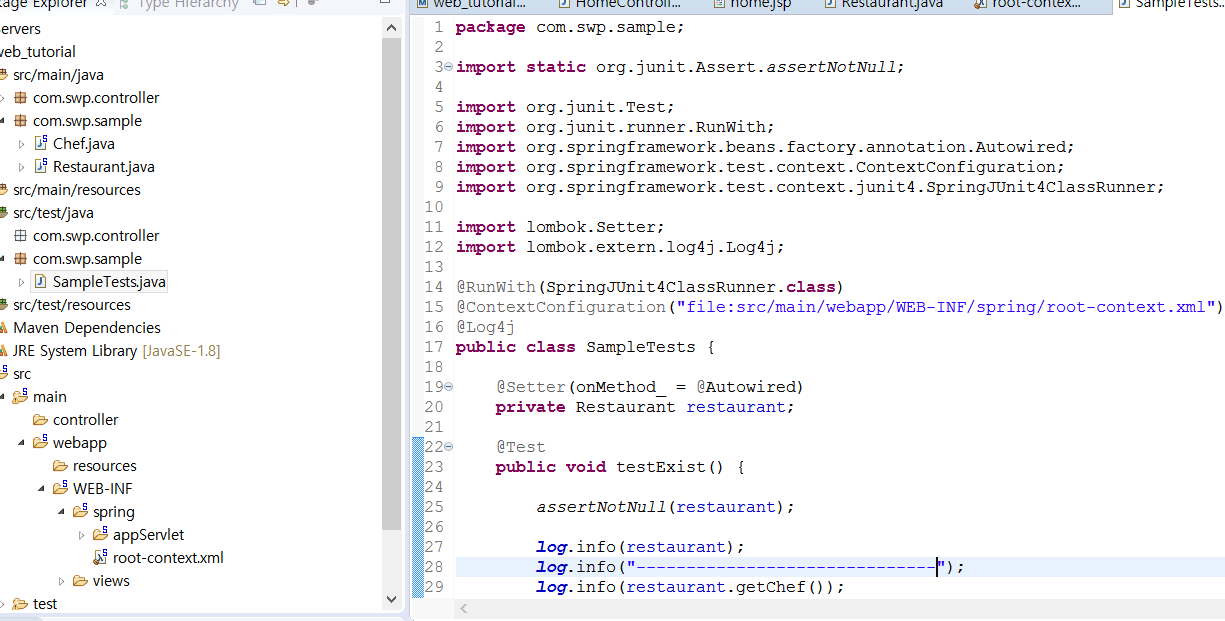
root-context.xml의 Namespace에서 context를 체크하였다.



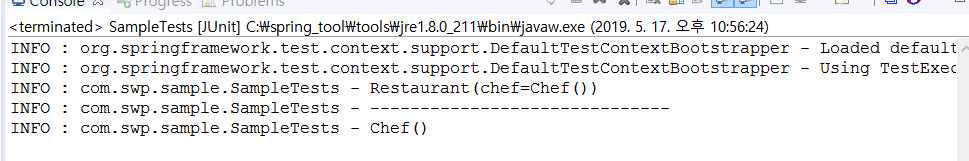
context 코드를 추가 작성하였다. com.swp.sample 패키지가 base 패키지이다.



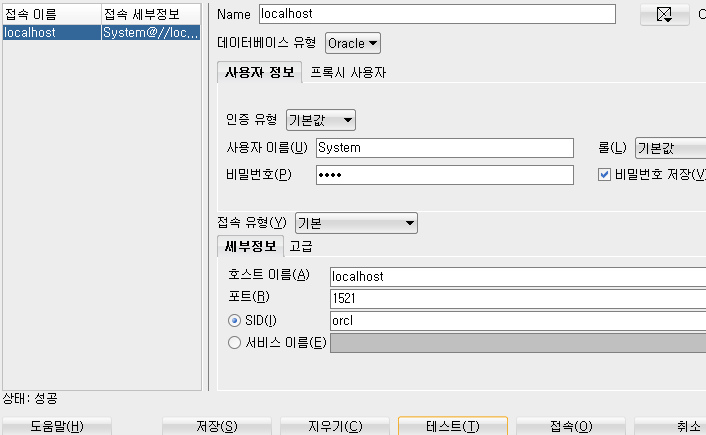
Bean Graph에 restaurant와 chef 객체가 설정된 것을 확인하였다.



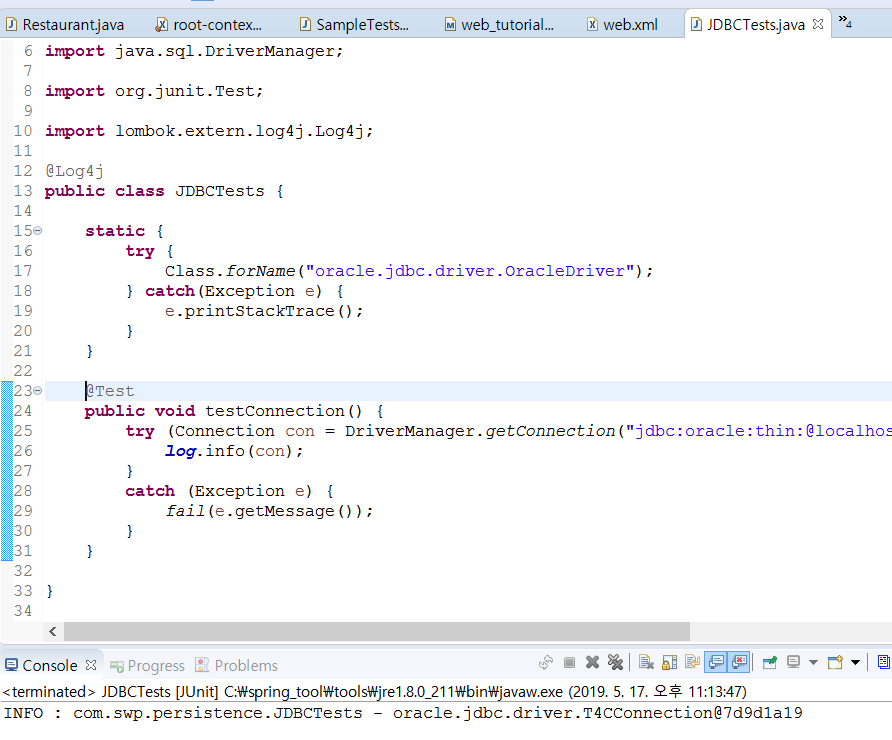
SampleTests 클래스를 작성하였다. - Junit을 이용한 코드의 테스트를 위함이다.



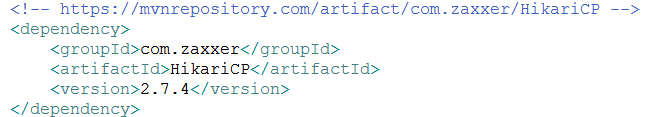
Run As - Junit test를 한 결과이다. 두 객체가 모두 출력되었다.



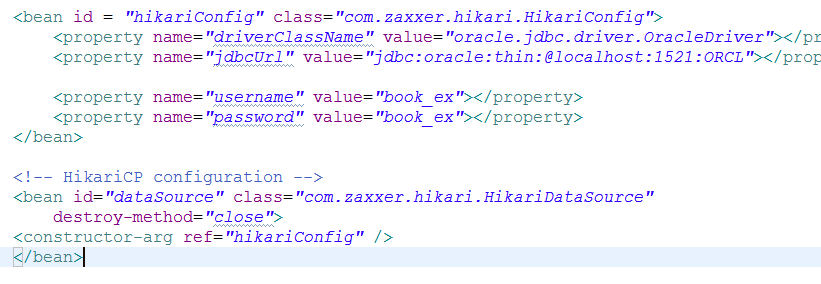
sql developer를 통해 DB에 접속 테스트를 하였으며, 접속하여 사용자를 생성하고 사용자 BOOK\_EX에게 권한을 부여하였다.



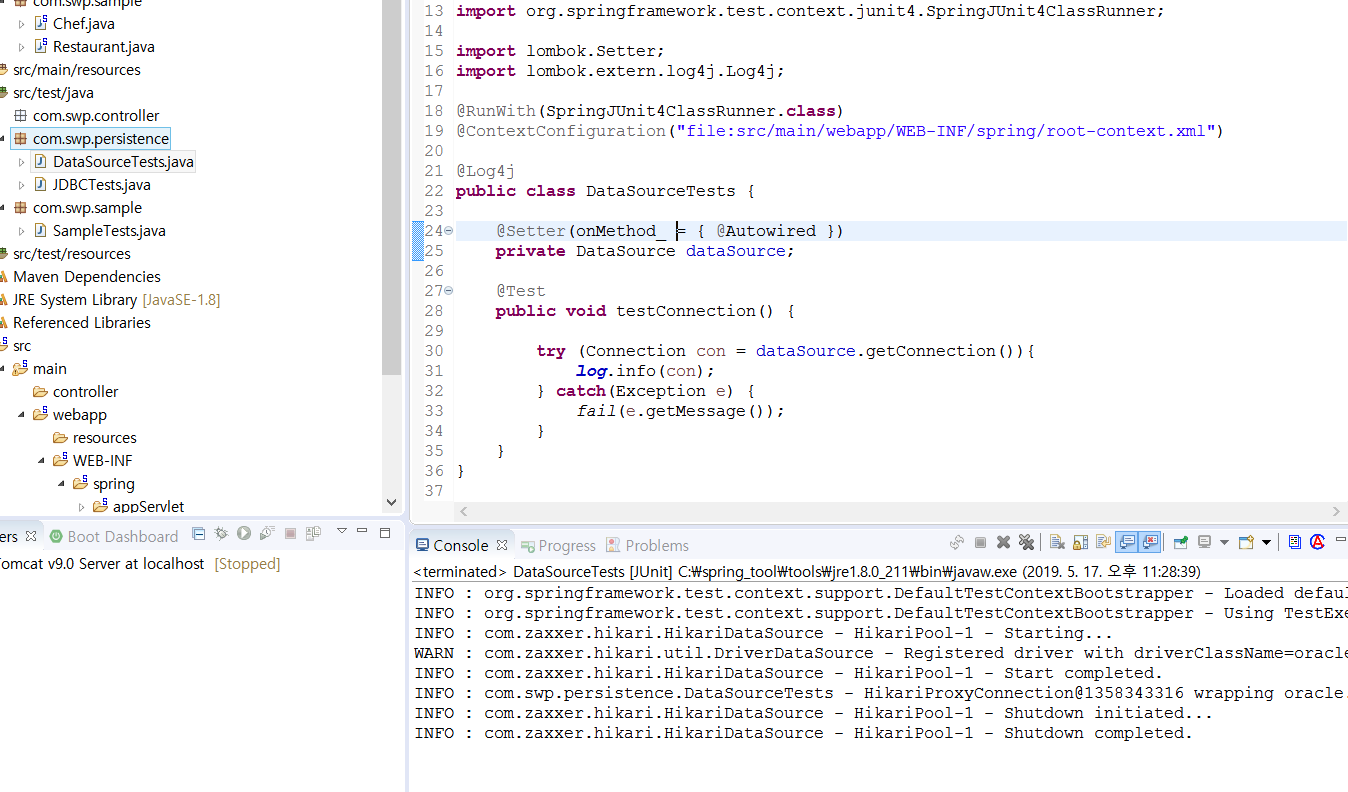
JDBCTests 클래스를 작성하여 테스트하였다. - Junit을 이용한 코드의 테스트



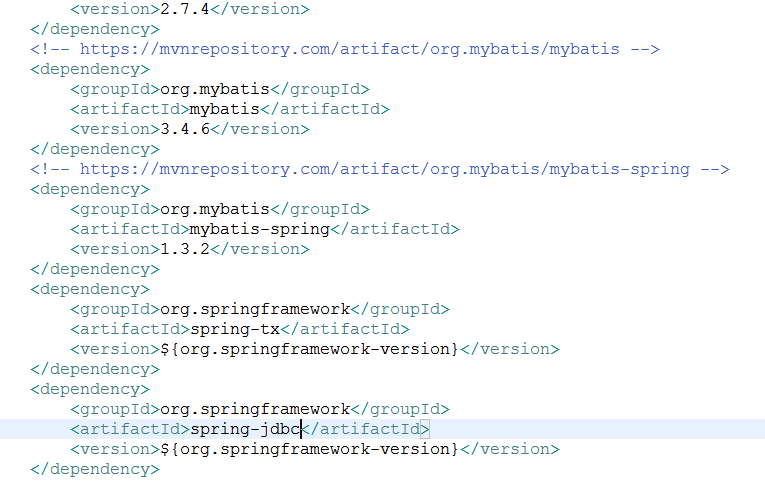
커넥션 풀 설정을 위해 pom.xml 에 HikariCP를 추가한 모습이다. 버전은 2.7.4 버전이다.



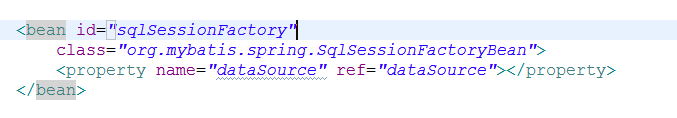
root-context.xml 파일에도 HikariCP를 추가한 모습이다.



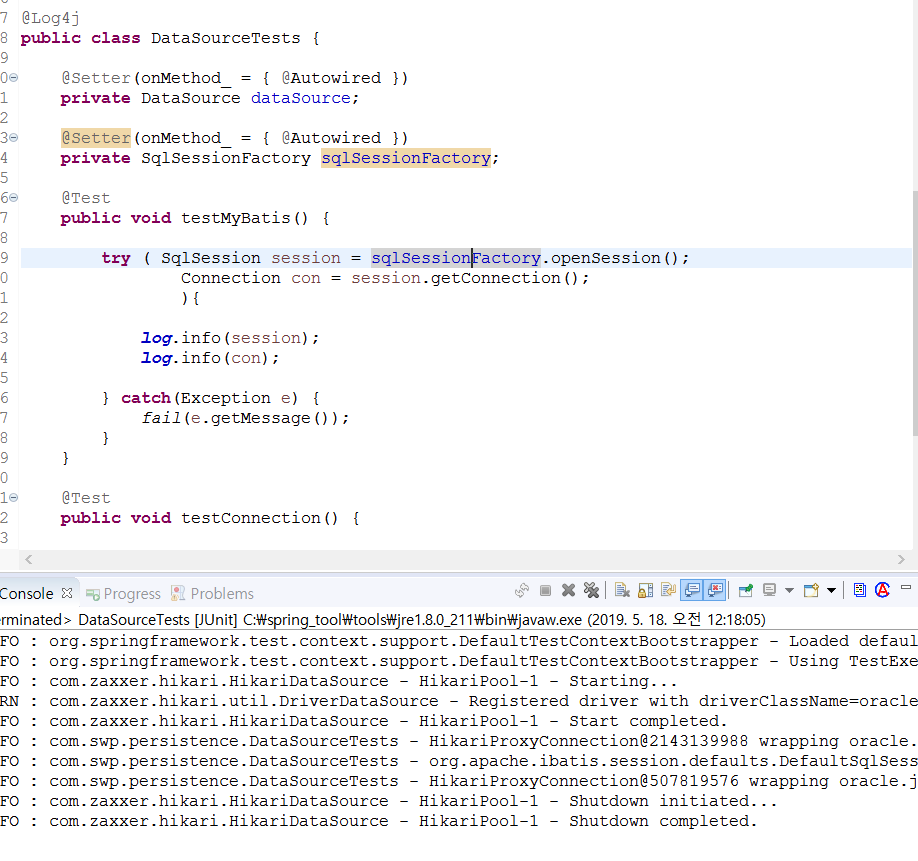
DataSourceTest 클래스를 작성하여 Junit 테스트를 진행한 모습이다. Hikari가 적용이 된 것을 확인 할 수 있다.



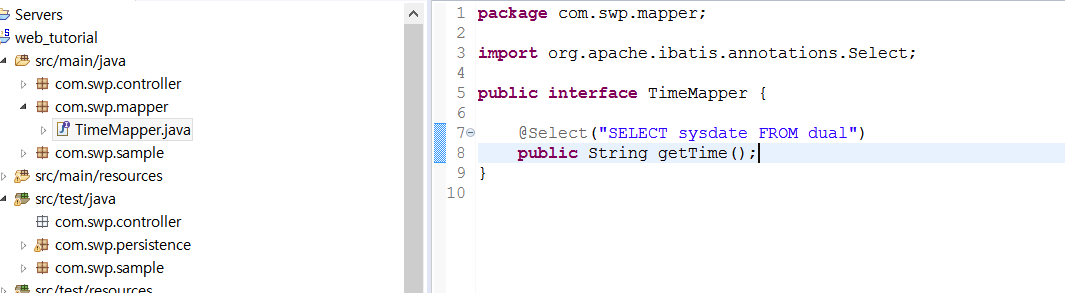
MyBatis를 적용하기 위하여 pom.xml에 추가한 모습이다.



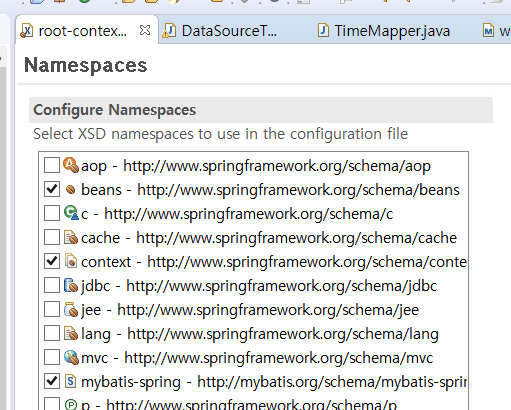
root-context.xml 파일에 SqlSessionFactory를 추가한 모습이다.



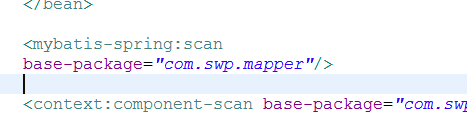
DataSourceTest 클래스에도 SqlSessionFactory를 추가하고 Junit 테스트를 진행한 모습이다.



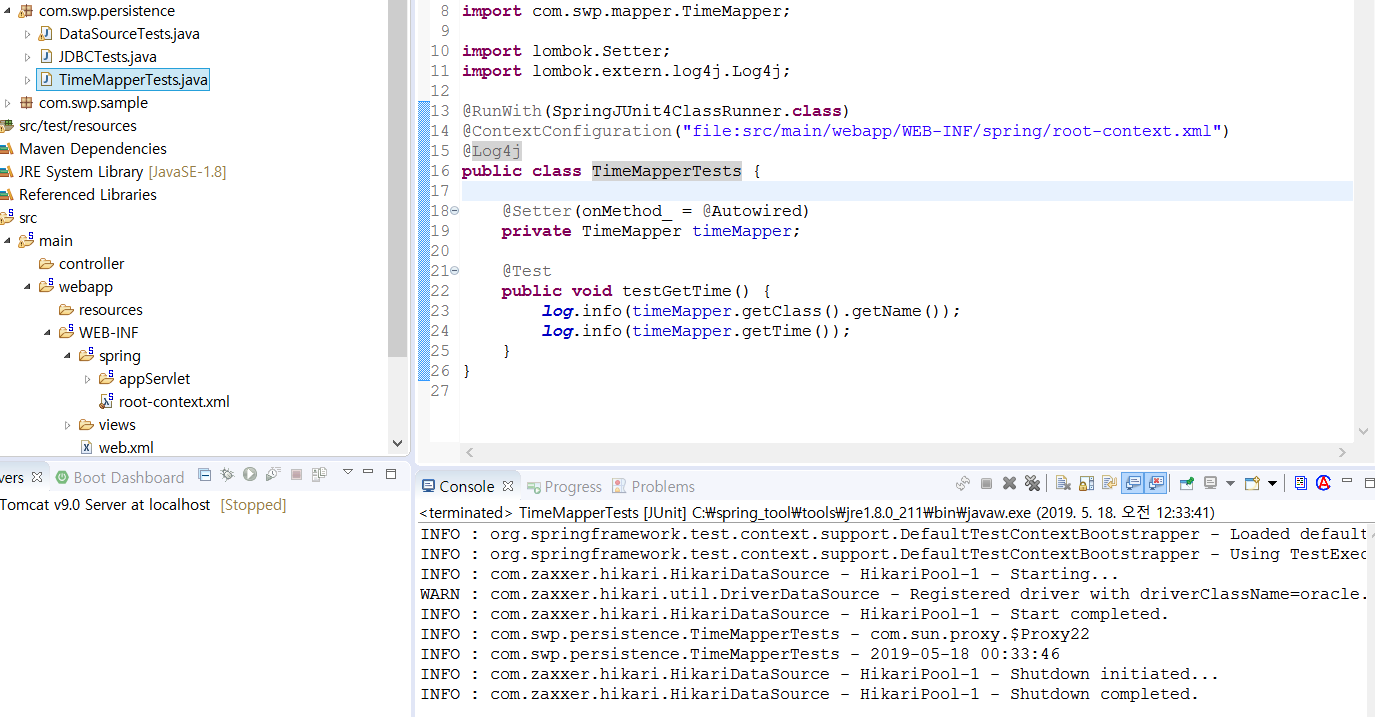
Mapper 인터페이스를 작성하였다.



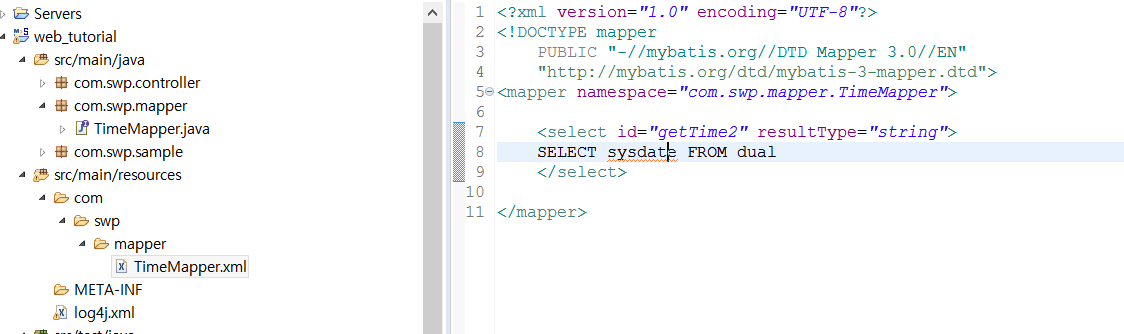
root-context 의 Namespace에서 mybatis를 체크한 모습이다.



root-context.xml 파일의 mybatis의 base 패키지를 com.swp.mapper로 지정한 코드를 작성하였다.



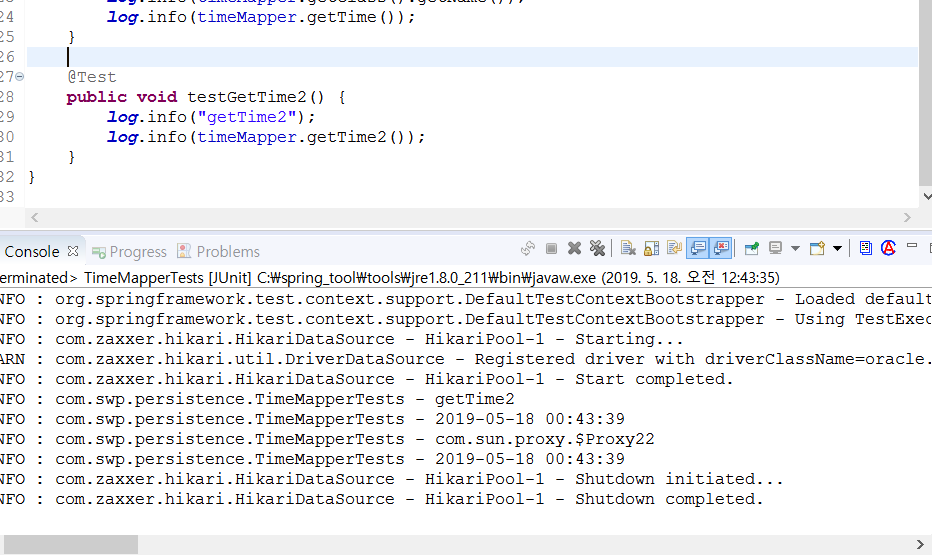
Mapper를 테스트하기 위해 TimeMapperTest 클래스를 작성하고 Junit 테스트를 진행한 모습이다.



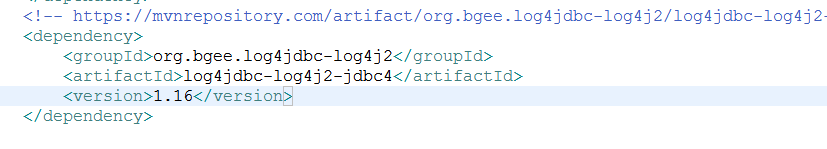
TimeMapper.xml 파일을 작성하였다.



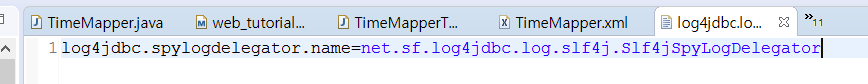
TimeMapper 인터페이스에 getTime2 코드를 추가하였다.



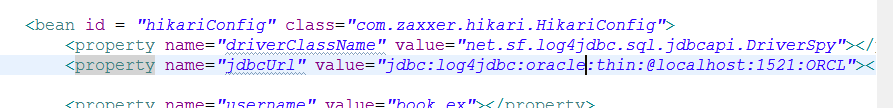
TimeMapperTest 클래스를 수정하고 Junit 테스트를 진행한 모습이다.



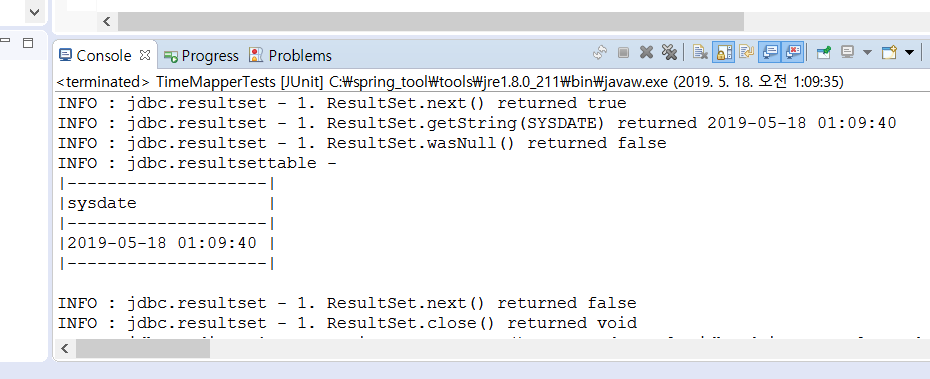
log4jdbc-log4j2 라이브러리를 사용하기 위해 pom.xml을 수정하였다



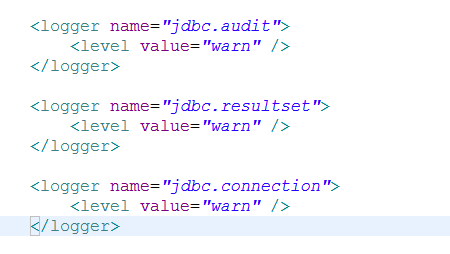
log4jdbc.log4j2.properties 파일을 작성하였다.



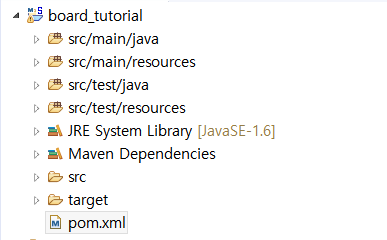
root-context.xml을 수정하였다.



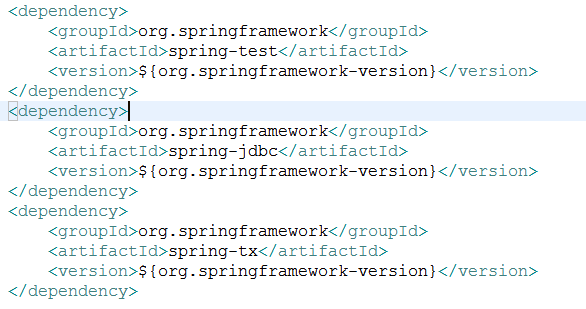
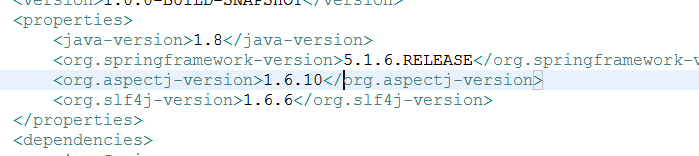
테스트 결과가 이전과 달라진 모습을 확인할 수 있다.



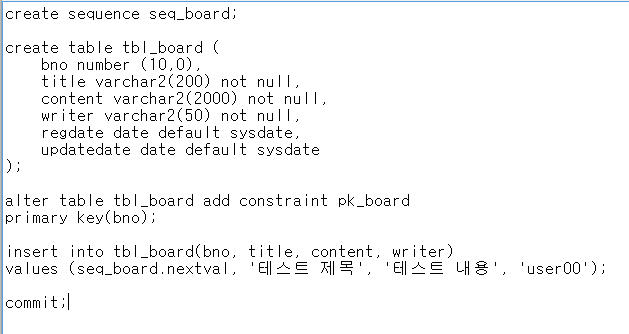
로그의 양을 조절하기 위한 log4j.xml의 코드를 수정한 모습이다.



board\_tutorial 프로젝트를 생성한 모습이다.



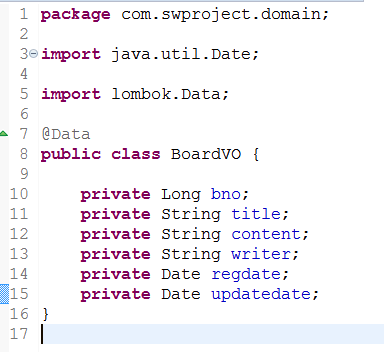
pom.xml 파일을 수정한 모습이다. 이외에도 이전 튜토리얼과 마찬가지로 lombok, junit 수정, 라이브러리 추가 등 모두 수정하였다.



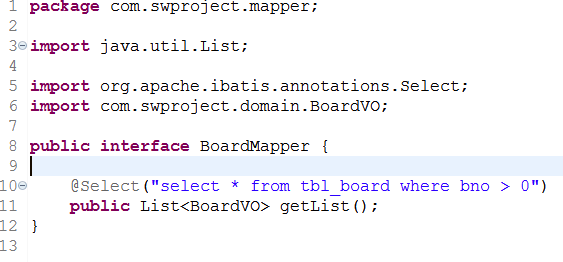
sql developer를 통해 sequence를 생성하고 table을 생성하고, table을 변경하고 행을 삽입한뒤 commit 해준 명령어이다.



root-context를 수정해준 모습이다.



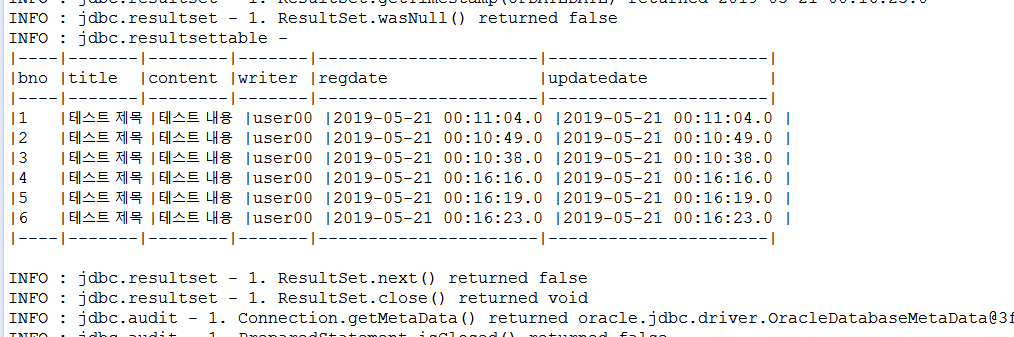
BoardVO 클래스를 작성한 모습이다.



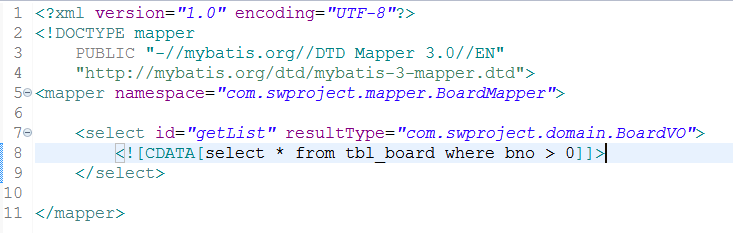
BoardMapper 인터페이스를 작성한 모습이다.



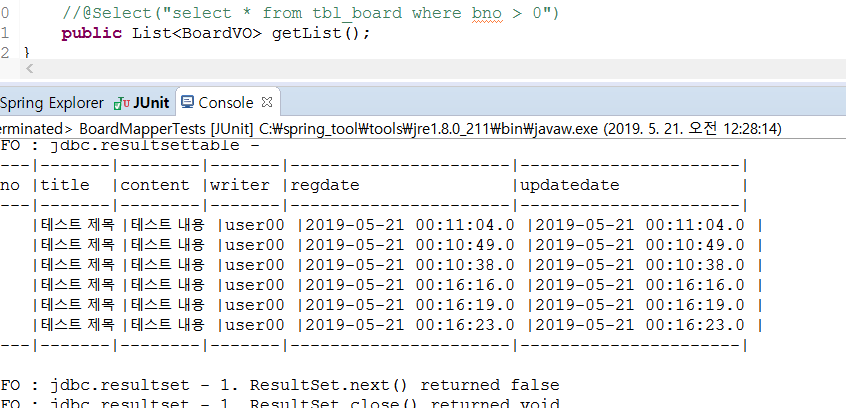
BoardMapperTest 클래스를 작성한 모습이다.



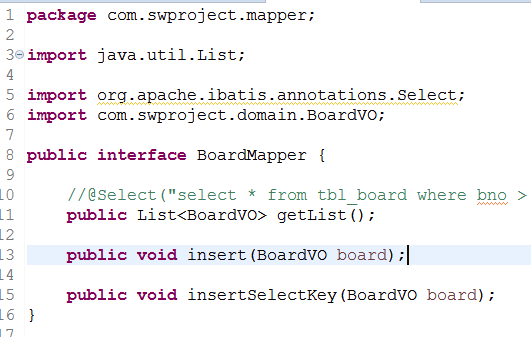
Junit 테스트를 통한 결과화면의 모습이다.



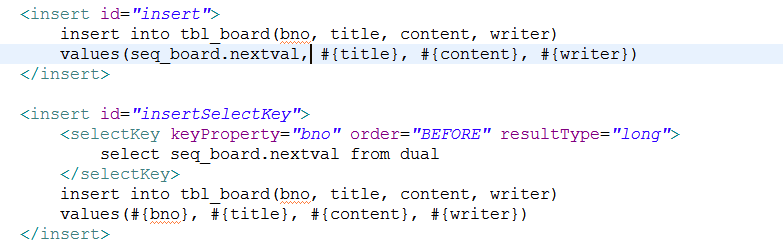
BoardMapper.xml 파일을 작성한 모습이다.



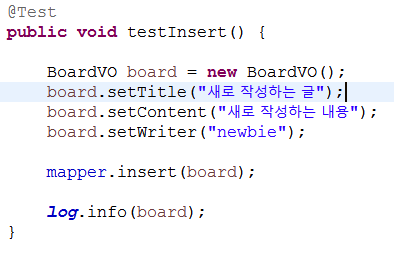
이후의 Junit 테스트를 통한 결과화면의 모습이다.



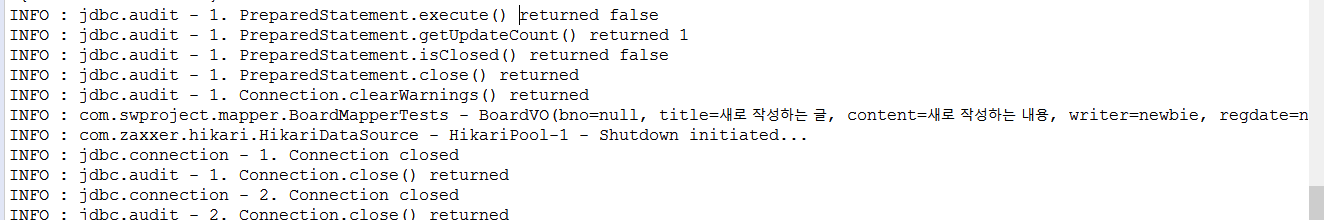
인터페이스에 메소드를 추가 선언한 모습이다.



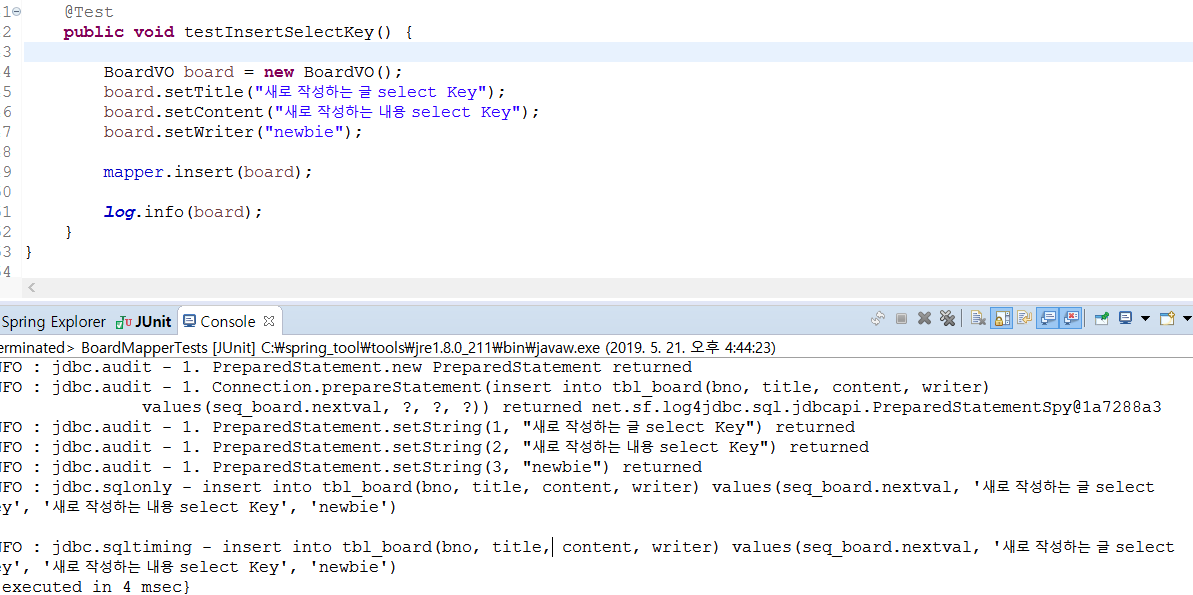
BoardMapper.xml 파일을 수정한 모습이다.



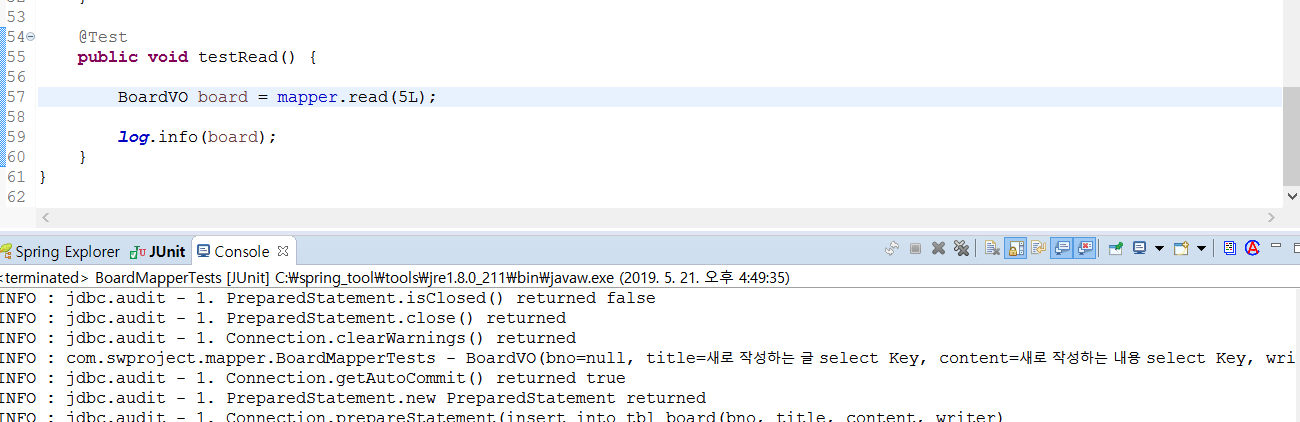
BoardMapperTest 클래스에 insert에 대한 테스트 코드를 작성한 모습이다.



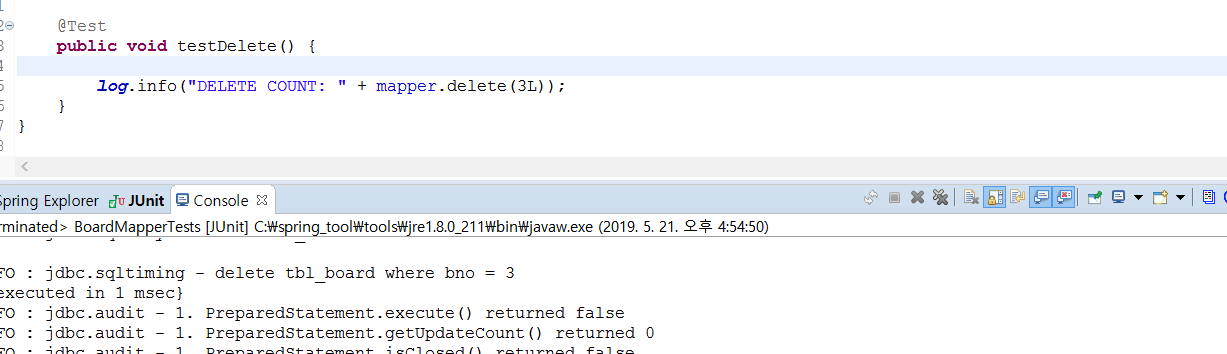
이후 Junit 테스트를 진행한 결과 화면이다.



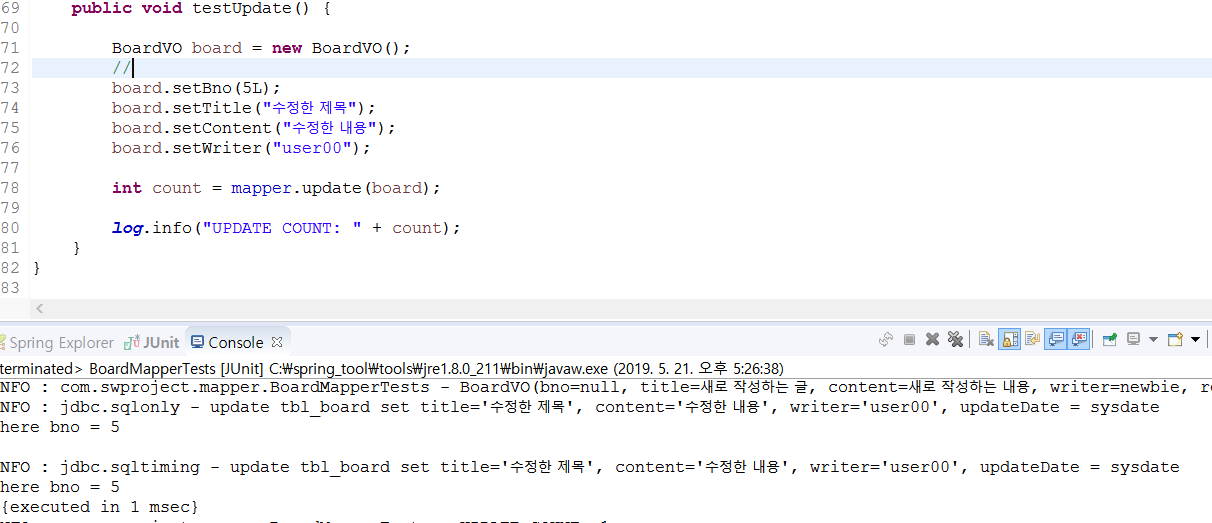
BoardMapperTest 클래스에 @selectkey를 사용하는 insert에 대한 테스트 코드를 작성하고 테스트를 진행한 결과이다.



BoardMapperTest 클래스에 read에 대한 테스트 코드를 작성하고 테스트를 진행한 결과이다.



BoardMapperTest 클래스에 delete에 대한 테스트 코드를 작성하고 테스트를 진행한 결과이다.



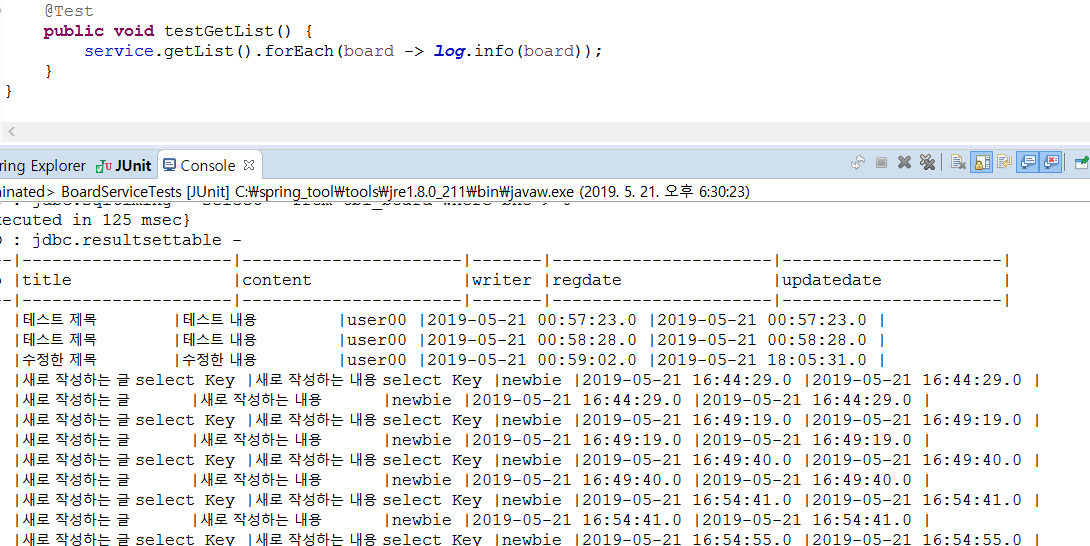
BoardMapperTest 클래스에 update에 대한 테스트 코드를 작성하고 테스트를 진행한 결과이다.



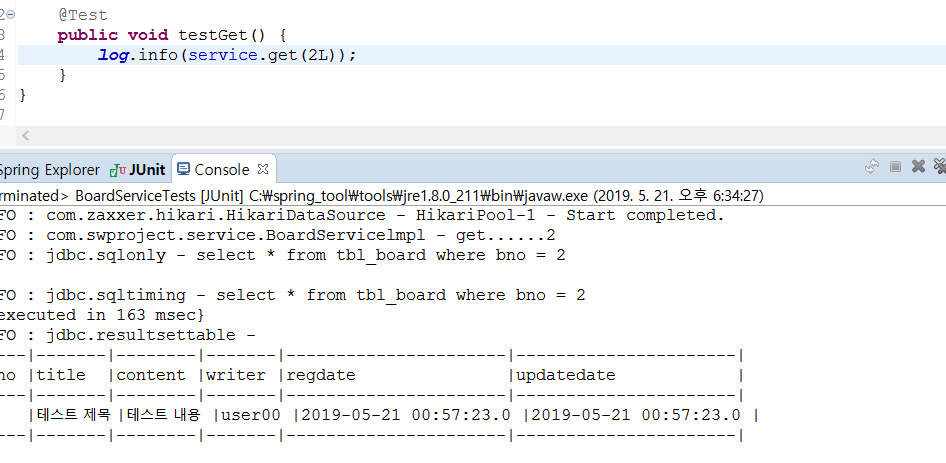
root-context.xml 에 context 문을 추가한 모습이다.



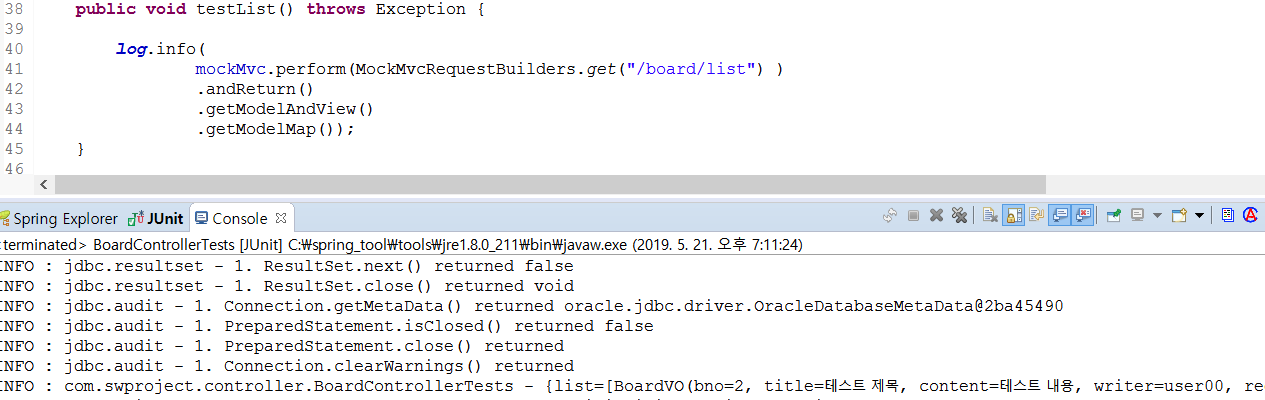
BoardServiceTest 클래스를 작성하고 Register에 대한 테스트 코드를 작성하고 테스트를 진행한 결과이다.



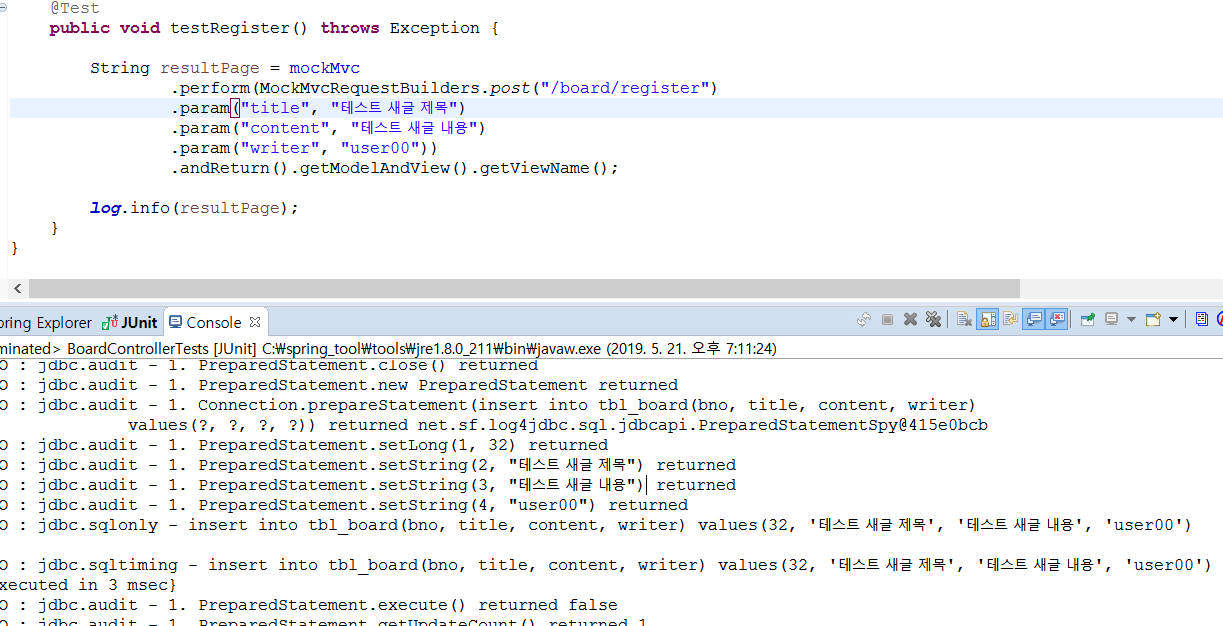
BoardServiceTest 클래스에 GetList에 대한 테스트 코드를 추가하고 테스트를 진행한 결과이다.



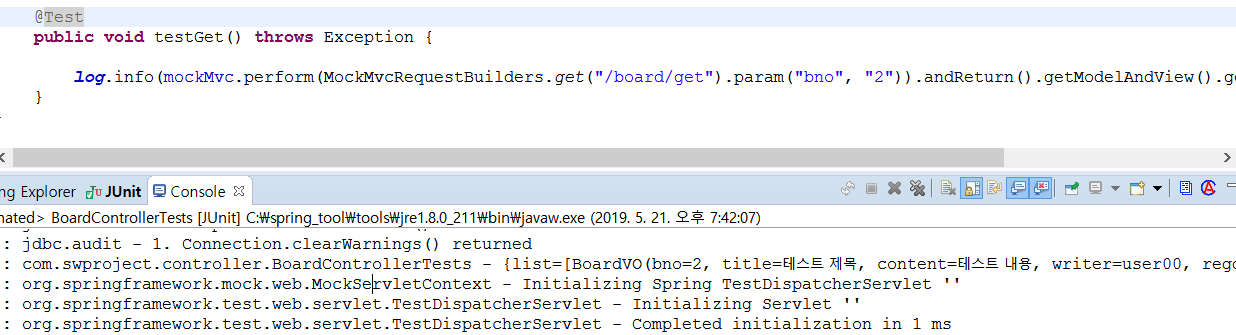
BoardServiceTest 클래스에 Get에 대한 테스트 코드를 추가하고 테스트를 진행한 결과이다.



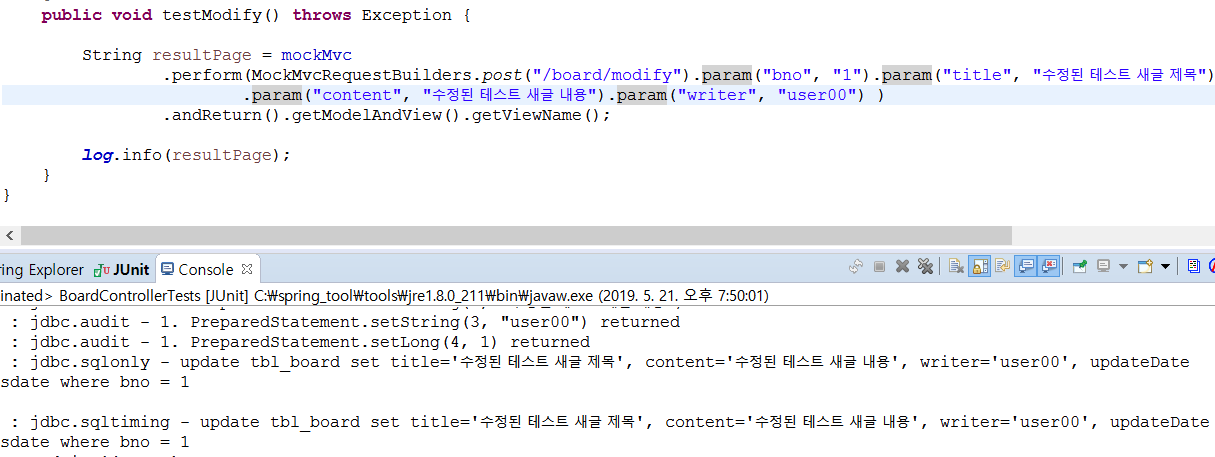
BoardControllerTest 클래스를 작성하고 List에 대한 테스트 코드를 작성하고 테스트를 진행한 결과이다.



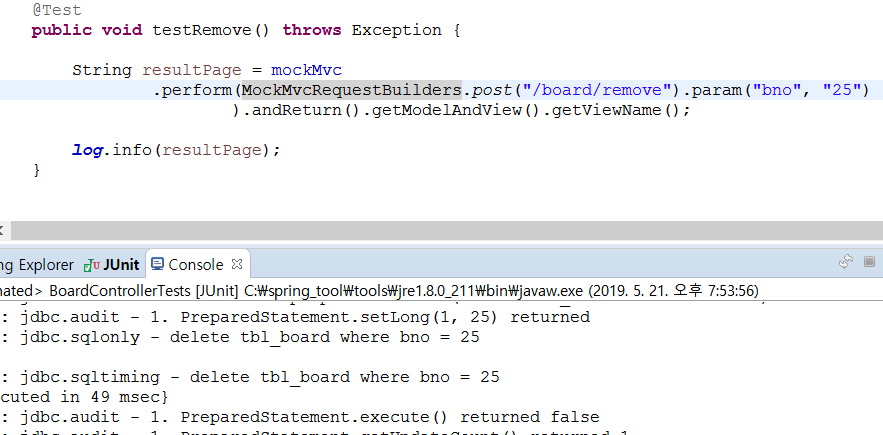
BoardControllerTest 클래스에 Register에 대한 테스트 코드를 추가하고 테스트를 진행한 결과이다.



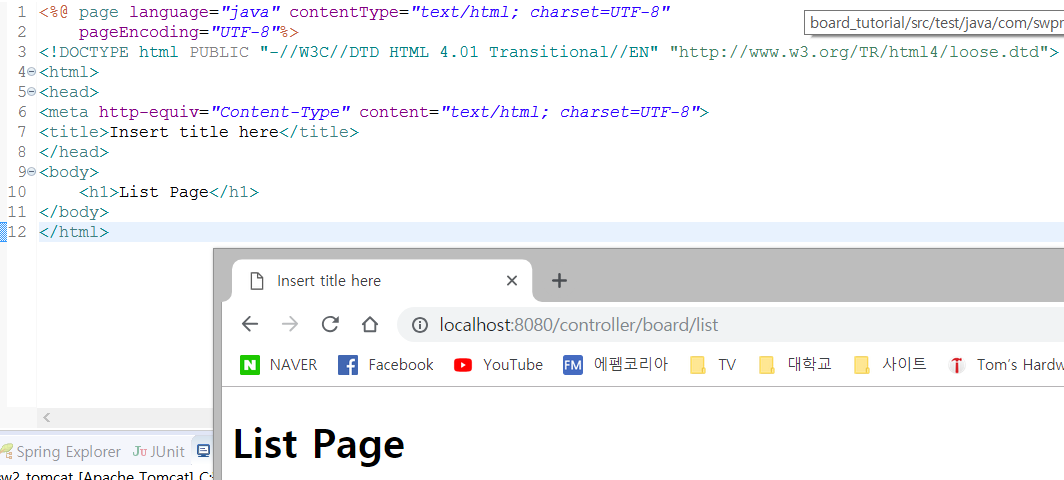
BoardControllerTest 클래스에 Get에 대한 테스트 코드를 추가하고 테스트를 진행한 결과이다.



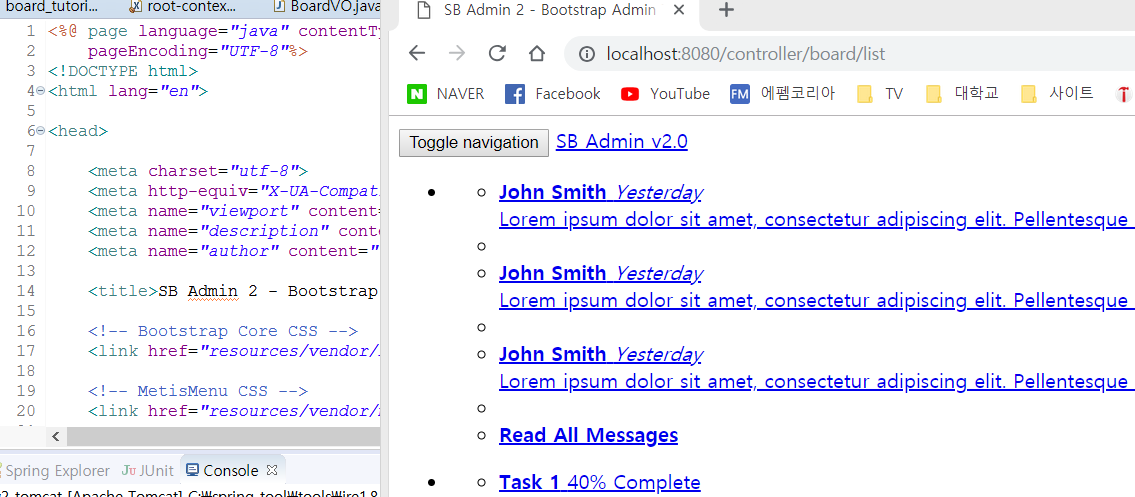
BoardControllerTest 클래스에 Modify에 대한 테스트 코드를 추가하고 테스트를 진행한 결과이다.



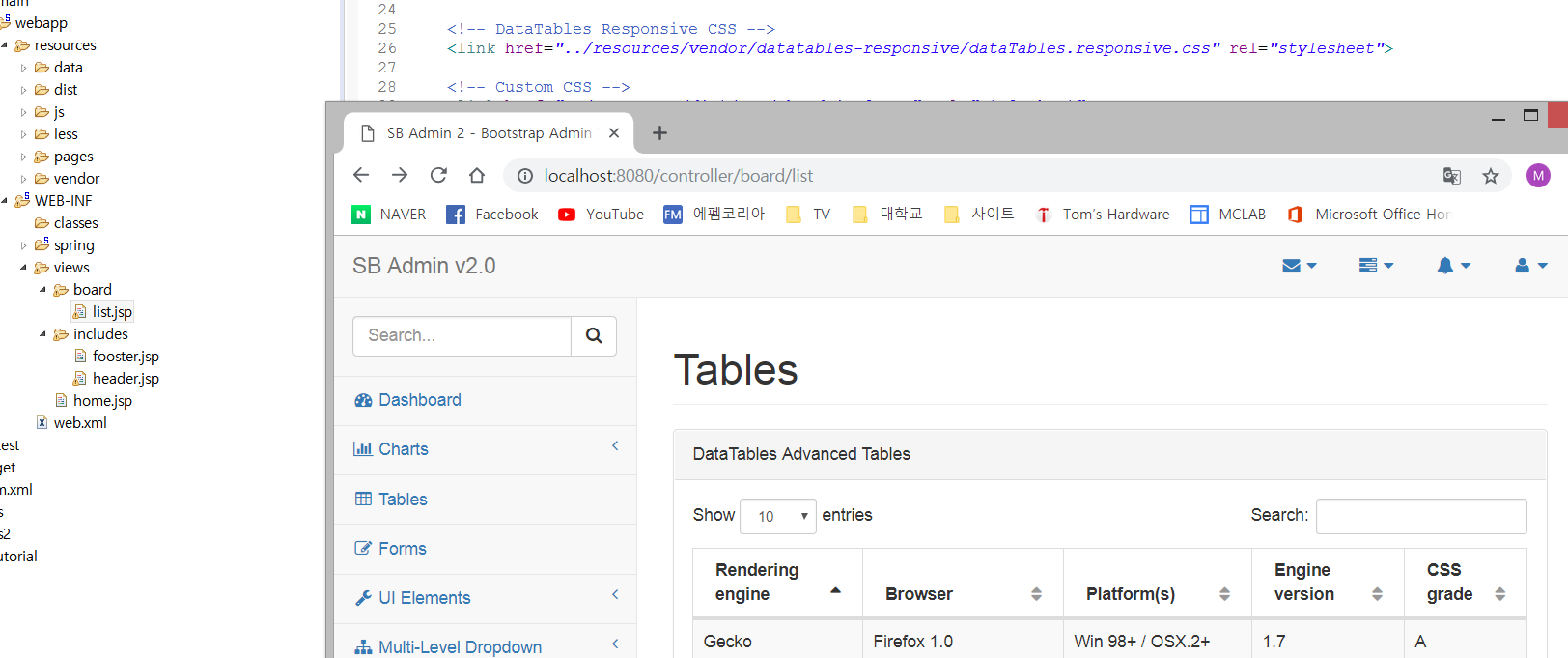
BoardControllerTest 클래스에 Remove에 대한 테스트 코드를 추가하고 테스트를 진행한 결과이다.



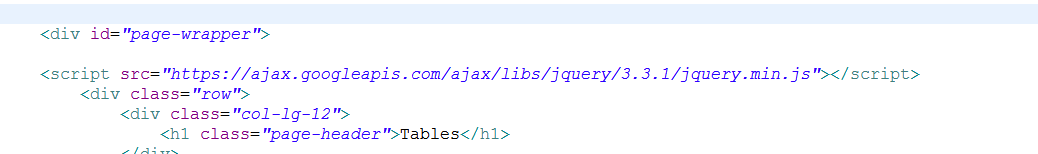
화면처리를 위해 list.jsp 파일을 생성하여 작성한 후 결과를 확인해본 모습이다.



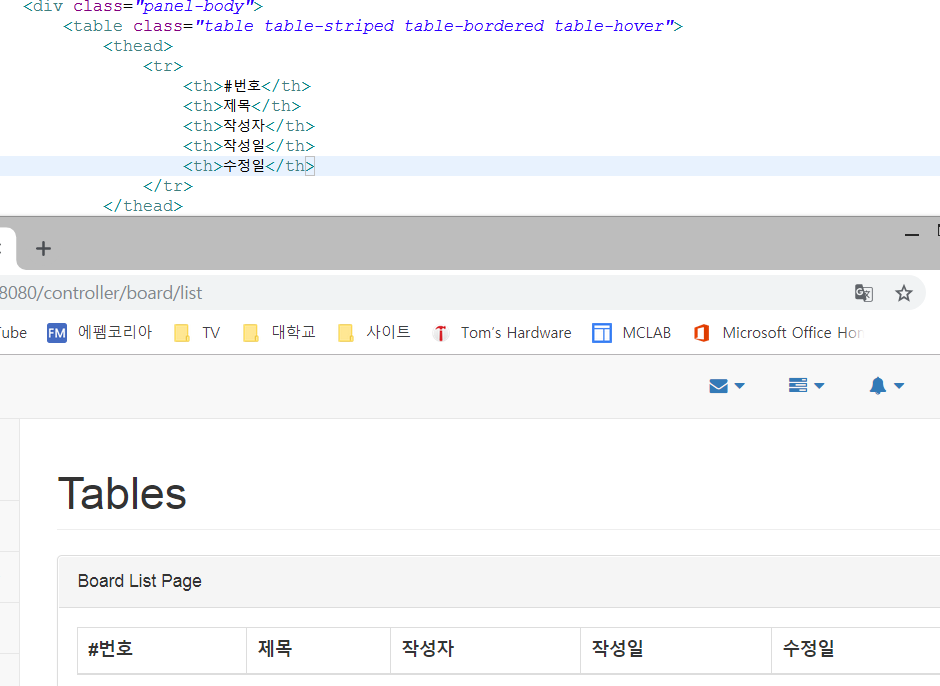
SB Admin2 페이지의 table의 내용을 list로 옮긴 뒤의 결과를 확인해본 모습이다.



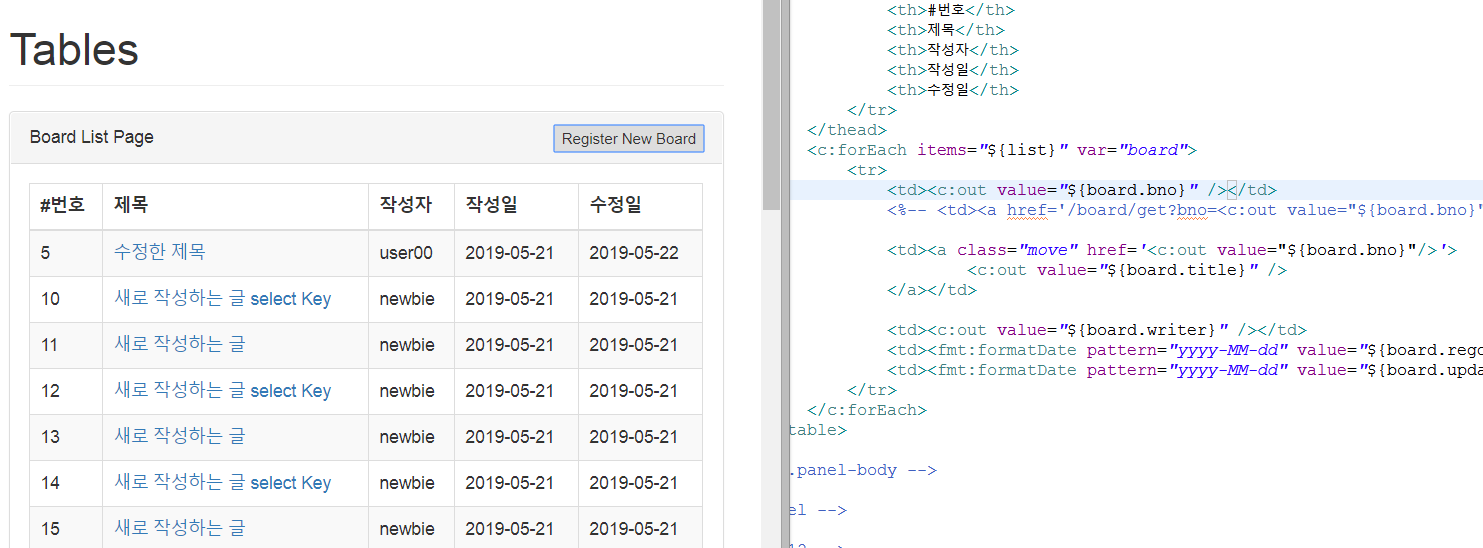
SB Admin2 페이지의 모든 내용을 옮긴 뒤 header.jsp, footer.jsp를 생성한 뒤 적용하여 결과를 확인해본 모습이다.



header.jsp 파일에 jQuery를 다운 받을 수 있도록 링크를 검색하여 추가한 모습이다.

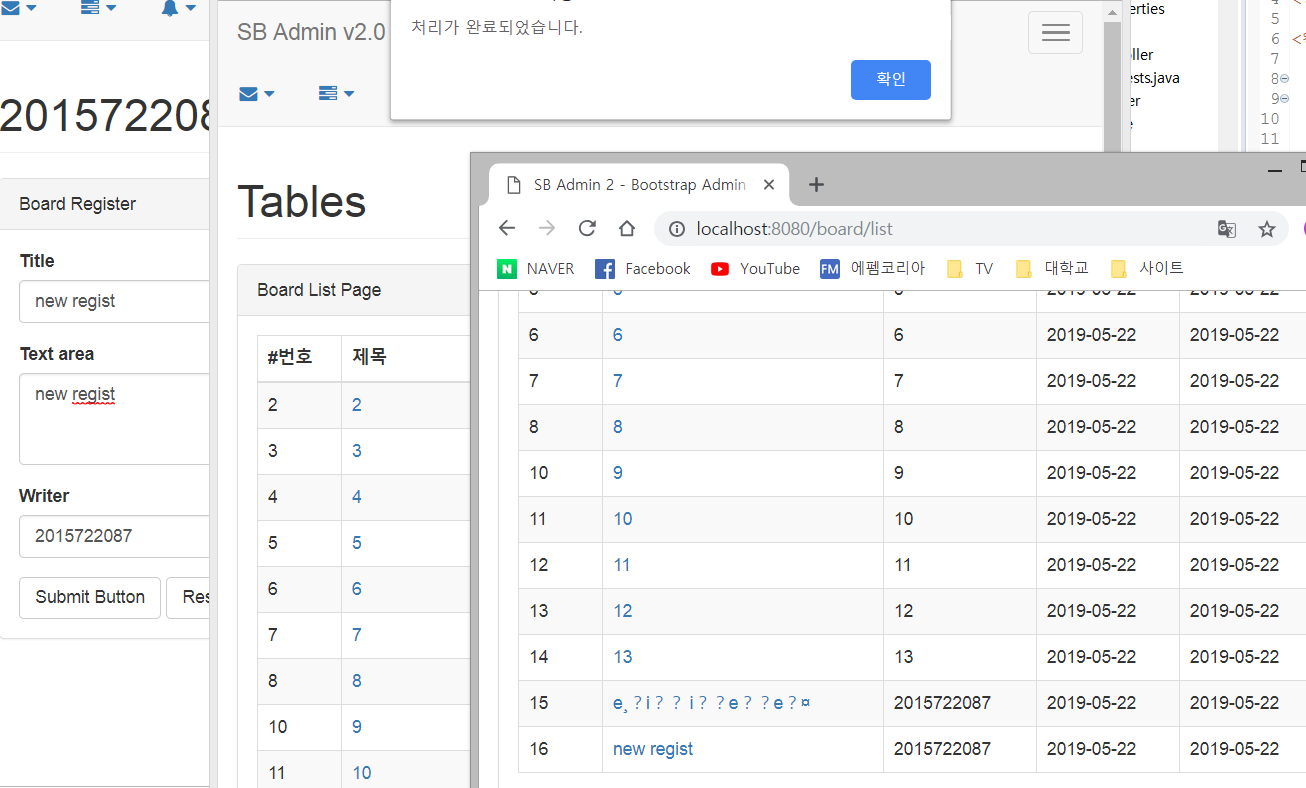


list의 내용을 수정하여 글 목록의 제목을 수정한 모습이다.



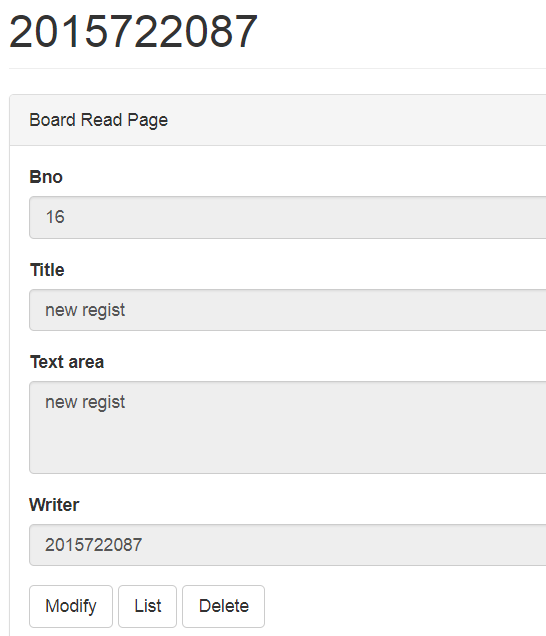
Model에 담긴 데이터를 출력하는 코드를 작성한 뒤 브라우저에서 확인해본 모습이다.

1. **게시판 기능 동작에 대한 과정 설명 및 캡쳐**
   * Create



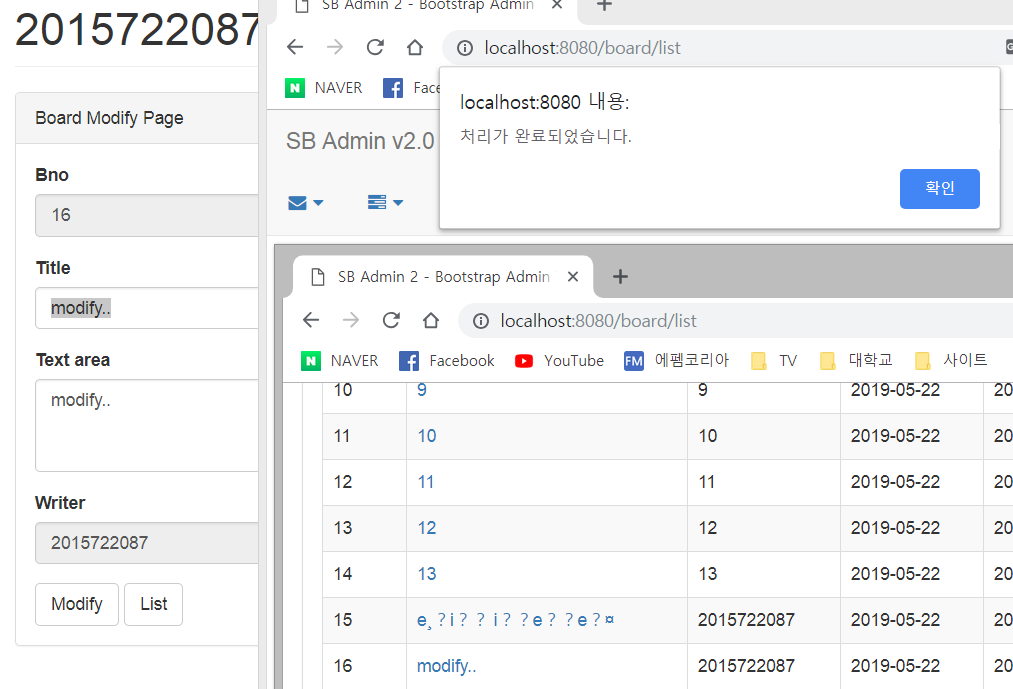
- 글 등록 기능을 위한 메소드는 모두 튜토리얼을 통해 구현이 되었고, views 폴더에 register.jsp 파일을 작성하여 브라우저에서 글 등록을 처리할 수 있는 페이지를 생성하였다. 글 목록에서 상단부분에 있는 Register New Board 버튼을 클릭하면 글 등록 페이지로 이동하는데, 이때 글의 번호(bno)는 순서에 따라 번호가 부여되며, 제목(title), 내용(content), 작성자(writer)을 입력하여 글을 등록하게 하였다. 이때 3가지 항목 중 입력되지 않은 것이 있다면 입력되지 않았으니 입력을 하라고 커서를 해당 칸에 위치하게 하여 예외처리를 해주었고, 모두 입력한 뒤에야 글을 등록할 수 있으며 글이 등록되면 list 페이지로 돌아가고 처리가 되었다는 메세지를 출력한다.

* + Read



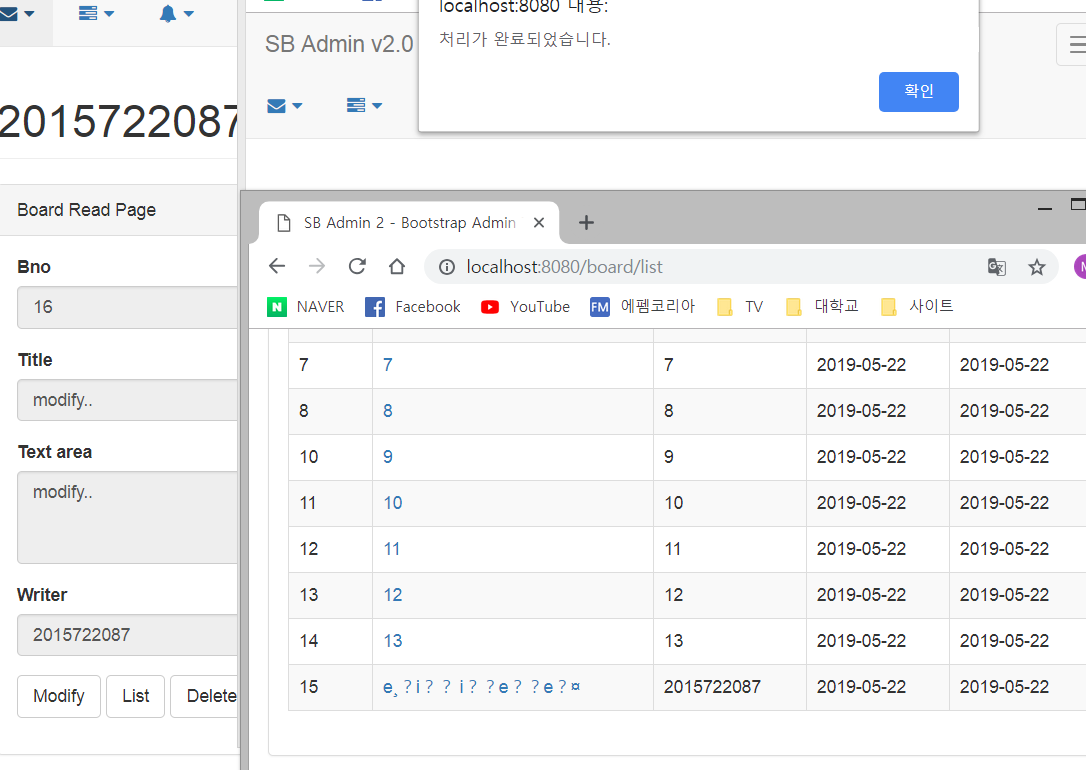
- 글을 확인하는 기능을 위한 메소드 또한 튜토리얼을 통해 구현되었으며, get.jsp 파일을 작성하여 브라우저에서 글 열람이 가능하도록 하였다. 글의 제목을 클릭하면 열람하는 페이지로 이동하는데, 이때 글의 번호, 제목, 내용, 글쓴이를 보여준다. 이는 모두 Readonly로 출력하여 해당 페이지에서는 수정할 수 없고 말 그대로 열람만 가능한 페이지이다. 글의 아래쪽에 해당 글 수정, List로 돌아가기, 해당 글 삭제 버튼을 만들어 해당 기능이 수행하도록 만들어 두었다.

* + Update



- 글을 수정하는 기능을 위한 메소드 또한 튜토리얼을 통해 구현되었으며, modify.jsp 파일을 작성하여 브라우저에서 글을 수정할 수 있도록 하였다. 이 기능을 위해서는 수정하고 싶은 해당 글의 제목을 클릭하여 get.jsp 페이지로 이동한 뒤, 글 아래쪽에 버튼으로 만들어둔 Modify 버튼을 클릭하면 해당 페이지로 이동한다. 이때 글의 번호, 제목, 내용, 글쓴이를 보여주는데, 번호와 글쓴이는 readonly로 출력하여 수정할 수 없으며, 제목과 내용만을 수정할 수 있게 하였다. 글을 수정하는 과정에서도 제목 혹은 내용이 입력되 않았을 경우 예외처리를 하여 입력하지않으면 수정할 수 없도록 하였으며, 수정이 완료되는 경우 List 페이지로 돌아가고 처리가 완료되었다는 메세지를 출력한다.

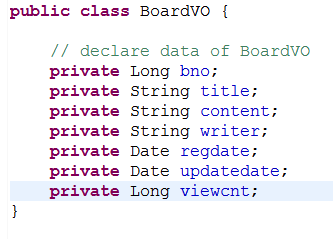
* + Delete



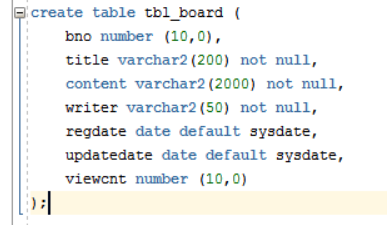
- 글을 삭제하는 기능을 위한 메소드 또한 이미 구현이 되어있고, 이는 글을 삭제하는데 따로 페이지가 필요하지 않기때문에 jsp 파일은 생성하지 않았다. 이 기능을 위해서는 해당 글의 제목을 클릭하여 get.jsp 페이지로 이동한 뒤 Delete 버튼을 클릭하면 삭제하는 메소드를 실행하여 글을 삭제하도록 하였다.

1. **추가 기능 구현**
   * 글 조회 수 증가 기능

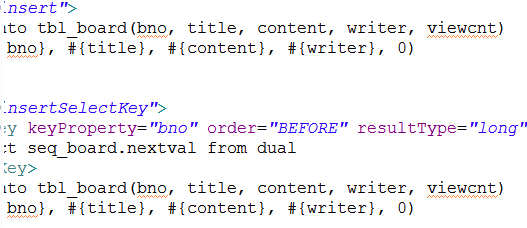
- 글 조회 수 증가 기능은 글을 한번 열람할 때마다 1씩 증가하여 몇번 열람하였는지 확인할 수 있는 기능이다. 이 기능을 추가하기 위해서는 기존 데이터베이스에 생성한 table의 데이터 - 글번호, 제목, 내용, 글쓴이, 작성시간, 수정시간 - 이외에 추가로 조회수를 추가하여 글마다 조회수를 확인할 수 있도록 구현한다.



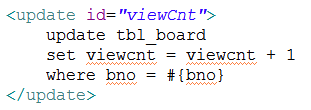
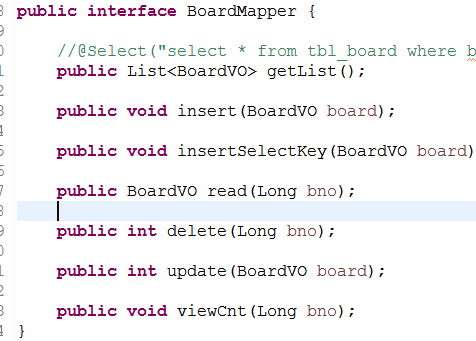
- 우선 boardVO에 viewcnt 데이터를 추가로 선언하였다.



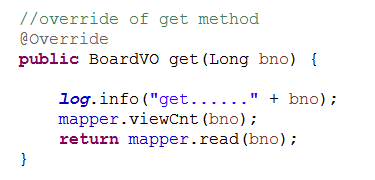
- 테이블 생성시 viewcnt 데이터도 추가하였다.



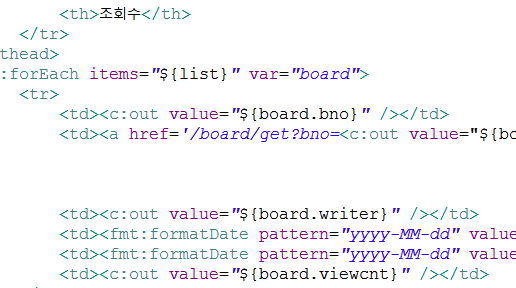
- 글을 insert 할때 viewcnt 데이터 값에도 처음에 0을 부여하도록 하였다. 글이 등록되면 처음 조회수는 0이어야 하기 때문이다.



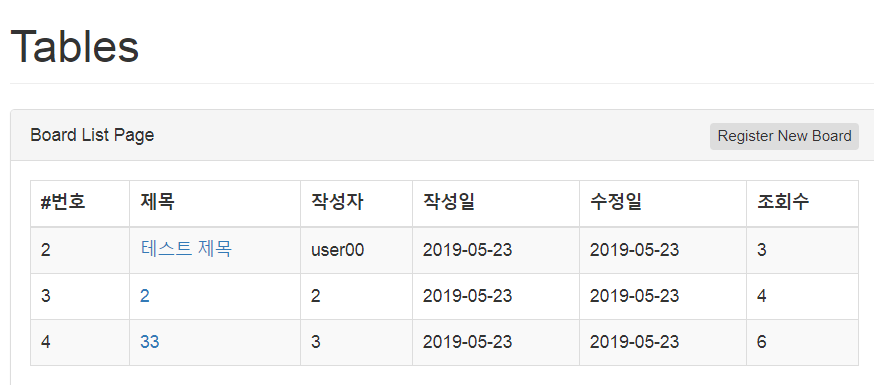
- BoardMapper 인터페이스에도 viewCnt 메소드를 추가하였다. 이 메소드는 인자로 입력되는 bno의 데이터의 viewcnt 데이터 값을 +1 해주는 역할을 수행하도록 구현한다.



- read시에 viewCnt 메소드를 수행하고 read 메소드를 수행하도록 작성하였다.



- list.jsp 파일에서 viewcnt 데이터도 출력하도록 테이블에 출력 데이터를 추가로 작성하였다.



- 글을 한번 읽고 list로 돌아오면 조회수가 +1 씩 증가하는 것을 확인 할 수 있다.

1. **튜토리얼 관련 내용 추가 조사**
   * MVC 패턴

MVC 패턴은 디자인 패턴 중의 하나인데, 디자인 패턴이란 프로그램이나 어떤 특정한 것을 개방하는 중에 발생한 문제점들을 정리해서 상황에 따라 간편하게 적용해서 쓸 수 있는 것을 정리하여 특정한 '규약'을 통해 쉽게 쓸 수 있는 형태로 만든 것이다. MVC는 'Model-View-Controller'의 약자이다. 사용자가 Controller를 이용하여 Model의 데이터를 View를 통해서 사용자가 결과를 확인하는 방식이다. 이를 이용하면 코드의 중복 작성을 해결할 수 있으며, 유연성, 확장성이 증가한다.

- Controller : 명령을 내려 Model이나 View를 변경시킬 수 있다.

- Model : 상태가 변할때 컨트롤러와 뷰에 신호를 보낸다. 이에 따라 Controller는 변화에 따른 명령을 내리며, View에서는 최신의 결과를 출력한다.

- View : 결과물을 출력하기 위하여 모델에서 데이터를 가져온다.

* + Maven

Maven은 Project Object Model의 개념을 바탕으로한 프로젝트 관리 도구이다. 의존성, 라이브러리 관리 등의 기능을 제공한다. 쉽게 말하면 개발자가 개발을 할때, 자신의 코드만 사용하는 것이 아니라, 다양한 라이브러리를 활용하여 개발을 할 수 있다. 이때 라이브러리의 개수가 너무 많아지면 관리하기 어려워 질 수 있는데, 이런 문제를 해결하기 위한 도구가 Maven이다.

* + Lombok

개발자가 Java로 작성된 Model Object를 만들 때, 반복적으로 불필요하게 작성하게 되는 코드를 Annotation을 이용하여 줄일 수 있게 해주는 단축 Annotation을 제공하는 자바 라이브러리이다.

Ex ) @Setter - 세터 메소드를 생성. @ToString - toString 메소드를 생성, @AllArgsConstructor - 모든 인자를 받는 생성자.

* + GET 방식과 POST 방식

이 방식은 HTTP 프로토콜을 통해 데이터가 전달될 때 사용하는 방식이다. 먼저 GET 방식은 클라이언트에서 서버로 데이터를 전달할 때, 주소뒤에 Name과 Value가 결합된 문자열 형태로 전달하는데, 이때 주소창에 문자열이 존재하기 때문에 보안성이 떨어지며, 길이에 제한이 생긴다. 이는 전송 데이터의 한계를 의미한다. POST 방식은 일정 크기 이상의 데이터를 전달할 때 사용하는데, 보내기 전에 Encoding 작업을 하고, 서버에서는 Decoding 작업을 수행한다. 이 방식은 주소창에 문자열이 나타나지 않아 보안성이 높지만 GET 방식보다 느리다는 단점이 있다.

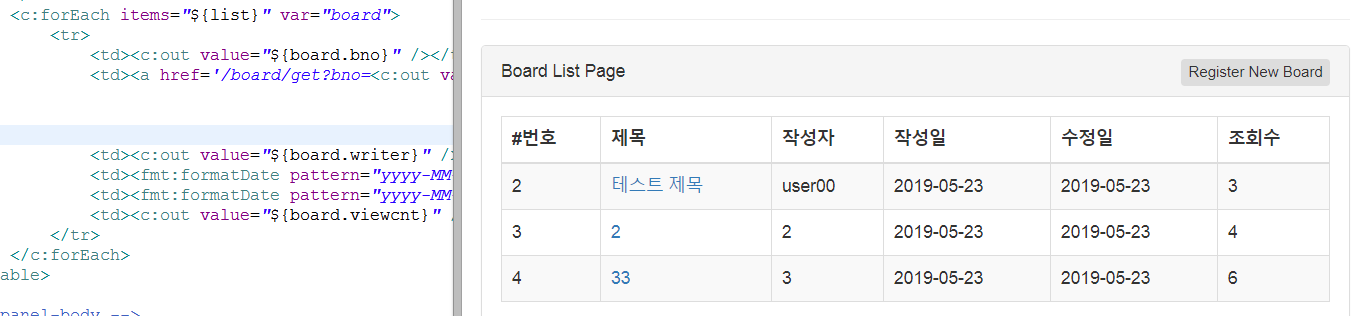
* + Junit

Junit이란 스프링 개발자가 지정한 일부 코드를 독립적으로 테스트할 수 있도록 해주는 툴이자 프레임워크이다. 다른 코드에 영향을 주지 않으면서 일부 코드만을 구동하여 확인할 수 있다. 이번 프로젝트에서는 프로젝트의 Test 폴더에 작성된 test 클래스 파일들을 이용하여 Junit을 이용하였다.

* + Front-End

- JSTL

JSTL이란 JSP Standard Tag Library의 약자이다. 여기서 라이브러리는 여러 프로그램에서 공통으로 사용하는 코드의 집합을 말하는데 JSTL도 같은 의미이다. 그러나 JSTL은 일반 라이브러리와는 다르게 JSP 페이지 안에서 사용 할 수 있는 것을 제공한다. 이 JSTL을 이용하면 여러가지 작업이 수행가능하다.



프로젝트에서 사용한 JSTL 이다. <c:forEach>, <c:out>, <fmt:formatDate>가 바로 JSTL의 사용이다.

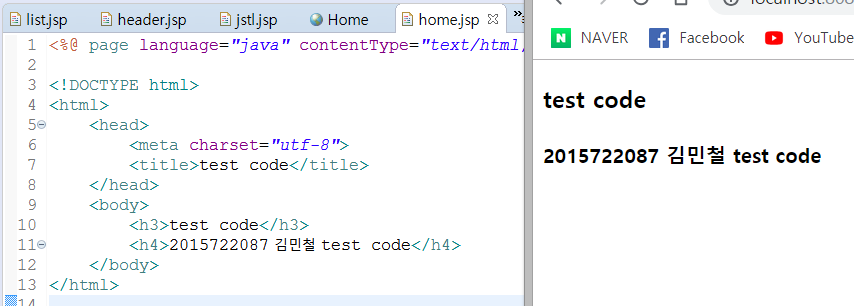
- <c:forEach> : jstl의 반복문이다. items는 컬렉션 객체를 말하고, var은 사용할 변수명을 말한다.

- <c:out> : 변수를 출력할때 사용한다. 변수 board.writer를 출력하게 하였다.

- <fmt:formatDate> : 날짜 정보를 가지고 있는 변수를 포맷팅하여 출력할때 사용한다. pattern은 패턴을 나타내고, value는 변수이다.

- HTML

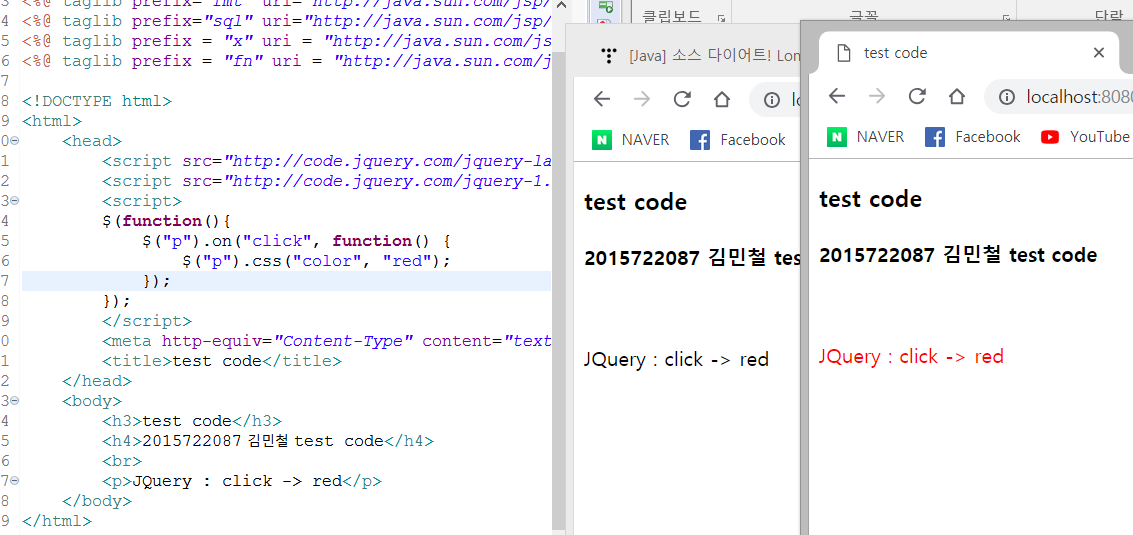
Hyper Text Markup Language의 약자이다. 이 언어로 만들어진 문서를 브라우저에서 확인하는 것이 바로 웹 페이지이다. 웹 페이지의 모습을 기술하기 위한 규약이기도 하며, 간단한 테스트만 진행하였지만 굉장히 많은 기능을 지원한다.



기본 포맷을 작성한 모습이다. <html></html>이 바로 html 문서임을 나타내며, <head> 영역과 <body> 영역으로 존재한다. <title>은 페이지의 제목, <h3, h4>는 굵기를 나타내는 것이다.

- JQuery

수많은 자바스크립트 라이브러리 중 하나이며 가장 많이 쓰이는 라이브러리 중 하나이다.



테스트 해본 코드는 클릭하여 글씨의 색깔을 바꾸게 하는 코드이다. 먼저 15번째 열에 작성된 것은 <p> 로 작성된 코드 부분은 click 할시 color를 red로 바꾸어준다는 의미이다. 이를 사용하기 위해 12번째 열에서 소스를 받아오게 하였으며, 27열에서 <p></p> 를 이용하여 사용하였다.

- JavaScript

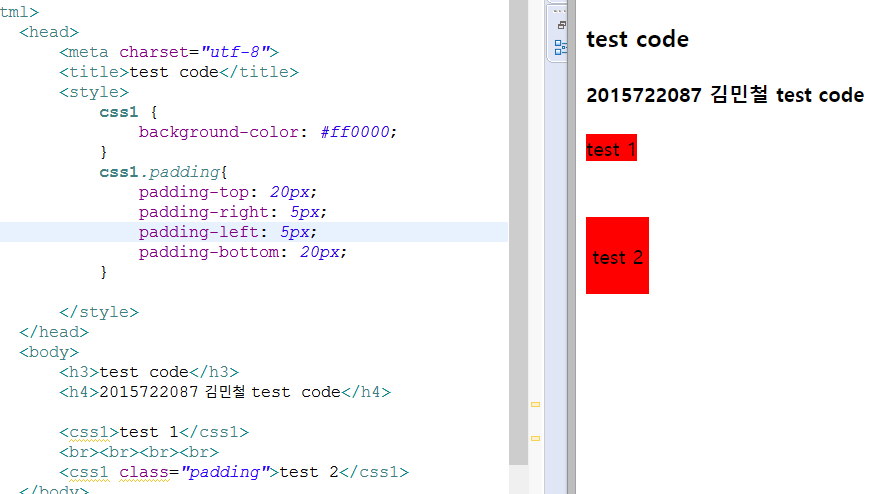
객체 기반 스크립트 언어이다. 주로 웹 브라우저에 사용되며 웹의 동작을 구현한다.



사용한 코드는 document.write() 메소드 이다. 보통 테스트나 디버깅을 위해 이용되는 코드이고, 실행시 모든 데이터를 지우고 결과 데이터를 출력한다. 이를 버튼에 선언하여 버튼을 클릭할시 실행되게 하여 페이지 출력의 변화를 확인하였다.

- CSS

웹의 디자인을 위해 주로 사용되는 스타일 시트 언어이다. Cascading Style Sheets의 약자이며, 웹 페이지의 디자인과 내용을 분리해 놓은 서식이다.



CSS의 패딩 기능을 테스트 해본 모습이다. test1에는 css1.padding의 설정이 적용되지 않았으나, test2에는 css1.padding의 설정이 적용되어 다른 모습으로 출력되었다.

background-color는 배경 색상, padding-top은 위쪽의 공간을 지정하는 것이다.

1. **프로젝트 실행 및 기능 설명을 업로드 한 Youtube URL**
   * **https://youtu.be/WNyIJCGU6z4**
2. **고찰**

이번 프로젝트는 1차 프로젝트 2번의 과제에서 학습한 Java 언어로 스프링 프레임 워크에서 웹사이트를 구현해 보는 프로젝트이다. 대부분의 사람들이 데스크탑 컴퓨터나 노트북, 스마트폰으로 가장 많이 이용하는 것 중 하나가 웹 서핑인데, 웹 서핑은 결국 웹 페이지를 보는 것이다. 그런 웹 페이지를 교양 과목에서 해본 단순한 HTML 파일의 웹이 아니라, 스프링 프레임 워크를 이용하여 더 복잡하고, 기능이 많은 웹페이지를 구현하는 프로젝트를 해야 한다는 생각에 걱정이 많았던 것 같다. 그냥 웹페이지를 만드는 것도 잘못하는데 복잡한 웹페이지를 구현해야 한다는 것이 처음에는 막막했다. 그러나 제공된 튜토리얼에 과정이 상세히 나와있어서 튜토리얼을 보면서 한 단계, 한 단계 씩 하다보니 어느새 웹페이지가 만들어져 있었다.

중간 중간에 오류가 발생하면, 무엇이 오류인지 알 수가 없어서 한참을 고민하고 구글에서 검색하고, 프로그램을 다시 깔아보고, 코드를 처음부터 다시 작성해보면서 시간이 굉장히 많이 소요되었고 튜토리얼을 하는데만 일주일이 넘게 걸린 것 같다. 그래도 결국 하다보니 개념이 어느정도 잡히기 시작해서 나중에는 오류가 생기면 어느 부분에서 오류가 발생했는지 정확히는 아니지만 어느정도 알고 진행할 수 있어서 튜토리얼을 하면서 조금 알게된 것 같다는 생각이 들기도 했지만, 무엇인지 모르고 코드를 작성하게 되어 내가 코드를 작성하는 것이 맞는건가 하는 생각이 들어 구글을 통해 공부를 하면서 진행하려 했다. 그러나 다른 과목들의 과제와 기간이 겹치는 바람에 완전한 이해를 하지 못한채 프로젝트를 급하게 마무리하게 되어 아쉬움이 남는 프로젝트이며, 학기가 끝나면 공부해보고 싶은 분야라는 생각이 들었다.