광운대학교 KwangWoon University

2019 Software Project

Spring 개발환경 구축

Kwangwoon Univ.
School of Computer and Information Engineering

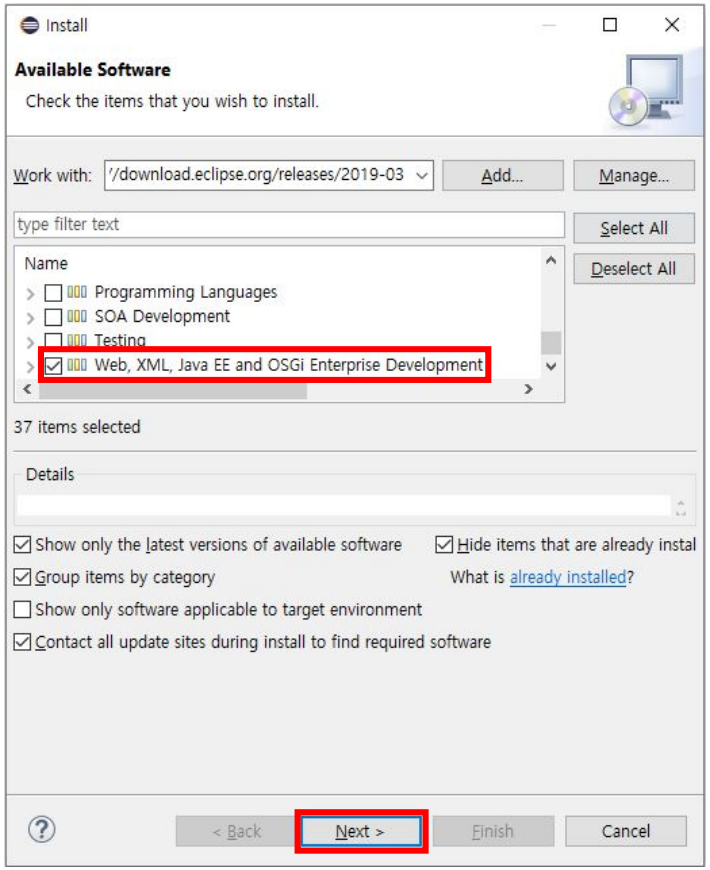
조교 : 임영범 (ylim@kw.ac.kr)

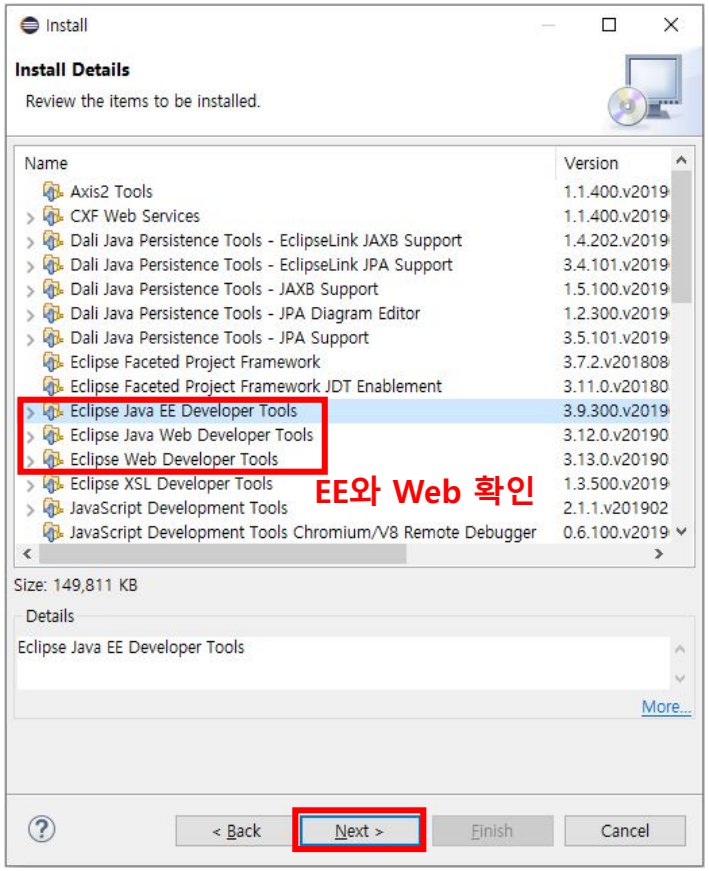
개발환경 설치

- JDK 1.8
  - https://www.oracle.com/technetwork/java/javase/downloads
- STS3 (Spring Tool Suite 3)
  - https://spring.io/tools3/sts/all
- Tomcat
  - https://tomcat.apache.org/download-90.cgi
  - C드라이브로 압축 해제

Eclipse EE 개발환경 설정

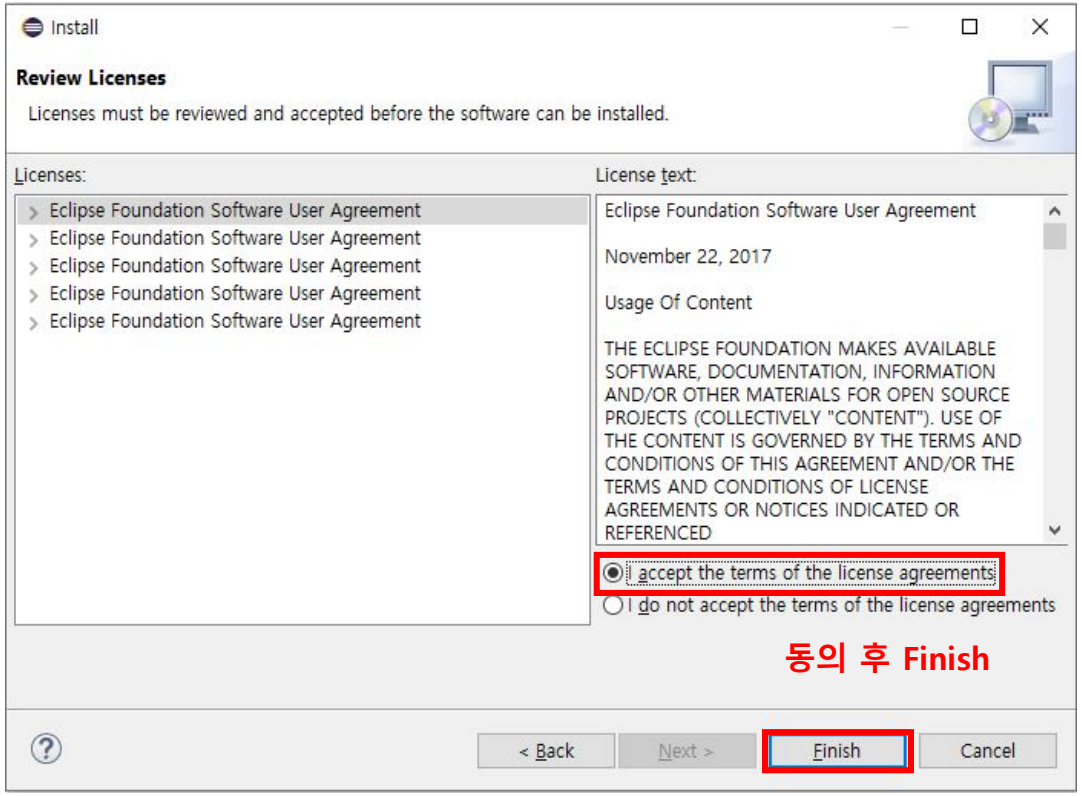
- [Eclipse IDE for Java EE Developer]로 설치하지 않았다면 다음 방법으로 EE(Enterprise Edition) 플러그인 추가





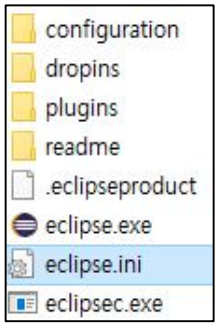
Eclipse EE 개발환경 설정

- [Eclipse IDE for Java EE Developer]로 설치하지 않았다면 다음 방법으로 EE(Enterprise Edition) 플러그인 추가



Eclipse 실행 환경 편집

- 이후 사용할 라이브러리를 위해 이클립스가 설치된 폴더 안에 있는 <eclipse.ini> 파일 수정



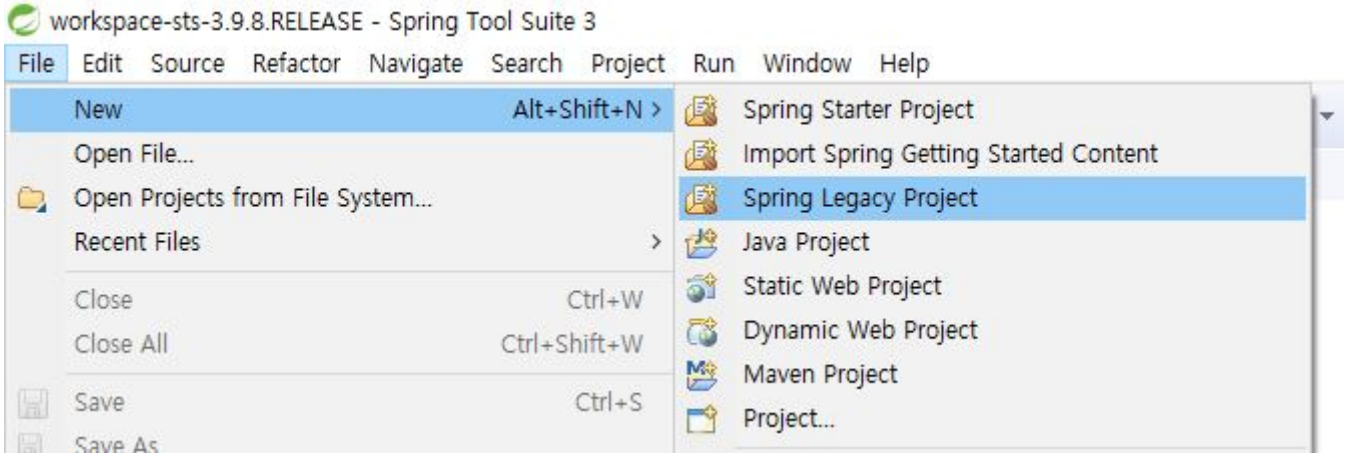
- <eclipse.ini>파일 상단에 아래 내용

```
-vm
C:₩Program Files₩Java₩jdk1.8.0_201₩bin₩javaw.exe
```

JDK 경로₩bin₩javaw.exe

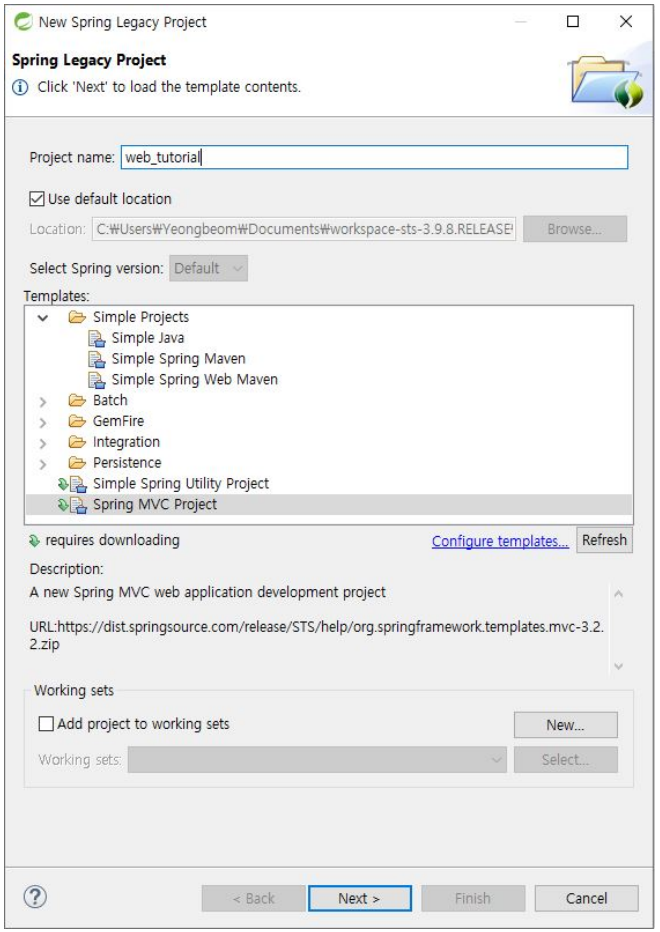
Spring 프로젝트 생성

- 프로젝트 생성
  - STS 실행 후 [File]->[New]->[Spring Legacy Project] 클릭

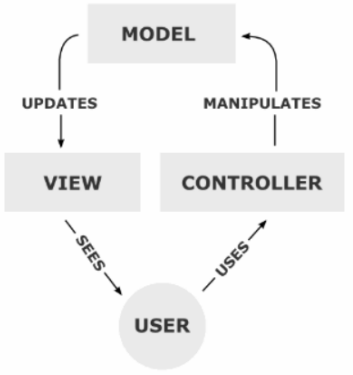


Spring 프로젝트 생성

- 프로젝트 생성
  - 프로젝트 이름 입력 "web_tutorial"
  - Spring MVC Project 선택

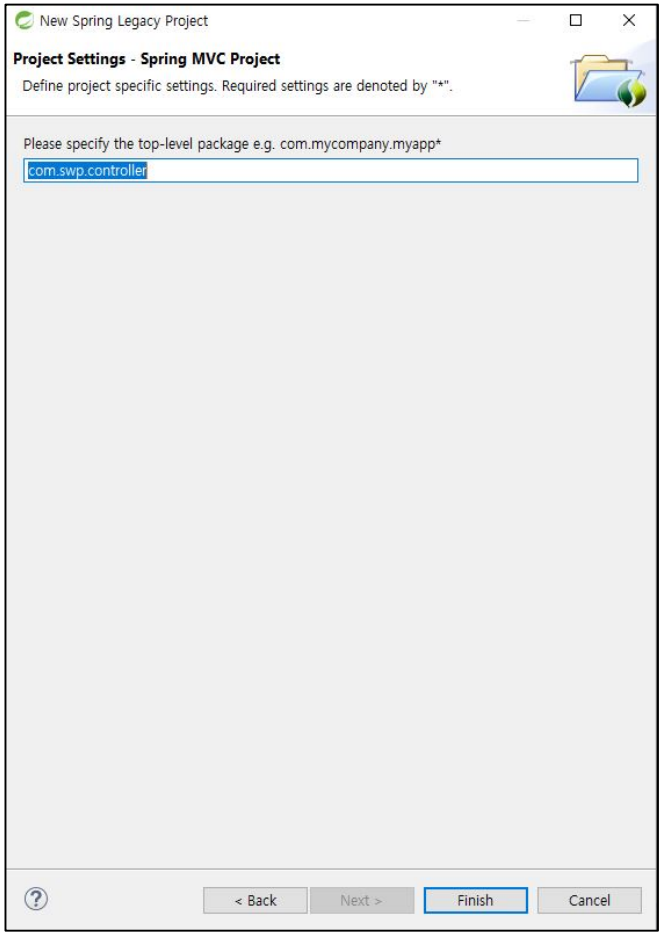


- 모델(Model) :데이터를 처리하는 영역
- 뷰(View) : 결과화면을 만들어 내는 데 사용하는 자원
- 컨트롤러(Controller) : 웹의 요청을 처리하며 뷰와 모델의 중간통신역할



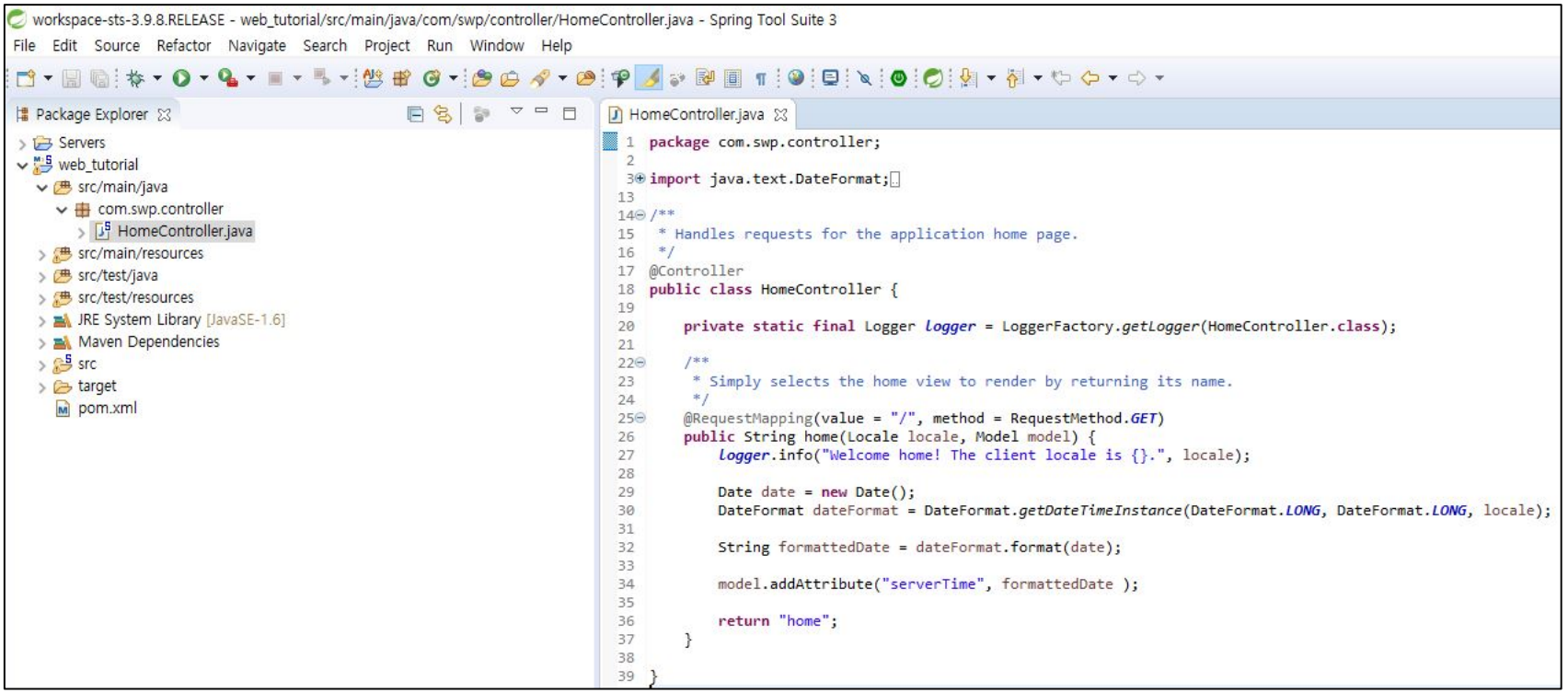
Spring 프로젝트 생성

- 프로젝트 생성
  - Package 설정
  - "com.swp.controller" 입력
  - Finish를 누르게 되면 프로젝트 생성과 동시에 Maven이 자동으로 인터넷에서 필요한 라이브러리를 다운로드
    (프로젝트 Build까지 약간의 시간이 소요)



Spring 프로젝트 생성

- MVC 패턴을 위한 기본 틀 생성



Spring 프로젝트 생성

- 프로젝트 설정 변경
  - pom.xml 파일의 properties 수정
    - Java-version 1.8로 변경
    - springframework-version 5.1.6.RELEASE로 변경

```
10    <properties>
11        <java-version>1.6</java-version>
12        <org.springframework-version>3.1.1.RELEASE</org.springframework-version>
13        <org.aspectj-version>1.6.10</org.aspectj-version>
14        <org.slf4j-version>1.6.6</org.slf4j-version>
15    </properties>
```

```
10    <properties>
11        <java-version>1.8</java-version>
12        <org.springframework-version>5.1.6.RELEASE</org.springframework-version>
13        <org.aspectj-version>1.6.10</org.aspectj-version>
14        <org.slf4j-version>1.6.6</org.slf4j-version>
15    </properties>
```

Spring 프로젝트 생성

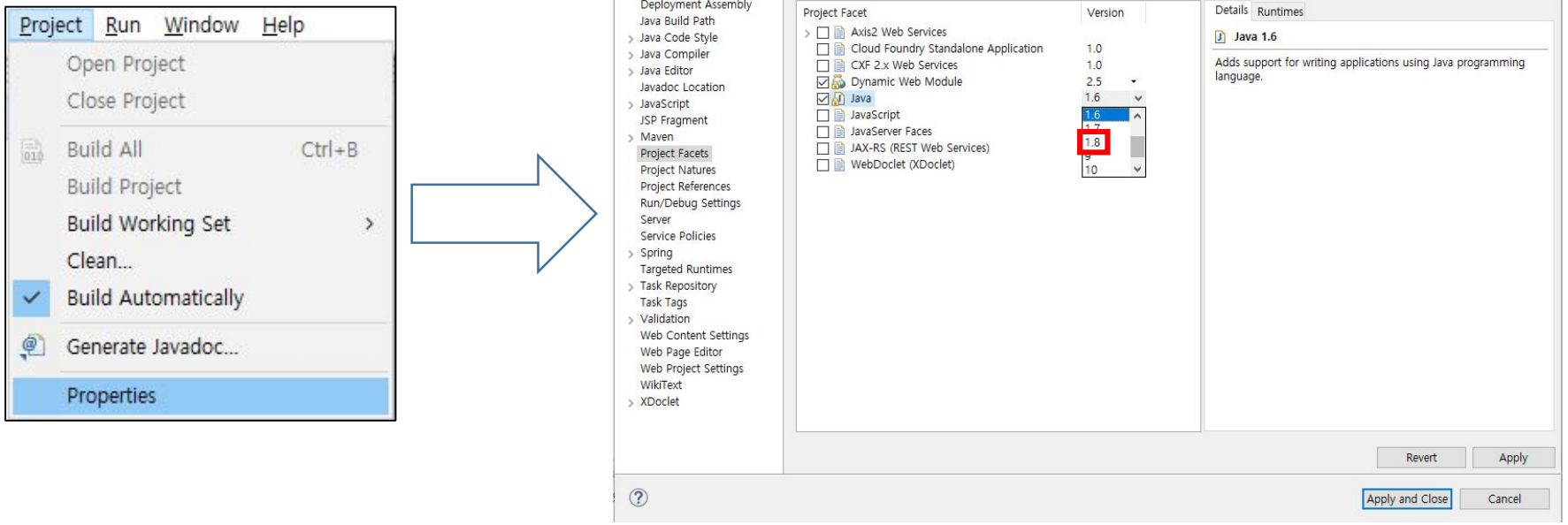
- 프로젝트 설정 변경
  - pom.xml 파일의 plugin 수정
    - configuration의 source 1.8로 변경, target 1.8로 변경

```
136        <plugin>
137            <groupId>org.apache.maven.plugins</groupId>
138            <artifactId>maven-compiler-plugin</artifactId>
139            <version>2.5.1</version>
140            <configuration>
141                <source>1.6</source>
142                <target>1.6</target>
143                <compilerArgument>-Xlint:all</compilerArgument>
144                <showWarnings>true</showWarnings>
145                <showDeprecation>true</showDeprecation>
146            </configuration>
147        </plugin>
```

```
136        <plugin>
137            <groupId>org.apache.maven.plugins</groupId>
138            <artifactId>maven-compiler-plugin</artifactId>
139            <version>2.5.1</version>
140            <configuration>
141                <source>1.8</source>
142                <target>1.8</target>
143                <compilerArgument>-Xlint:all</compilerArgument>
144                <showWarnings>true</showWarnings>
145                <showDeprecation>true</showDeprecation>
146            </configuration>
147        </plugin>
```

Spring 프로젝트 생성

- 프로젝트 설정 변경
  - [Project]->[Properties]->[Project Facets]에서 Java version 1.8로 변경



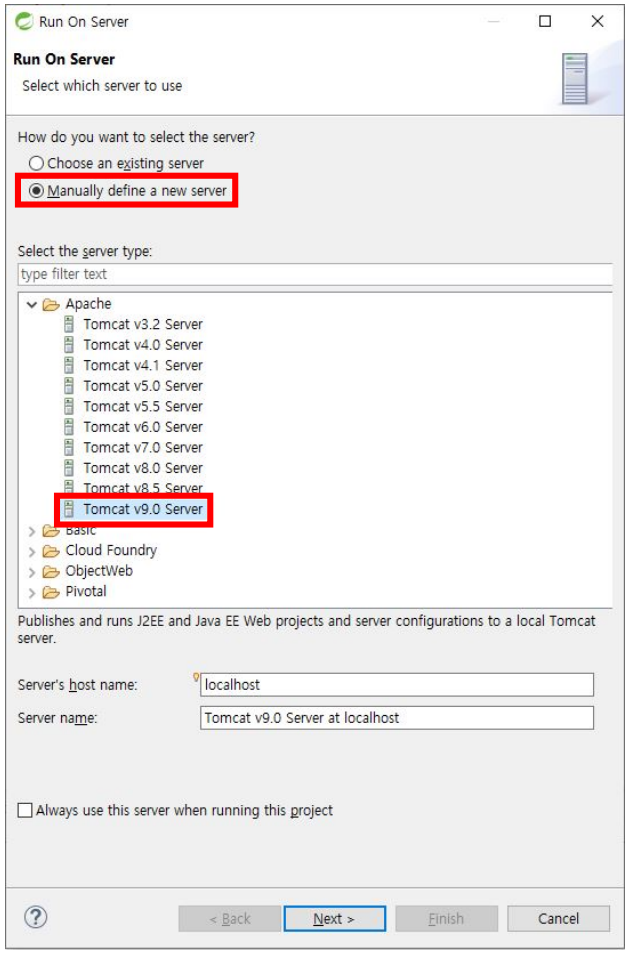
Spring 프로젝트 생성

- MVC 프로젝트 구조 분석

Spring 프로젝트 실행

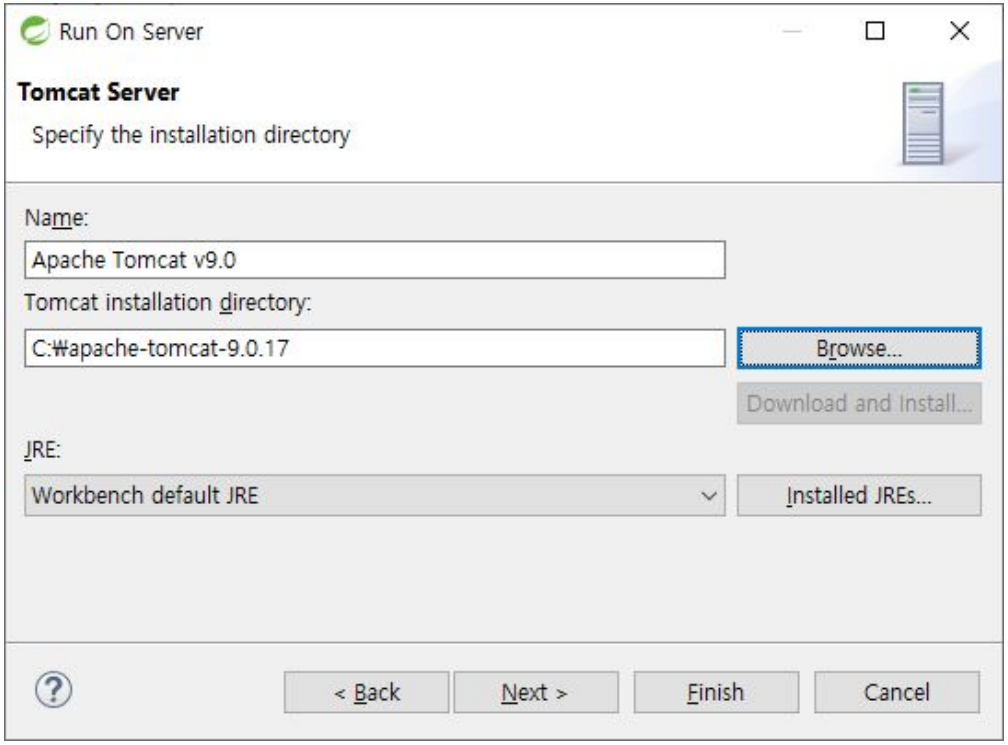
- [Run]->[Run as]->[Run on Server] 클릭

- Apache 폴더에서 이전에 다운 받은 Tomcat 9.0 선택



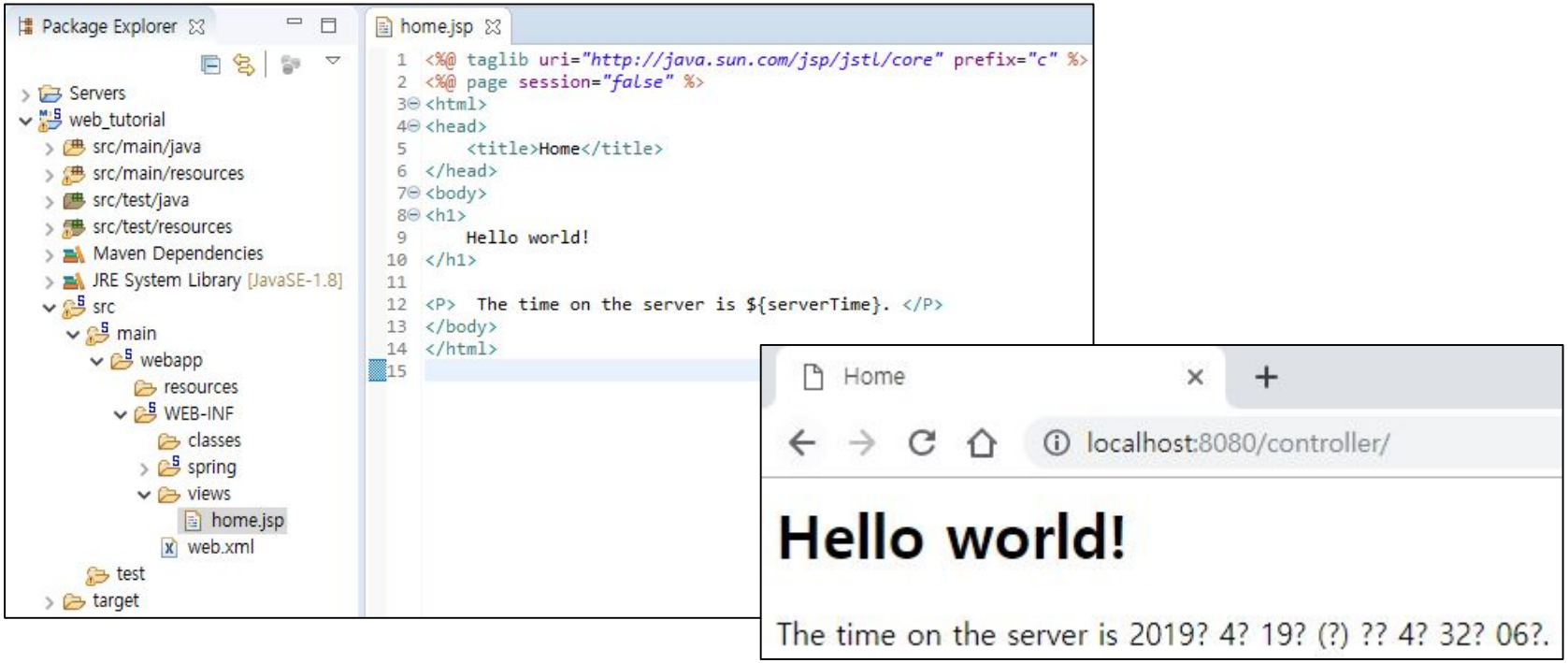
Spring 프로젝트 실행

- 이전에 압축 풀었던 Tomcat 폴더를 Browse로 등록



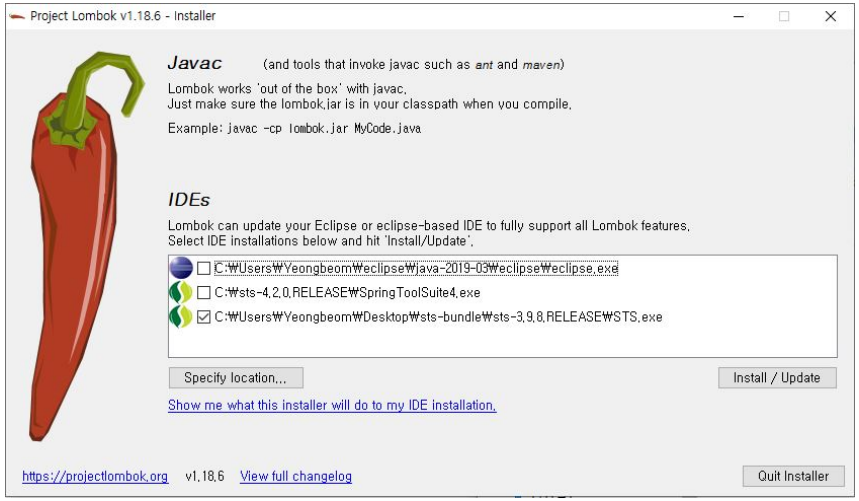
Spring 프로젝트 실행

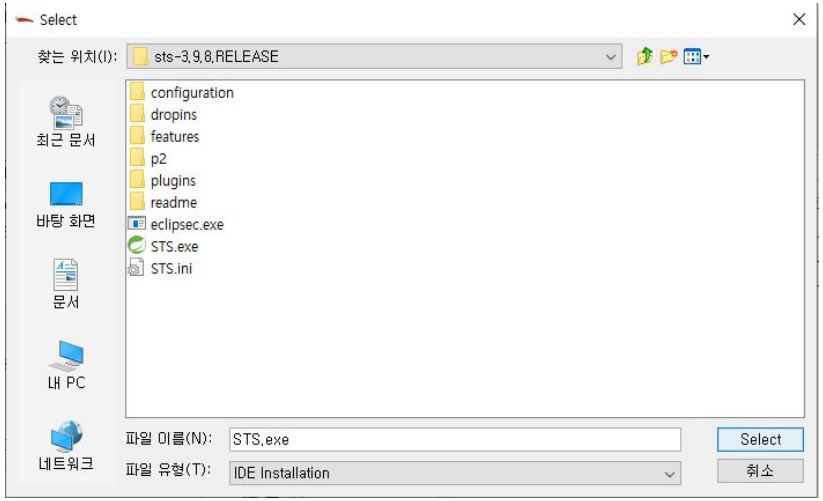
- Finish 이후 자동으로 서버가 열리고 웹 브라우저로 localhost:8080/controller/에 접속하면 views 안에 home.js가 출력됨



Lombok 설치

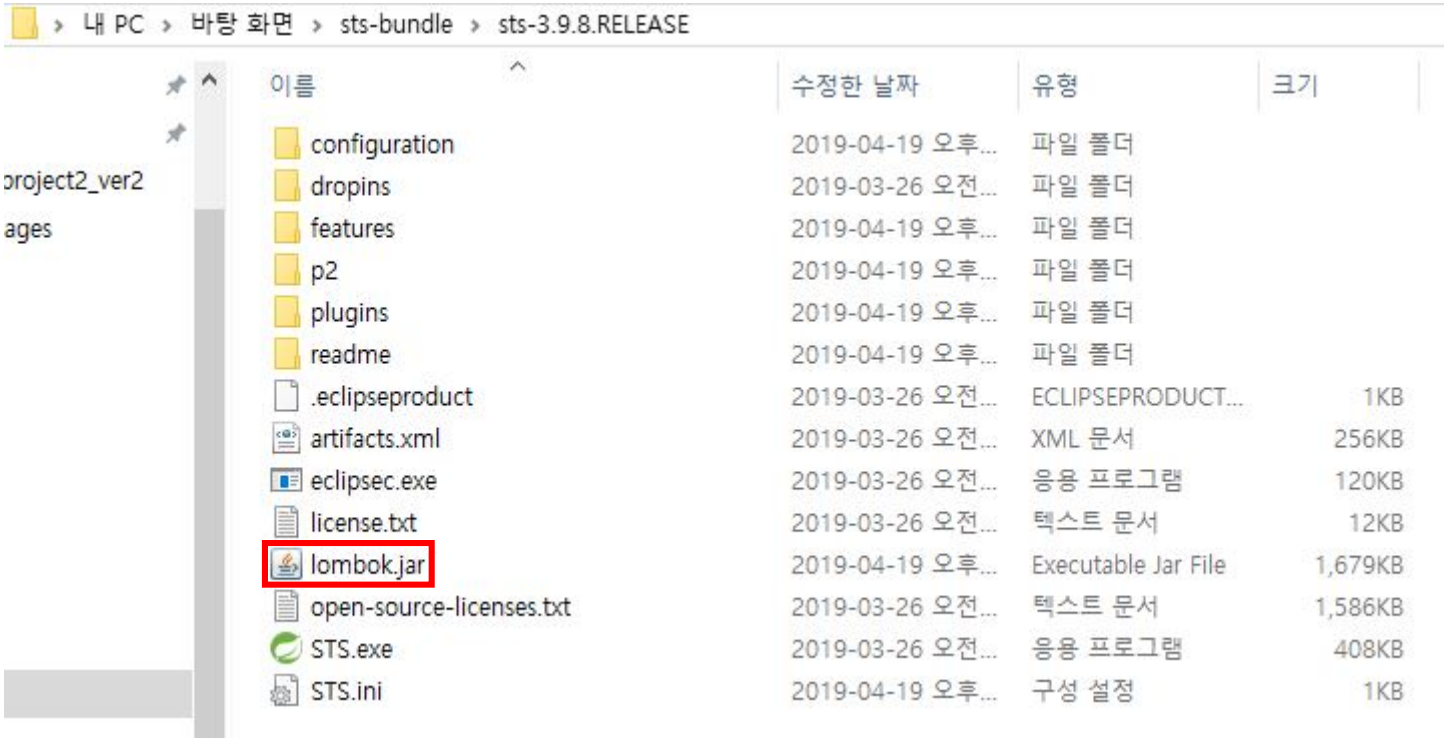
- Lombok을 이용하면 getter/setter, 생성자 등을 자동으로 생성.
  - https://projectlombok.org/download
- 설치파일 실행 후 설치될 IDE 선택, 찾지 못한다면 지정해서 설치





Lombok 설치

- 설치 완료 후 STS 실행 경로에 Lombok.jar 파일이 추가됨을 확인



의존성 주입 테스트

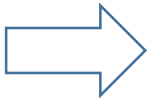
- pom.xml 추가
  - 아래 라이브러리들 추가
    - 기존 Log4j 라이브러리는 1.2.15로 설정되어 있으므로 아래와 같이 1.2.17 버전을 추가하고 기존 1.2.15 부분은 주석처리

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.6</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

의존성 주입 테스트

- pom.xml 변경

```
<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.7</version>
    <scope>test</scope>
</dependency>
```



```
<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

- 예제 클래스 생성
  - 프로젝트에 com.swp.sample 패키지를 생성하고, Restaurant 클래스와 Chef 클래스를 생성

의존성 주입 테스트

- Chef 클래스를 다음과 같이 작성

Chef.java | Restaurant.java

```
1  package com.swp.sample;
2
3  import org.springframework.stereotype.Component;
4
5  import lombok.Data;
6
7  @Component
8  @Data
9  public class Chef {
10
11 }
```

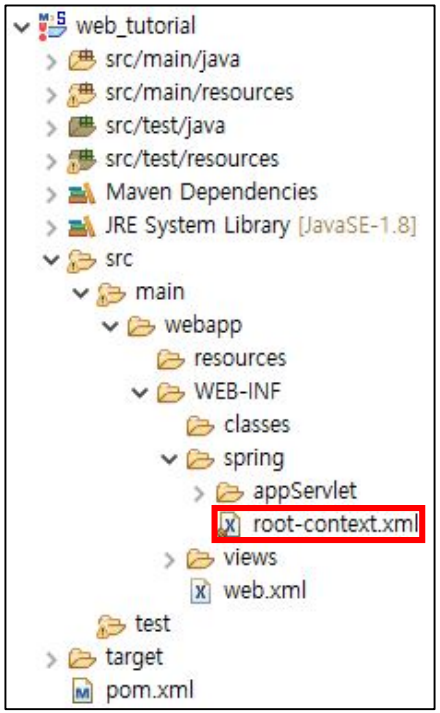
- Restaurant 클래스를 다음과 같이 작성

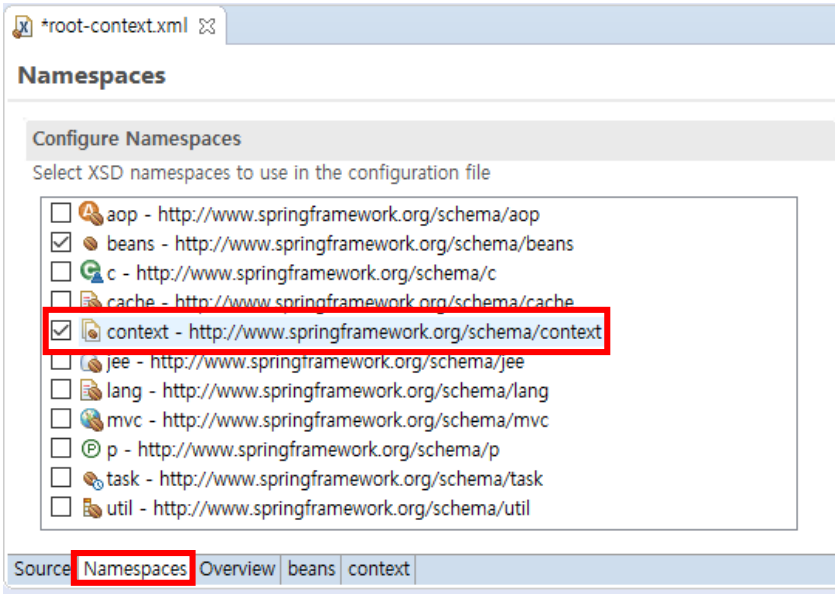
Chef.java | Restaurant.java

```
1  package com.swp.sample;
2
3  import org.springframework.beans.factory.annotation.Autowired;
4  import org.springframework.stereotype.Component;
5
6  import lombok.Data;
7  import lombok.Setter;
8
9  @Component
10 @Data
11 public class Restaurant {
12
13     @Setter(onMethod_ = @Autowired)
14     private Chef chef;
15 }
```

XML을 이용하는 의존성 주입 설정

- 스프링에서 관리되는 객체를 'Bean'이라 함.
- Src 폴더 내에 'root-context.xml'은 스프링 프레임워크에서 관리해야 하는 객체(Bean)을 설정하는 설정파일
- 'root-context.xml' 클릭 후 아래 Namespaces탭에서 'context'항목 체크





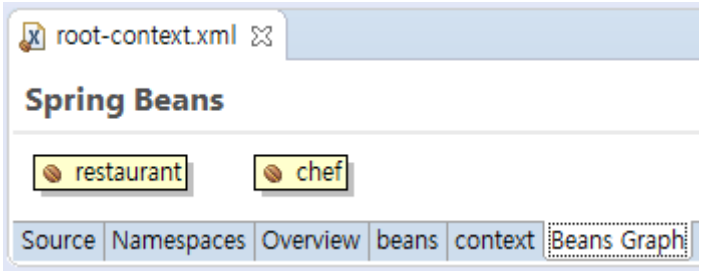
XML을 이용하는 의존성 주입 설정

- 'Source'탭을 선택해서 아래의 코드를 추가

root-context.xml

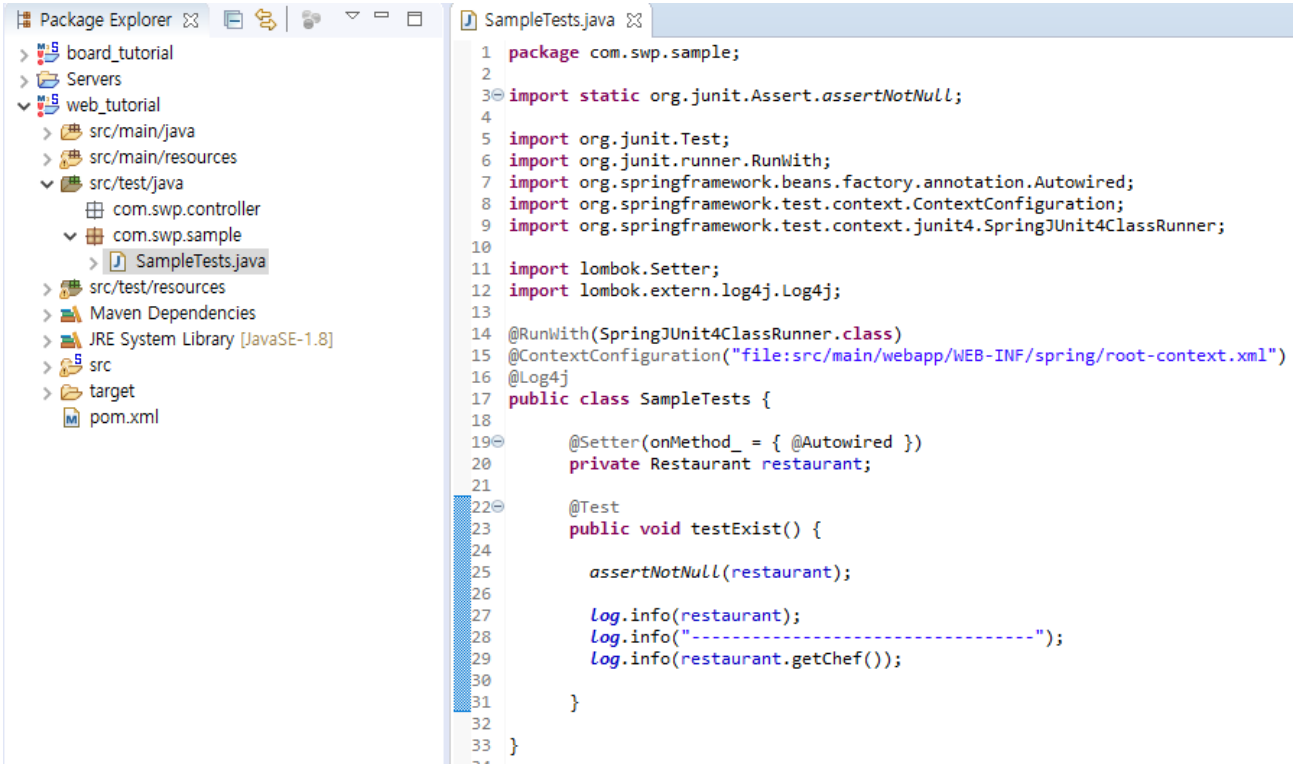
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:context="http://www.springframework.org/schema/context"
5      xsi:schemaLocation="http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd
6          http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd">
7
8      <!-- Root Context: defines shared resources visible to all other web components -->
9
10     <context:component-scan base-package="com.swp.sample">
11     </context:component-scan>
12
13 </beans>
```

- XML 저장 후 'Bean Graph' 탭을 선택하면 Restaurant와 Chef 객체가 설정된 것을 확인 가능
  - 'Bean Graph' 탭이 안보인다면 root-context.xml파일 우클릭 후 Spring- Add as Bean Configuration 선택



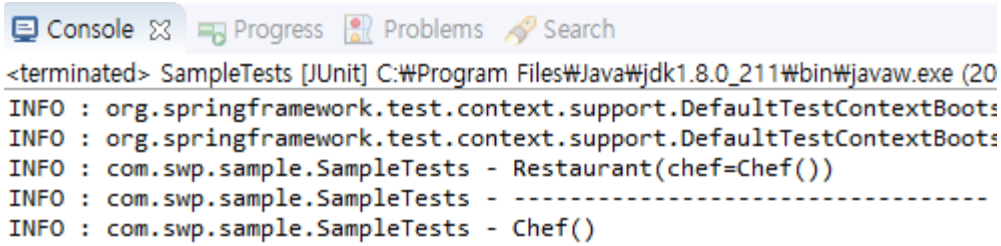
테스트 코드를 통한 확인

- 프로젝트 내 'src/test/java' 폴더 내에 'com.swp.sample.SampleTests' 클래스를 추가 후 아래 코드 입력

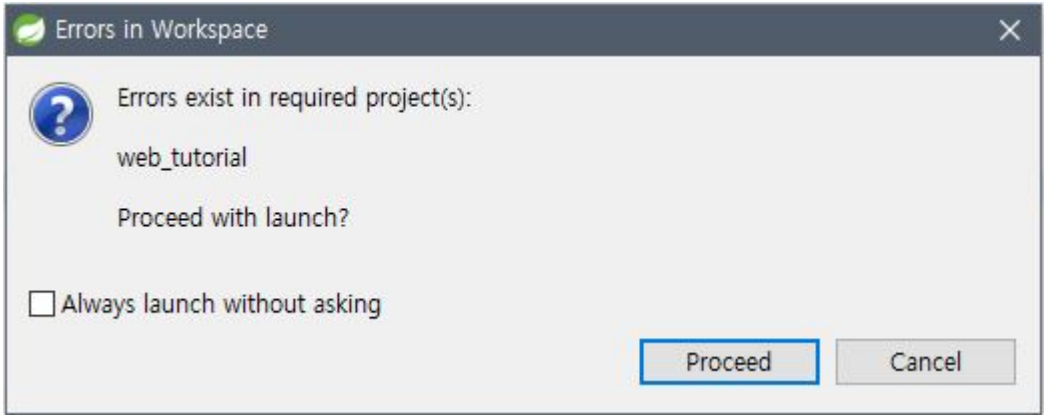


테스트 코드를 통한 확인

- 마우스 우클릭 후 'Run As – Junit Test'를 눌러 테스트 결과를 확인

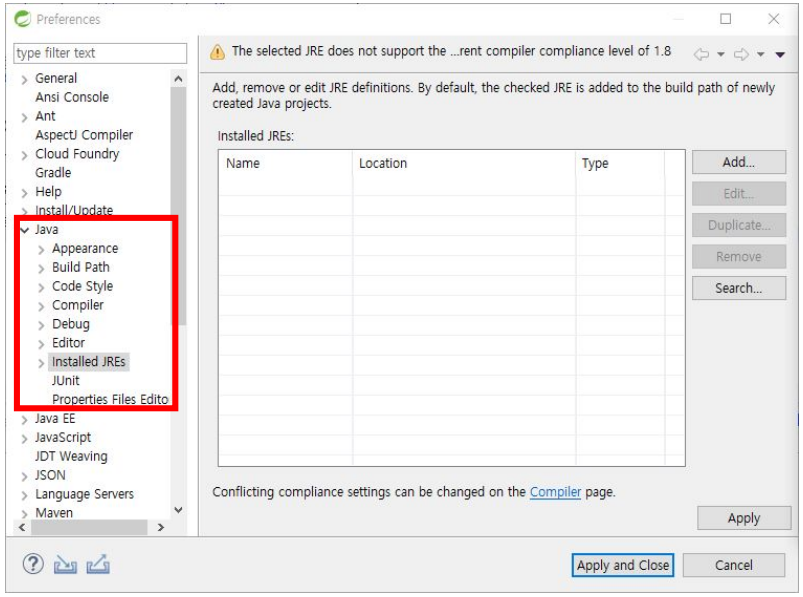


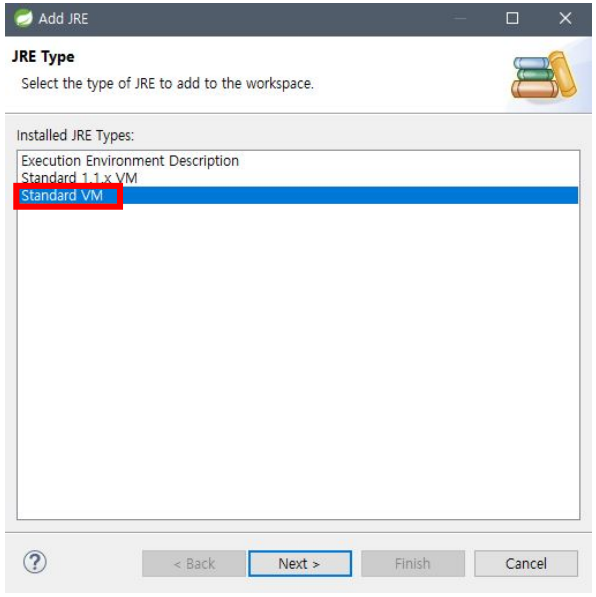
- 만일 아래와 같은 에러 창이 출력된다면 화면 상단에 있는 'Project – Properties' 클릭



테스트 코드를 통한 확인

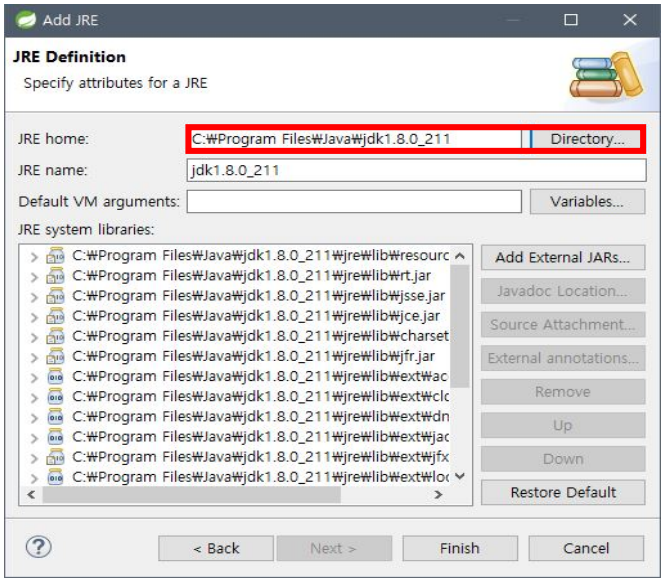
- ‘Preferences’에서 ‘Java–Installed JREs–Add’ 클릭 후 Standard VM 선택



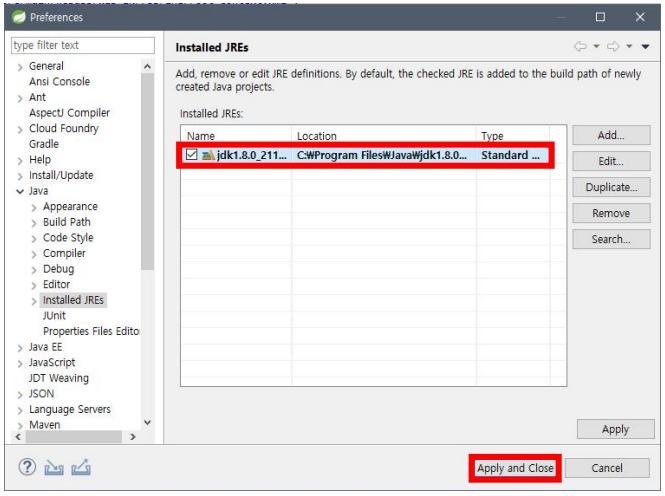


테스트 코드를 통한 확인

- JDK가 설치된 폴더 지정 후 Finish



- JDK 선택 후 Apply and Close



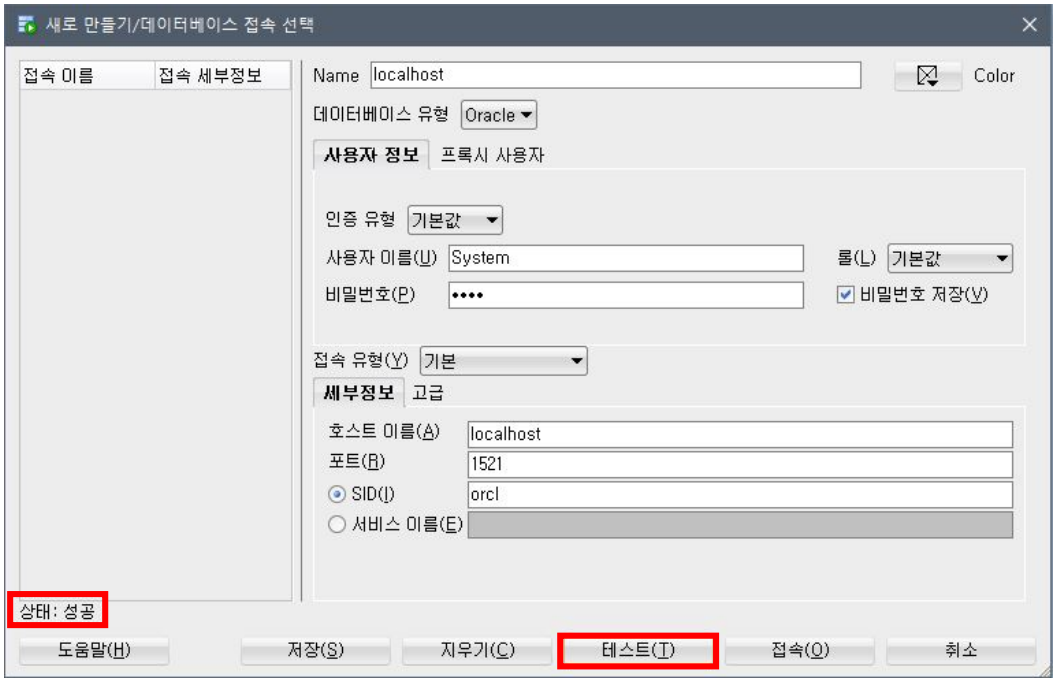
Oracle / SQL Developer 설치

- Oracle Database 11g Release 2 다운로드
  - https://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html
  - 설치 과정에서 입력한 패스워드 꼭 기억하기!

- SQL Developer 다운로드
  - https://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html

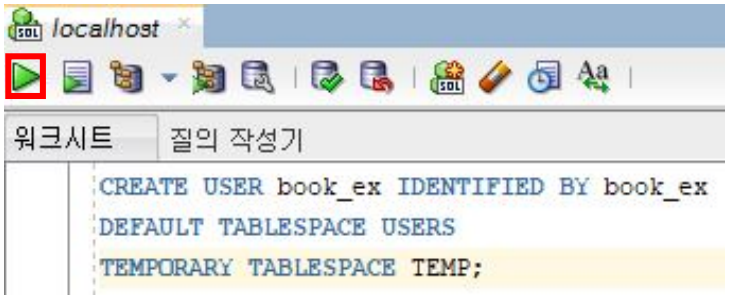
SQL Developer 접속

- 새 접속을 선택하여 아래와 같이 설정한 후 테스트 눌러 성공 확인



Database 권한 설정

- 접속 후에 보이는 텍스트 편집기에서 예제에서 사용할 사용자를 생성하기 위해 아래와 같이 SQL문을 입력한 후 실행

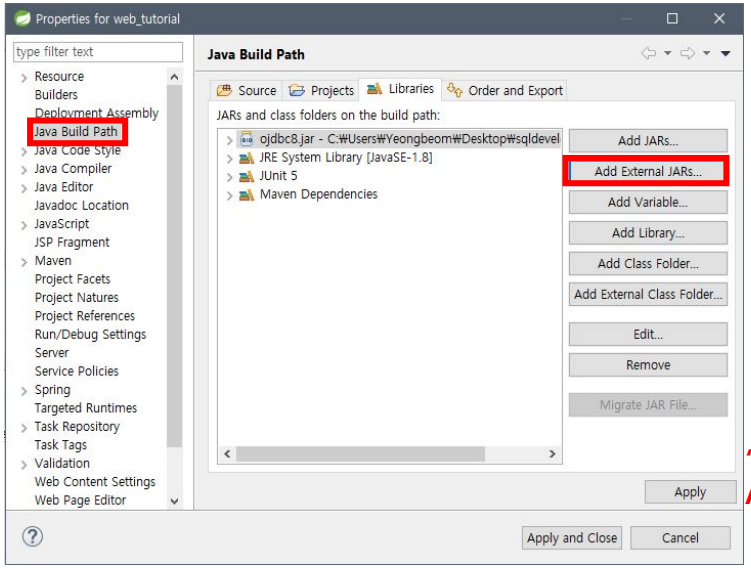


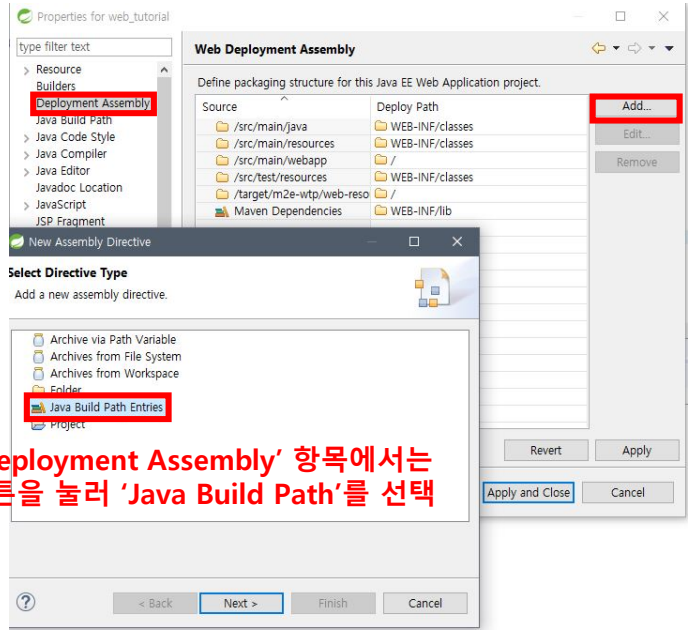
- 오라클의 사용자 계정으로 특정하기 위해 권한이 필요하므로 GRANT 구문을 이용해서 처리

```
GRANT CONNECT, DBA TO BOOK_EX;
```

프로젝트의 JDBC 연결

- Tomcat과 포트번호의 충돌을 막기 위해 오라클의 포트번호를 변경
  - 질의 창에 'exec dbms_xdb.sethttpport(9090);'를 입력하여 포트번호를 9090으로 변경
- SQL Developer 설치 경로에 jdbc/lib 폴더에 JDK 8버전용 ojdbc8.jar 확인
- 예제 프로젝트를 선택한 후 'Build Path'를 이용해서 'ojdbc8.jar' 파일을 경로에 추가 후 'Web Deployment Assembly' 항목에도 추가





'Web Deployment Assembly' 항목에서는 Add 버튼을 눌러 'Java Build Path'를 선택

프로젝트의 JDBC 연결

- src/test/java 폴더에 com.swp.persistence.JDBCTests 클래스 추가한 후 아래 코드 입력하고 'Run As - Junit Test'로 결과 확인

JDBCTests.java

```java
package com.swp.persistence;

import static org.junit.Assert.fail;

import java.sql.Connection;
import java.sql.DriverManager;

import org.junit.Test;

import lombok.extern.log4j.Log4j;

@Log4j
public class JDBCTests {

    static {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Test
    public void testConnection() {

        try (Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:ORCL", "book_ex",
                "book_ex")) {

            log.info(con);
        } catch (Exception e) {
            fail(e.getMessage());
        }
    }

}
```

- 결과
  - 데이터베이스 연결이 가능하다면 정상적으로 데이터베이스가 연결된 Connection 객체가 출력됨

```
INFO : com.swp.persistence.JDBCTests - oracle.jdbc.driver.T4CConnection@6d2a209c
```

커넥션 풀 설정

- 여러 명의 사용자를 동시에 처리하기 위해 커넥션 풀(Connection Pool) 사용

- 이번 과제에서는 HikariCP 사용

- pom.xml 파일을 수정해서 HikariCP를 추가

```
<!-- https://mvnrepository.com/artifact/com.zaxxer/HikariCP -->
<dependency>
    <groupId>com.zaxxer</groupId>
    <artifactId>HikariCP</artifactId>
    <version>2.7.4</version>
</dependency>
```

커넥션 풀 설정

- root-context. xml 파일을 수정해서 HikariCP를 추가

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <!-- Root Context: defines shared resources visible to all other web components -->

    <bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
        <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"></property>
        <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:ORCL"></property>

        <property name="username" value="book_ex"></property>
        <property name="password" value="book_ex"></property>
    </bean>

    <!-- HikariCP configuration -->
    <bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource"
        destroy-method="close">
    <constructor-arg ref="hikariConfig" />
    </bean>

    <context:component-scan base-package="com.swp.sample">
    </context:component-scan>

</beans>
```

커넥션 풀 설정

- src/test/java 폴더에 com.swp.persistence.DataSourceTests.java 클래스 추가한 후 아래 코드 입력하고 'Run As - Junit Test'로 결과 확인

```java
package com.swp.persistence;

import static org.junit.Assert.fail;

import java.sql.Connection;

import javax.sql.DataSource;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import lombok.Setter;
import lombok.extern.log4j.Log4j;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")

@Log4j
public class DataSourceTests {

  @Setter(onMethod_ = { @Autowired })
  private DataSource dataSource;

  @Test
  public void testConnection() {


    try (Connection con = dataSource.getConnection()){

      log.info(con);

    }catch(Exception e) {
      fail(e.getMessage());
    }
  }
}
```

- 성공시 HikariCP가 시작되고, 종료되는 로그를 확인할 수 있음

```
Console  Progress  Problems  Search
<terminated> DataSourceTests [JUnit] C:₩Program Files₩Java₩jdk1.8.0_211₩bin₩javaw.exe (2019. 4. 28. 오전 3:16:14)
INFO : org.springframework.test.context.support.DefaultTestContextBootstrapper - Loaded default TestExecutionListener class names from locatio
INFO : org.springframework.test.context.support.DefaultTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.conte
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
WARN : com.zaxxer.hikari.util.DriverDataSource - Registered driver with driverClassName=oracle.jdbc.driver.OracleDriver was not found, trying
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
INFO : com.swp.persistence.DataSourceTests - HikariProxyConnection@1638435724 wrapping oracle.jdbc.driver.T4CConnection@e3b3b2f
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

MyBatis와 스프링 연동

- MyBatis란?
  - SQL 매핑(mapping) 프레임워크로 JDBC 코드의 복잡함을 피하기 위해 사용
- MyBatis 관련 라이브러리 추가
  - MyBatis와 mybatis-spring을 사용하기 위해서 pome.xml파일에 추가적인 라이브러리 설정

```
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.2</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
```

MyBatis와 스프링 연동

- SQLSessionFactory
  - Root-context.xml 파일수정

```
18      <!-- HikariCP configuration -->
19      <bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource"
20          destroy-method="close">
21      <constructor-arg ref="hikariConfig" />
22      </bean>
23
24      <bean id="sqlSessionFactory"
25          class="org.mybatis.spring.SqlSessionFactoryBean">
26          <property name="dataSource" ref="dataSource"></property>
27      </bean>
```

  - DataSourceTests 클래스 수정

```
23  @Log4j
24  public class DataSourceTests {
25
26    @Setter(onMethod_ = { @Autowired })
27    private DataSource dataSource;
28
29    @Setter(onMethod_ = { @Autowired })
30    private SqlSessionFactory sqlSessionFactory;
31
32    @Test
33    public void testMyBatis() {
34
35      try (SqlSession session = sqlSessionFactory.openSession();
36         Connection con = session.getConnection();
37        ) {
38
39        log.info(session);
40        log.info(con);
41
42
43      } catch (Exception e) {
44        fail(e.getMessage());
45      }
46
47    }
48
49    @Test
50    public void testConnection() {
```

MyBatis와 스프링 연동

- 로그 확인

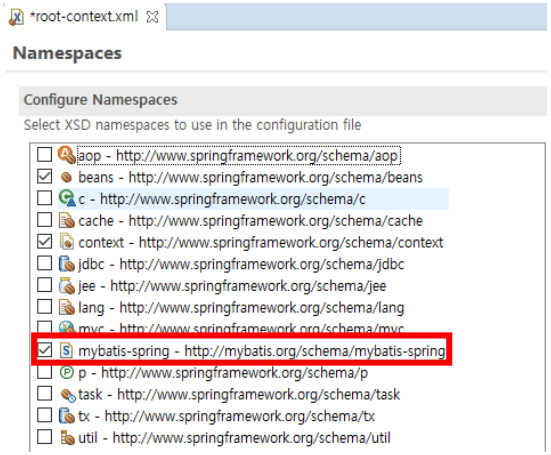
```
<terminated> DataSourceTests [JUnit] C:₩Program Files₩Java₩jdk1.8.0_211₩bin₩javaw.exe (2019. 4. 28. 오전 3:58:51)
INFO : org.springframework.test.context.support.DefaultTestContextBootstrapper - Loaded default TestExecutionListener class names from locatio
INFO : org.springframework.test.context.support.DefaultTestContextBootstrapper - Using TestExecutionListeners: [org.springframework.test.conte
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
WARN : com.zaxxer.hikari.util.DriverDataSource - Registered driver with driverClassName=oracle.jdbc.driver.OracleDriver was not found, trying
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
INFO : com.swp.persistence.DataSourceTests - HikariProxyConnection@2059592603 wrapping oracle.jdbc.driver.T4CConnection@1cf56a1c
INFO : com.swp.persistence.DataSourceTests - org.apache.ibatis.session.defaults.DefaultSqlSession@32c8e539
INFO : com.swp.persistence.DataSourceTests - HikariProxyConnection@1943855334 wrapping oracle.jdbc.driver.T4CConnection@1cf56a1c
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

MyBatis와 스프링 연동

- Mapper 인터페이스 작성
  - 'src/main/java'에 'com.swp.mapper' 패키지 생성 후 'TimeMapper'인터페이스 추가하고 아래 코드 입력

```java
package com.swp.mapper;

import org.apache.ibatis.annotations.Select;

public interface TimeMapper {

    @Select("SELECT sysdate FROM dual")
    public String getTime();

}
```

- Mapper 설정
  - root-context 파일을 열고 아래쪽의 Namespaces 항목에서 mybatis-spring탭 선택



MyBatis와 스프링 연동

- root-context.xml 파일 수정

```
26    <bean id="sqlSessionFactory"
27        class="org.mybatis.spring.SqlSessionFactoryBean">
28        <property name="dataSource" ref="dataSource"></property>
29    </bean>
30
31    <mybatis-spring:scan
32    base-package="com.swp.mapper" />
33
34    <context:component-scan base-package="com.swp.sample">
35    </context:component-scan>
```

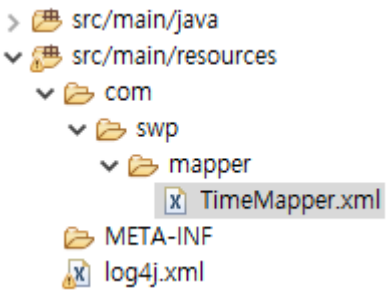
MyBatis와 스프링 연동

- Mapper 테스트
  - 'src/test/java'에 'com.swp.persistence.TimeMapperTests' 클래스 생성 후 아래 코드 입력하고 결과 확인

```
1  package com.swp.persistence;
2
3  import org.junit.Test;
4  import org.junit.runner.RunWith;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.test.context.ContextConfiguration;
7  import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
8  import com.swp.mapper.TimeMapper;
9
10 import lombok.Setter;
11 import lombok.extern.log4j.Log4j;
12
13 @RunWith(SpringJUnit4ClassRunner.class)
14 @ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
15 @Log4j
16 public class TimeMapperTests {
17
18     @Setter(onMethod_ = @Autowired)
19     private TimeMapper timeMapper;
20
21     @Test
22     public void testGetTime() {
23         log.info(timeMapper.getClass().getName());
24         log.info(timeMapper.getTime());
25     }
26
27 }
```

MyBatis와 스프링 연동

- XML 매퍼와 같이 쓰기
  - 'src/main/resources'에 com – swp – mapper로 폴더 생성 후 'TimeMapper.xml' 파일을 생성 하고 아래 코드 입력




```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.swp.mapper.TimeMapper">

  <select id="getTime2" resultType="string">
  SELECT sysdate FROM dual
  </select>

</mapper>
```

  - TimeMapper 인터페이스 수정

```java
package com.swp.mapper;

import org.apache.ibatis.annotations.Select;

public interface TimeMapper {

    @Select("SELECT sysdate FROM dual")
    public String getTime();

    public String getTime2();

}
```

MyBatis와 스프링 연동

- XML 매퍼와 같이 쓰기
  - TimeMapperTests 클래스 수정 후 결과 확인
    - getTime2() 테스트 코드의 결과는 getTime()와 동일

```
public class TimeMapperTests {

    @Setter(onMethod_ = @Autowired)
    private TimeMapper timeMapper;

    @Test
    public void testGetTime() {
        log.info(timeMapper.getClass().getName());
        log.info(timeMapper.getTime());
    }

    @Test
    public void testGetTime2() {
        log.info("getTime2");
        log.info(timeMapper.getTime2());
    }

}
```

```
INFO : org.springframework.test.context.support.DefaultTestContextBootstrapper -
INFO : org.springframework.test.context.support.DefaultTestContextBootstrapper -
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
WARN : com.zaxxer.hikari.util.DriverDataSource - Registered driver with driverCla
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
INFO : com.swp.persistence.TimeMapperTests - getTime2
INFO : com.swp.persistence.TimeMapperTests - 2019-04-28 04:51:00
INFO : com.swp.persistence.TimeMapperTests - com.sun.proxy.$Proxy25
INFO : com.swp.persistence.TimeMapperTests - 2019-04-28 04:51:00
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

MyBatis와 스프링 연동

- log4jdbc-log4j2 설정
  - Mybatis는 JDBC와 같이 SQL에 전달되는 파라미터가 '?'로 치환되어 처리
  - '?'값을 확인하기 위해 'log4jdbc-log4j2' 라이브러리 사용
  - pom.xml 파일 수정

```
<!-- https://mvnrepository.com/artifact/org.bgee.log4jdbc-log4j2/log4jdbc-log4j2-jdbc4 -->
<dependency>
    <groupId>org.bgee.log4jdbc-log4j2</groupId>
    <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
    <version>1.16</version>
</dependency>
```

  - src/main/resources 밑에 log4jdbc.log4j2.properties 파일 추가 및 코드 입력

*log4jdbc.log4j2.properties

```
1 log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
```

MyBatis와 스프링 연동

- log4jdbc-log4j2 설정
  - log4jdbc를 이용하는 경우 JDBC 드라이버와 URL 정보를 수정해야 함
  - root-context.xml 파일 수정

```
12    <bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
13        <!--
14        <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"></property>
15        <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:ORCL"></property>
16        -->
17
18        <property name="driverClassName"
19            value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></property>
20        <property name="jdbcUrl"
21            value="jdbc:log4jdbc:oracle:thin:@localhost:1521:ORCL"></property>
22
23        <property name="username" value="book_ex"></property>
24        <property name="password" value="book_ex"></property>
25    </bean>
```

  - 테스트 코드 실행 수 이전과 달리 JDBC와 관련된 로그들의 출력 확인 가능

```
INFO : jdbc.resultset - 1. ResultSet.getString(SYSDATE) returned 2019-04-28 05:14:16
INFO : jdbc.resultset - 1. ResultSet.wasNull() returned false
INFO : jdbc.resultsettable -
|--------------------|
|sysdate             |
|--------------------|
|2019-04-28 05:14:16 |
|--------------------|

INFO : jdbc.resultset - 1. ResultSet.next() returned false
INFO : jdbc.resultset - 1. ResultSet.close() returned void
INFO : jdbc.audit - 1. Connection.getMetaData() returned oracle.jdbc.driver.OracleDat
```

MyBatis와 스프링 연동

- log4jdbc-log4j2 설정
  - 로그의 양을 조절하기 위해 로그의 레벨을 수정
  - src/test/resources 밑에 log4j.xml 파일 수정

```
35    <logger name="jdbc.audit">
36        <level value="warn" />
37    </logger>
38
39    <logger name="jdbc.resultset">
40        <level value="warn" />
41    </logger>
42
43    <logger name="jdbc.connection">
44        <level value="warn" />
45    </logger>
```

  - 이후 테스트 코드를 실행하면 로그의 수가 줄어들었음을 확인할 수 있음

광운대학교 KwangWoon University

2019 Software Project

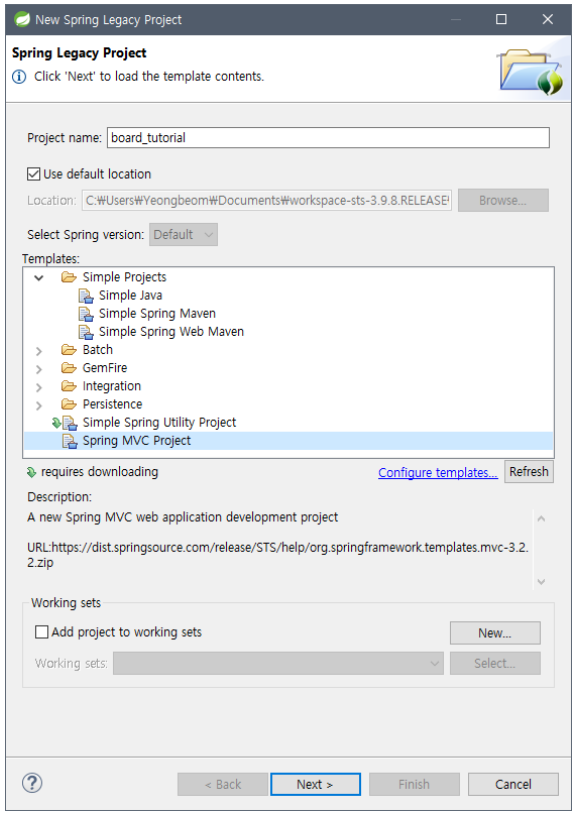
기본적인 웹 게시물 관리

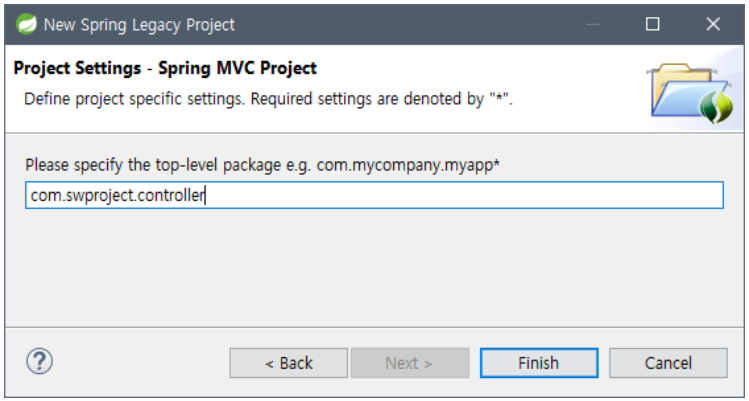
Kwangwoon Univ.
School of Computer and Information Engineering

조교 : 임영범 (ylim@kw.ac.kr)

프로젝트 구성

- 프로젝트 이름은 'board_tutorial'로 하고, 'Spring Legacy Project' - 'spring MVC Project' 생성
- 'com.swproject.controller'로 Top-level package 설정





프로젝트 구성

- 프로젝트를 생성한 후 'pom.xml' 파일을 수정하여 스프링의 버전과 Java 버전 등을 수정

```
<properties>
    <java-version>1.8</java-version>
    <org.springframework-version>5.1.6.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```

- 스프링 관련 라이브러리인 'spring-tx', 'spring-jdbc', 'spring-test' 추가

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-test</artifactId>
  <version>${org.springframework-version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>${org.springframework-version}</version>
</dependency>
dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-tx</artifactId>
  <version>${org.springframework-version}</version>
</dependency>
```

프로젝트 구성

- MyBatis를 이용하기 위해 HikariCP, MyBatis, mybatis-spring, Log4jdbc 라이브러리 추가

```
<dependency>
  <groupId>com.zaxxer</groupId>
  <artifactId>HikariCP</artifactId>
  <version>2.7.8</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.4.6</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>1.3.2</version>
</dependency>

<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
  <version>1.16</version>
</dependency>
```

- Lombok 추가 및 jUnit 버전 변경

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.6</version>
    <scope>provided</scope>
</dependency>
```

프로젝트 구성

- pom.xml에 있는 Servlet 버전 수정

```
<!-- Servlet -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
    <scope>provided</scope>
</dependency>
```

```
<!-- Servlet -->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.1.0</version>
  <scope>provided</scope>
</dependency>
```

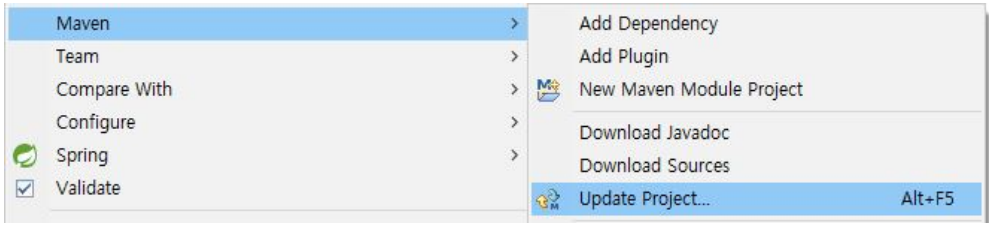
- Servlet 3.1 버전과 JDK8을 사용하기 위해 Maven관련 Java 버전을 1.8로 수정

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>2.5.1</version>
    <configuration>
        <source>1.6</source>
        <target>1.6</target>
        <compilerArgument>-Xlint:all</compilerArgument>
        <showWarnings>true</showWarnings>
        <showDeprecation>true</showDeprecation>
    </configuration>
</plugin>
```

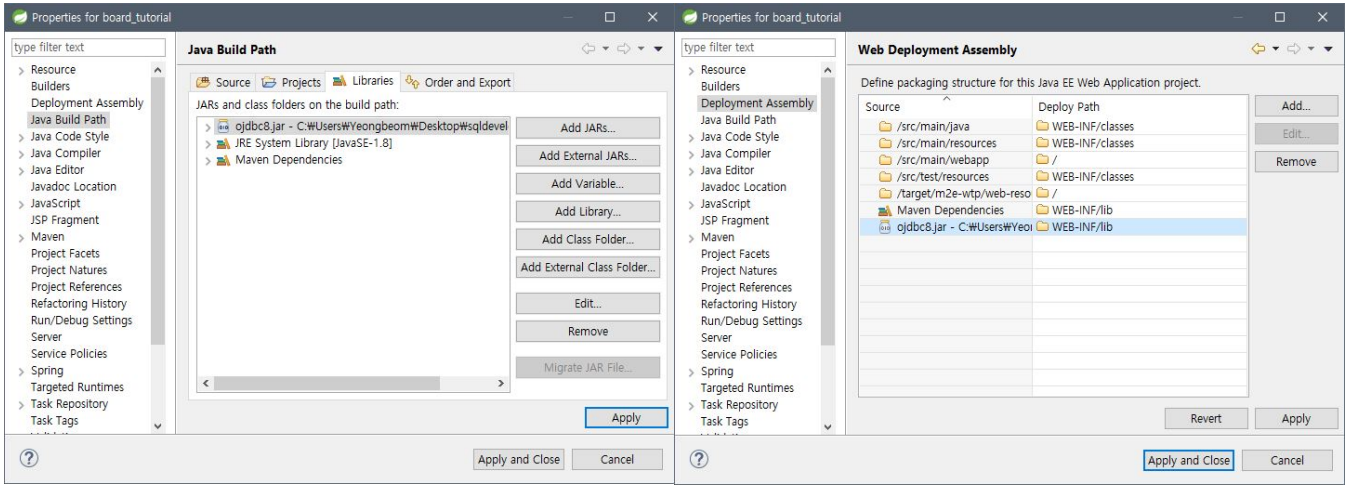
```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>2.5.1</version>
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <compilerArgument>-Xlint:all</compilerArgument>
        <showWarnings>true</showWarnings>
        <showDeprecation>true</showDeprecation>
    </configuration>
</plugin>
```

프로젝트 구성

- ‘board_tutorial’ 프로젝트를 선택하고 Maven – Update Project 실행

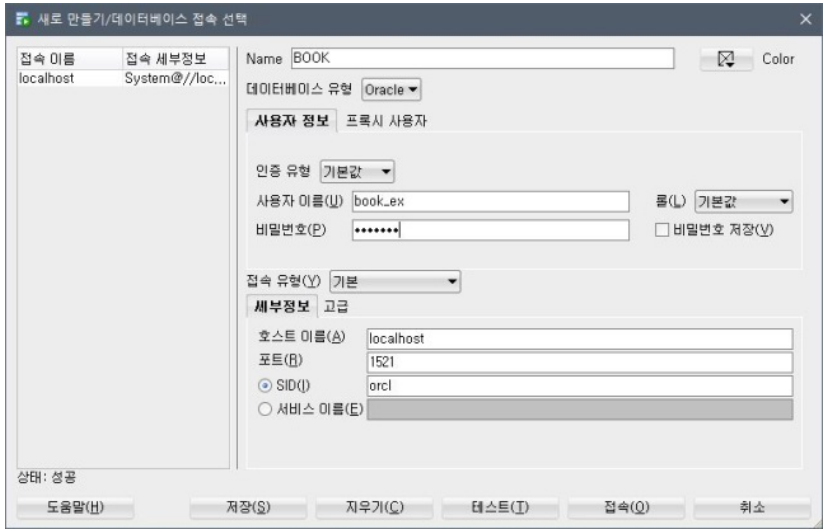


- Oracle JDBC Driver를 프로젝트의 Build Path에 추가하고, Deployment Assembly에도 추가



프로젝트 구성

- SQL Developer에서 새로운 계정 생성 후 테이블 생성
  - 사용자 이름 / 비밀번호: book_ex




```
create sequence seq_board;

create table tbl_board (
  bno number(10,0),
  title varchar2(200) not null,
  content varchar2(2000) not null,
  writer varchar2(50) not null,
  regdate date default sysdate,
  updatedate date default sysdate
);

alter table tbl_board add constraint pk_board
primary key (bno);
```

- 더미 데이터 추가 후 commit

```
insert into tbl_board (bno, title, content, writer)
values (seq_board.nextval, '테스트 제목', '테스트 내용', 'user00');

commit;
```

데이터베이스 관련 설정 및 테스트

- root-context.xml 에는 mybatis-spring 네임스페이스를 추가하고, DataSource의 설정과 MyBatis의 설정을 추가

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
    xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
        http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Root Context: defines shared resources visible to all other web components -->

    <bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">

        <property name="driverClassName"
            value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></property>
        <property name="jdbcUrl"
            value="jdbc:log4jdbc:oracle:thin:@localhost:1521:ORCL"></property>
        <property name="username" value="book_ex"></property>
        <property name="password" value="book_ex"></property>

    </bean>

    <!-- HikariCP configuration -->
    <bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource"
        destroy-method="close">
        <constructor-arg ref="hikariConfig" />
    </bean>

    <bean id="sqlSessionFactory"
        class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
    </bean>

    <mybatis-spring:scan
        base-package="com.swproject.mapper" />

</beans>
```

데이터베이스 관련 설정 및 테스트

- com.swproject.domain 패키지를 생성하고 BoardVO 클래스 정의
  - BoardVO 클래스는 Lombok을 이용(@Data 어노테이션을 적용)해서 생성자와 getter/setter, toStrng()등을 만들어 내는 방식을 사용.

```
package com.swproject.domain;

import java.util.Date;

import lombok.Data;

@Data
public class BoardVO {

  private Long bno;
  private String title;
  private String content;
  private String writer;
  private Date regdate;
  private Date updateDate;
}
```

영속 계층의 구현 준비

- com.swproject.mapper 패키지를 생성하고 BoardMapper 인터페이스 정의

```java
package com.swproject.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import com.swproject.domain.BoardVO;

public interface BoardMapper {

    @Select("select * from tbl_board where bno > 0")
    public List<BoardVO> getList();
}
```

- BoardMapper 인터페이스를 테스트 할 수 있게 테스트 환경인 'src/test/java'에 'com.swproject.mapper' 패키지를 작성하고 'BoardMapperTests' 클래스 추가

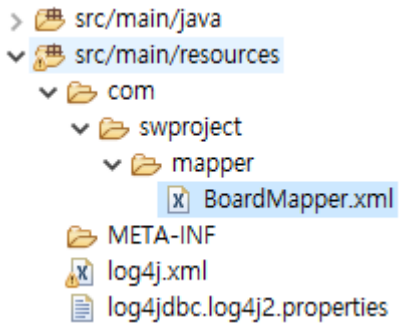
```java
package com.swproject.mapper;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import lombok.Setter;
import lombok.extern.log4j.Log4j;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
@Log4j
public class BoardMapperTests {

    @Setter(onMethod_ = @Autowired)
    private BoardMapper mapper;

    @Test
    public void testGetList() {

        mapper.getList().forEach(board -> log.info(board));

    }
}
```

영속 계층의 구현 준비

- 정상 동작시 결과 화면에 DB에 있는 데이터들이 출력됨

```
INFO : jdbc.resultset - 1. ResultSet.wasNull() returned false
INFO : jdbc.resultsettable -
|----|-------|--------|-------|---------------------|---------------------|
|bno |title  |content |writer |regdate              |updatedate           |
|----|-------|--------|-------|---------------------|---------------------|
|2   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:22.0 |2019-04-28 15:16:22.0 |
|3   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:33.0 |2019-04-28 15:16:33.0 |
|4   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:34.0 |2019-04-28 15:16:34.0 |
|5   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:35.0 |2019-04-28 15:16:35.0 |
|6   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:36.0 |2019-04-28 15:16:36.0 |
|----|-------|--------|-------|---------------------|---------------------|

INFO : jdbc.resultset - 1. ResultSet.next() returned false
INFO : jdbc.resultset - 1. ResultSet.close() returned void
```

- src/main/resources 내에 com/swproject/mapper 단계의 폴더를 생성하고 BoardMapper.xml 파일을 작성 후 아래 코드 작성




```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.swproject.mapper.BoardMapper">

    <select id="getList" resultType="com.swproject.domain.BoardVO">
        <![CDATA[
            select * from tbl_board where bno > 0
        ]]>
    </select>

</mapper>
```

영속 계층의 구현 준비

- XML에 SQL문이 처리되었으니 BoardMapper 인터페이스에 SQL 제거

```java
package com.swproject.mapper;

import java.util.List;

public interface BoardMapper {

    //@Select("select * from tbl_board where bno > 0")
    public List<BoardVO> getList();
}
```

- 결과 확인

```
INFO : jdbc.resultset - 1. ResultSet.wasNull() returned false
INFO : jdbc.resultsettable -
|----|-------|--------|-------|----------------------|----------------------|
|bno |title  |content |writer |regdate               |updatedate            |
|----|-------|--------|-------|----------------------|----------------------|
|2   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:22.0 |2019-04-28 15:16:22.0 |
|3   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:33.0 |2019-04-28 15:16:33.0 |
|4   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:34.0 |2019-04-28 15:16:34.0 |
|5   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:35.0 |2019-04-28 15:16:35.0 |
|6   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:36.0 |2019-04-28 15:16:36.0 |
|----|-------|--------|-------|----------------------|----------------------|

INFO : jdbc.resultset - 1. ResultSet.next() returned false
INFO : jdbc.resultset - 1. ResultSet.close() returned void
```

영속 영역의 CRUD 구현

- create(insert) 처리
  - BoardMapper 인터페이스에 다음과 같이 메서드를 추가 선언

```java
package com.swproject.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import com.swproject.domain.BoardVO;

public interface BoardMapper {

    //@Select("select * from tbl_board where bno > 0")
    public List<BoardVO> getList();

    public void insert(BoardVO board);

    public void insertSelectKey(BoardVO board);
}
```

  - BoardMapper.xml에 다음 내용을 추가

```xml
<insert id="insert">
    insert into tbl_board (bno,title,content,writer)
    values (seq_board.nextval, #{title}, #{content}, #{writer})
</insert>

<insert id="insertSelectKey">

    <selectKey keyProperty="bno" order="BEFORE"
        resultType="long">
        select seq_board.nextval from dual
    </selectKey>

    insert into tbl_board (bno,title,content, writer)
    values (#{bno},
    #{title}, #{content}, #{writer})
</insert>
```

영속 영역의 CRUD 구현

- create(insert) 처리
  - insert()에 대한 테스트 코드를 BoardMapperTests 클래스에 추가 작성

```java
@Test
public void testInsert() {

    BoardVO board = new BoardVO();
    board.setTitle("새로 작성하는 글");
    board.setContent("새로 작성하는 내용");
    board.setWriter("newbie");

    mapper.insert(board);

    log.info(board);
}
```

  - 실행 결과

```
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
INFO : com.swproject.mapper.BoardMapperTests - BoardVO(bno=null, title=새로 작성하는 글, content=새로 작성하는 내용, writer=newbie, regdate=null, updateDate=null)
```

영속 영역의 CRUD 구현

- create(insert) 처리
  - @SelectKey를 사용하는 경우 테스트 코드는 다음과 같음

```
@Test
public void testInsertSelectKey() {

    BoardVO board = new BoardVO();
    board.setTitle("새로 작성하는 글 select key");
    board.setContent("새로 작성하는 내용 select key");
    board.setWriter("newbie");

    mapper.insertSelectKey(board);

    log.info(board);
}
```

  - 실행 결과

```
INFO : jdbc.audit - 1. PreparedStatement.setLong(1, 8) returned
INFO : jdbc.audit - 1. PreparedStatement.setString(2, "새로 작성하는 글 select key") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(3, "새로 작성하는 내용 select key") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(4, "newbie") returned
INFO : jdbc.sqlonly - insert into tbl_board (bno,title,content, writer) values (8, '새로 작성하는 글 select key', '새로 작성하는
내용 select key', 'newbie')

INFO : jdbc.sqltiming - insert into tbl_board (bno,title,content, writer) values (8, '새로 작성하는 글 select key', '새로 작성하는
내용 select key', 'newbie')
 {executed in 2 msec}
```

영속 영역의 CRUD 구현

- read(select) 처리
  - BoardMapper 인터페이스에 다음과 같이 메서드를 추가 선언

```
public BoardVO read(Long bno);
```

  - BoardMapper.xml에 다음 내용을 추가

```
<select id="read" resultType="com.swproject.domain.BoardVO">
    select * from tbl_board where bno = #{bno}
</select>
```

  - BoardMapperTests 클래스에 다음 내용을 추가

```
@Test
public void testRead() {

    // 존재하는 게시물 번호로 테스트
    BoardVO board = mapper.read(5L);

    log.info(board);

}
```

  - 실행 결과

```
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
INFO : com.swproject.mapper.BoardMapperTests - BoardVO(bno=5, title=테스트 제목, content=테스트 내용, writer=user00, regdate=Sun Apr 28 15:16:35
INFO : jdbc.audit - 1. Connection.getAutoCommit() returned true
INFO : jdbc.audit - 1. PreparedStatement.new PreparedStatement returned
```

영속 영역의 CRUD 구현

- delete 처리
  - BoardMapper 인터페이스에 다음과 같이 메서드를 추가 선언

```
public int delete(Long bno);
```

  - BoardMapper.xml에 다음 내용을 추가

```
<delete id="delete">
    delete tbl_board where bno = #{bno}
</delete>
```

  - BoardMapperTests 클래스에 다음 내용을 추가

```
@Test
public void testDelete() {

    log.info("DELETE COUNT: " + mapper.delete(3L));
}
```

  - 실행 결과

```
INFO : jdbc.sqltiming - delete tbl_board where bno = 3
 {executed in 1 msec}
INFO : jdbc.audit - 1. PreparedStatement.execute() returned false
INFO : jdbc.audit - 1. PreparedStatement.getUpdateCount() returned 1
```

영속 영역의 CRUD 구현

- update 처리
  - BoardMapper 인터페이스에 다음과 같이 메서드를 추가 선언

```
public int update(BoardVO board);
```

  - BoardMapper.xml에 다음 내용을 추가

```
<update id="update">
    update tbl_board
    set title= #{title},
    content=#{content},
    writer = #{writer},
    updateDate = sysdate
    where bno =
    #{bno}
</update>
```

영속 영역의 CRUD 구현

- update 처리
  - BoardMapperTests 클래스에 다음 내용을 추가

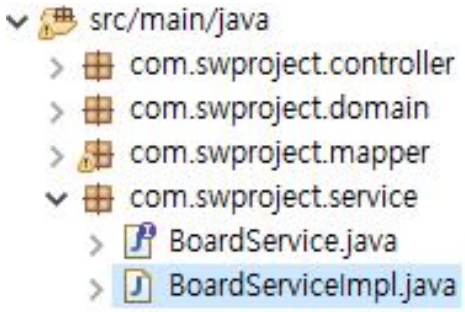
```
@Test
public void testUpdate() {

    BoardVO board = new BoardVO();
    // 실행전 존재하는 번호인지 확인할 것
    board.setBno(5L);
    board.setTitle("수정된 제목");
    board.setContent("수정된 내용");
    board.setWriter("user00");

    int count = mapper.update(board);
    log.info("UPDATE COUNT: " + count);

}
```

  - 실행 결과

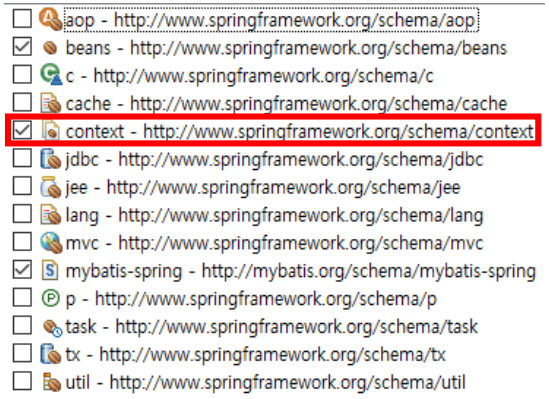
```
INFO : jdbc.sqlonly - update tbl_board set title= '수정된 제목', content='수정된 내용', writer = 'user00', updateDate = sysdate
where bno = 5

INFO : jdbc.sqltiming - update tbl_board set title= '수정된 제목', content='수정된 내용', writer = 'user00', updateDate = sysdate
where bno = 5
 {executed in 2 msec}
INFO : jdbc.audit - 1. PreparedStatement.execute() returned false
INFO : jdbc.audit - 1. PreparedStatement.getUpdateCount() returned 1
INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false
INFO : jdbc.audit - 1. PreparedStatement.close() returned
```

비즈니스 계층의 설정

- 비즈니스 계층을 위해서 com.swproject.service라는 패키지 작성하고 BoardService 인터페이스와 BoardServiceImpl 클래스 선언



- root-context.xml의 Namespaces에서 context 항목 추가 후 Source에 우측 코드 입력




```
<context:component-scan
    base-package="com.swproject.service"></context:component-scan>
```

등록 작업의 구현과 테스트

- BoardService 인터페이스 작성

```
package com.swproject.service;

import java.util.List;

import com.swproject.domain.BoardVO;

public interface BoardService {

    public void register(BoardVO board);

    public BoardVO get(Long bno);

    public boolean modify(BoardVO board);

    public boolean remove(Long bno);

    public List<BoardVO> getList();

}
```

등록 작업의 구현과 테스트

- BoardServiceImpl 클래스 작성

```
@Log4j
@Service
@AllArgsConstructor
public class BoardServiceImpl implements BoardService{

    //spring 4.3 이상에서 자동 처리
    @Setter(onMethod_ = @Autowired)
    private BoardMapper mapper;

    @Override
    public void register(BoardVO board) {

        log.info("register....." + board);

        mapper.insertSelectKey(board);
    }
    @Override
    public BoardVO get(Long bno) {
        // TODO Auto-generated method stub
        return null;
    }
    @Override
    public boolean modify(BoardVO board) {
        // TODO Auto-generated method stub
        return false;
    }
    @Override
    public boolean remove(Long bno) {
        // TODO Auto-generated method stub
        return false;
    }
    @Override
    public List<BoardVO> getList() {
        // TODO Auto-generated method stub
        return null;
    }
}
```

등록 작업의 구현과 테스트

- src/test/java밑에 com.swproject.service.BoardServiceTests 클래스 작성

```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
@Log4j
public class BoardServiceTests {

    @Setter(onMethod_ = { @Autowired })
    private BoardService service;

    @Test
    public void testExist() {

        log.info(service);
        assertNotNull(service);
    }

    @Test
    public void testRegister() {

        BoardVO board = new BoardVO();
        board.setTitle("새로 작성하는 글");
        board.setContent("새로 작성하는 내용");
        board.setWriter("newbie");

        service.register(board);

        log.info("생성된 게시물의 번호: " + board.getBno());
    }
}
```

- 실행 결과
  - 생성된 게시물의 번호를 확인할 수 있음

```
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
INFO : com.swproject.service.BoardServiceTests - 생성된 게시물의 번호: 16
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
```

목록(리스트) 작업의 구현과 테스트

- BoardServiceImpl 클래스 수정

```
@Override
public List<BoardVO> getList() {

log.info("getList.........");

return mapper.getList();
}
```

- BoardServiceTests 클래스 추가 작성

```
@Test
public void testGetList() {

    service.getList().forEach(board -> log.info(board));
}
```

- 테스트 실행 결과

```
INFO : jdbc.resultsettable -
|----|---------------------|---------------------|-------|---------------------|---------------------|
|bno |title                |content              |writer |regdate              |updatedate           |
|----|---------------------|---------------------|-------|---------------------|---------------------|
|2   |테스트 제목          |테스트 내용          |user00 |2019-04-28 15:16:22.0 |2019-04-28 15:16:22.0 |
|4   |테스트 제목          |테스트 내용          |user00 |2019-04-28 15:16:34.0 |2019-04-28 15:16:34.0 |
|5   |수정된 제목          |수정된 내용          |user00 |2019-04-28 15:16:35.0 |2019-04-28 19:15:22.0 |
|6   |테스트 제목          |테스트 내용          |user00 |2019-04-28 15:16:36.0 |2019-04-28 15:16:36.0 |
|7   |새로 작성하는 글       |새로 작성하는 내용       |newbie |2019-04-28 18:40:54.0 |2019-04-28 18:40:54.0 |
|8   |새로 작성하는 글 select key |새로 작성하는 내용 select key |newbie |2019-04-28 18:47:38.0 |2019-04-28 18:47:38.0 |
|9   |새로 작성하는 글       |새로 작성하는 내용       |newbie |2019-04-28 18:47:38.0 |2019-04-28 18:47:38.0 |
|10  |새로 작성하는 글 select key |새로 작성하는 내용 select key |newbie |2019-04-28 19:01:24.0 |2019-04-28 19:01:24.0 |
|11  |새로 작성하는 글       |새로 작성하는 내용       |newbie |2019-04-28 19:01:24.0 |2019-04-28 19:01:24.0 |
|12  |새로 작성하는 글 select key |새로 작성하는 내용 select key |newbie |2019-04-28 19:06:29.0 |2019-04-28 19:06:29.0 |
|13  |새로 작성하는 글       |새로 작성하는 내용       |newbie |2019-04-28 19:06:29.0 |2019-04-28 19:06:29.0 |
|14  |새로 작성하는 글 select key |새로 작성하는 내용 select key |newbie |2019-04-28 19:15:22.0 |2019-04-28 19:15:22.0 |
|15  |새로 작성하는 글       |새로 작성하는 내용       |newbie |2019-04-28 19:15:22.0 |2019-04-28 19:15:22.0 |
|16  |새로 작성하는 글       |새로 작성하는 내용       |newbie |2019-04-29 01:32:39.0 |2019-04-29 01:32:39.0 |
|----|---------------------|---------------------|-------|---------------------|---------------------|
```

조회 작업의 구현과 테스트

- BoardServiceImpl 클래스 수정

```
@Override
public BoardVO get(Long bno) {

    log.info("get....." + bno);

    return mapper.read(bno);

}
```

- BoardServiceTests 클래스 추가 작성

```
@Test
public void testGet() {

    log.info(service.get(1L));
}
```

- 테스트 실행 결과

```
INFO : jdbc.resultsettable -
|----|-------|--------|-------|---------------------|---------------------|
|bno |title  |content |writer |regdate              |updatedate           |
|----|-------|--------|-------|---------------------|---------------------|
|6   |테스트 제목 |테스트 내용 |user00 |2019-04-28 15:16:36.0 |2019-04-28 15:16:36.0 |
|----|-------|--------|-------|---------------------|---------------------|
```

삭제/수정 구현과 테스트

- BoardServiceImpl 클래스 수정

```
@Override
public boolean modify(BoardVO board) {

    log.info("modify....." + board);

    return mapper.update(board) == 1;
}

@Override
public boolean remove(Long bno) {

    log.info("remove...." + bno);

    return mapper.delete(bno) == 1;
}
```

- BoardServiceTests 클래스 추가 작성

```
@Test
public void testDelete() {

    // 게시물 번호의 존재 여부를 확인하고 테스트할 것
    log.info("REMOVE RESULT: " + service.remove(2L));

}

@Test
public void testUpdate() {

    BoardVO board = service.get(1L);

    if (board == null) {
        return;
    }

    board.setTitle("제목 수정합니다.");
    log.info("MODIFY RESULT: " + service.modify(board));
}
```

BoardController의 작성

- 목록에 대한 처리와 테스트
  - com.swproject.controller 패키지에 BoardController 클래스 작성

```
package com.swproject.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import com.swproject.service.BoardService;

import lombok.AllArgsConstructor;
import lombok.extern.log4j.Log4j;

@Controller
@Log4j
@RequestMapping("/board/*")
@AllArgsConstructor
public class BoardController {

    private BoardService service;

    @GetMapping("/list")
    public void list(Model model) {

    log.info("list");
    model.addAttribute("list", service.getList());

    }

}
```

BoardController의 작성

- 목록에 대한 처리와 테스트
  - src/test/java에 com.swproject.controller 패키지를 추가한 후 BoardControllerTests 클래스 작성

```
package com.swproject.controller;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;

import lombok.Setter;
import lombok.extern.log4j.Log4j;

@RunWith(SpringJUnit4ClassRunner.class)

//Test for Controller
@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 {

    @Setter(onMethod_ = { @Autowired })
    private WebApplicationContext ctx;

    private MockMvc mockMvc;

    @Before
    public void setup() {
        this.mockMvc = MockMvcBuilders.webAppContextSetup(ctx).build();
    }

    @Test
    public void testList() throws Exception {

        log.info(
                mockMvc.perform(MockMvcRequestBuilders.get("/board/list"))
                .andReturn()
                .getModelAndView()
                .getModelMap());
    }
}
```

- 테스트 실행 결과

```
INFO : com.swproject.controller.BoardControllerTests - {list=[BoardVO(bno=2, title=테스트 제목, c
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
```

BoardController의 작성

- 등록 처리와 테스트
  - BoardController 클래스 추가 작성

```
@PostMapping("/register")
public String register(BoardVO board, RedirectAttributes rttr) {

    log.info("register: " + board);

    service.register(board);

    rttr.addFlashAttribute("result", board.getBno());

    return "redirect:/board/list";
}
```

  - BoardControllerTests 클래스 추가 작성

```
@Test
public void testRegister() throws Exception {

    String resultPage = mockMvc
            .perform(MockMvcRequestBuilders.post("/board/register")
            .param("title", "테스트 새글 제목")
            .param("content", "테스트 새글 내용")
            .param("writer", "user00"))
            .andReturn().getModelAndView().getViewName();

    log.info(resultPage);

}
```

BoardController의 작성

- 등록 처리와 테스트
  - 테스트 실행 결과

```
INFO : jdbc.audit - 1. PreparedStatement.new PreparedStatement returned
INFO : jdbc.audit - 1. Connection.prepareStatement(insert into tbl_board (bno,title,content, writer)
               values (?,
               ?, ?, ?)) returned net.sf.log4jdbc.sql.jdbcapi.PreparedStatementSpy@415e0bcb
INFO : jdbc.audit - 1. PreparedStatement.setLong(1, 21) returned
INFO : jdbc.audit - 1. PreparedStatement.setString(2, "테스트 새글 제목") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(3, "테스트 새글 내용") returned
INFO : jdbc.audit - 1. PreparedStatement.setString(4, "user00") returned
INFO : jdbc.sqlonly - insert into tbl_board (bno,title,content, writer) values (21, '테스트 새글 제목', '테스트 새글 내용', 'user00')

INFO : jdbc.sqltiming - insert into tbl_board (bno,title,content, writer) values (21, '테스트 새글 제목', '테스트 새글 내용', 'user00')
 {executed in 1 msec}
INFO : jdbc.audit - 1. PreparedStatement.execute() returned false
INFO : jdbc.audit - 1. PreparedStatement.getUpdateCount() returned 1
```

BoardController의 작성

- 조회 처리와 테스트
  - BoardController 클래스 추가 작성

```
@GetMapping({ "/get" })
public void get(@RequestParam("bno") Long bno, Model model) {

    log.info("/get");
    model.addAttribute("board", service.get(bno));
}
```

  - BoardControllerTests 클래스 추가 작성

```
@Test
public void tetGet() throws Exception {

    log.info(mockMvc.perform(MockMvcRequestBuilders.get("/board/get").param("bno", "2")).andReturn()
            .getModelAndView().getModelMap());
}
```

  - 테스트 결과 확인

```
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
INFO : com.swproject.controller.BoardControllerTests - {board=BoardVO(bno=2, title=테스트 제목, content=테스트 내용,
INFO : org.springframework.mock.web.MockServletContext - Initializing Spring TestDispatcherServlet ''
INFO : org.springframework.test.web.servlet.TestDispatcherServlet - Initializing Servlet ''
INFO : org.springframework.test.web.servlet.TestDispatcherServlet - Completed initialization in 1 ms
```

BoardController의 작성

- 수정 처리와 테스트
  - BoardController 클래스 추가 작성

```
@PostMapping("/modify")
public String modify(BoardVO board, RedirectAttributes rttr) {
    log.info("modify:" + board);

    if (service.modify(board)) {
        rttr.addFlashAttribute("result", "success");
    }
    return "redirect:/board/list";
}
```

  - BoardControllerTests 클래스 추가 작성

```
@Test
public void testModify() throws Exception {

    String resultPage = mockMvc
            .perform(MockMvcRequestBuilders.post("/board/modify").param("bno", "1").param("title", "수정된 테스트 새글 제목")
                    .param("content", "수정된 테스트 새글 내용").param("writer", "user00"))
            .andReturn().getModelAndView().getViewName();

    log.info(resultPage);

}
```

  - 테스트 결과 확인

```
INFO : jdbc.audit - 1. PreparedStatement.setLong(4, 1) returned
INFO : jdbc.sqlonly - update tbl_board set title= '수정된 테스트 새글 제목', content='수정된 테스트 새글 내용', writer = 'user00', updateDate
= sysdate where bno = 1

INFO : jdbc.sqltiming - update tbl_board set title= '수정된 테스트 새글 제목', content='수정된 테스트 새글 내용', writer = 'user00', updateDate
= sysdate where bno = 1
 {executed in 0 msec}
```

BoardController의 작성

- 삭제 처리와 테스트
  - BoardController 클래스 추가 작성

```
@PostMapping("/remove")
public String remove(@RequestParam("bno") Long bno, RedirectAttributes rttr){

    log.info("remove..." + bno);
    if (service.remove(bno)) {
        rttr.addFlashAttribute("result", "success");
    }
    return "redirect:/board/list";
}
```

  - BoardControllerTests 클래스 추가 작성

```
@Test
public void testRemove() throws Exception {
    // 삭제전 데이터베이스에 게시물 번호 확인할 것
    String resultPage = mockMvc.perform(MockMvcRequestBuilders.post("/board/remove").param("bno", "25")).andReturn()
            .getModelAndView().getViewName();

    log.info(resultPage);
}
```

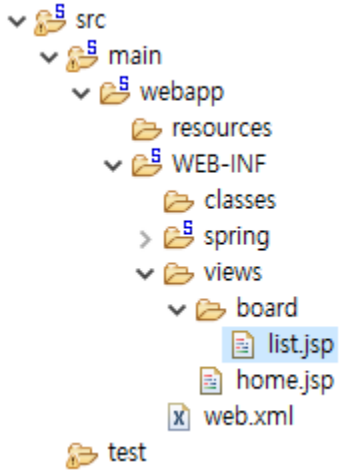
  - 테스트 결과 확인

```
INFO : jdbc.sqltiming - delete tbl_board where bno = 23
 {executed in 6 msec}
INFO : jdbc.audit - 1. PreparedStatement.execute() returned false
INFO : jdbc.audit - 1. PreparedStatement.getUpdateCount() returned 1
INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
```

화면 처리

- Tutorial에서는 Bootstrap을 사용해서 화면처리 진행
- 디자인은 'SB Admin2'
  - https://startbootstrap.com/themes/sb-admin-2/

- 게시물 리스트의 URL은 '/board/list'이므로 '/WEB-INF/views/board/list.jsp'가 되므로 해당 경로에 list.jsp 추가



목록 페이지 작업과 includes

- list.jsp 작성

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>List Page</h1>
</body>
</html>
```

- http://localhost:8080/controller/board/list 에서 결과 확인

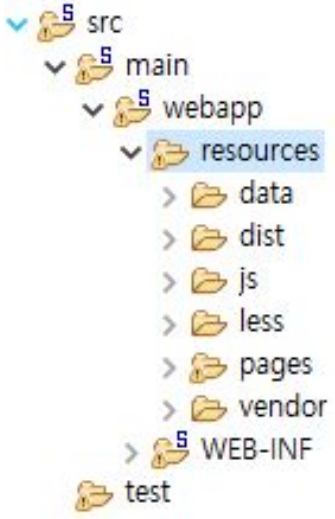
List Page

목록 페이지 작업과 includes

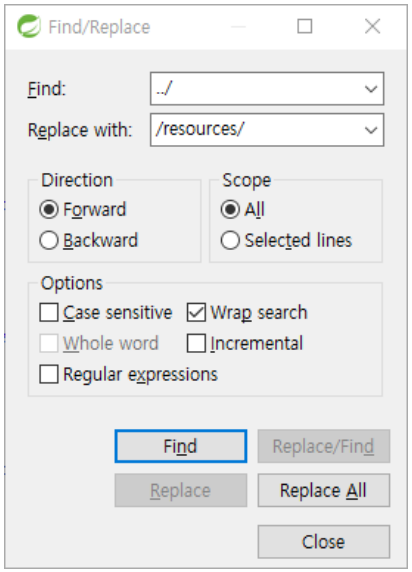
- SB Admin2 페이지 적용하기
  - SB Admin2의 pages 폴더에 있는 tables.html의 내용을 list.jsp의 내용으로 그대로 복사해서 수정하고 실행
  - 수정할 때 list.jsp의 상단에 JSP의 Page 지시자는 유지해야 함
  - CSS 등이 완전히 깨진 상태이므로 텍스트만 출력됨

목록 페이지 작업과 includes

- SB Admin2 페이지 적용하기
  - SB Admin2의 압축을 풀어둔 모든 폴더를 webapp 밑의 resources 폴더로 복사해 넣기

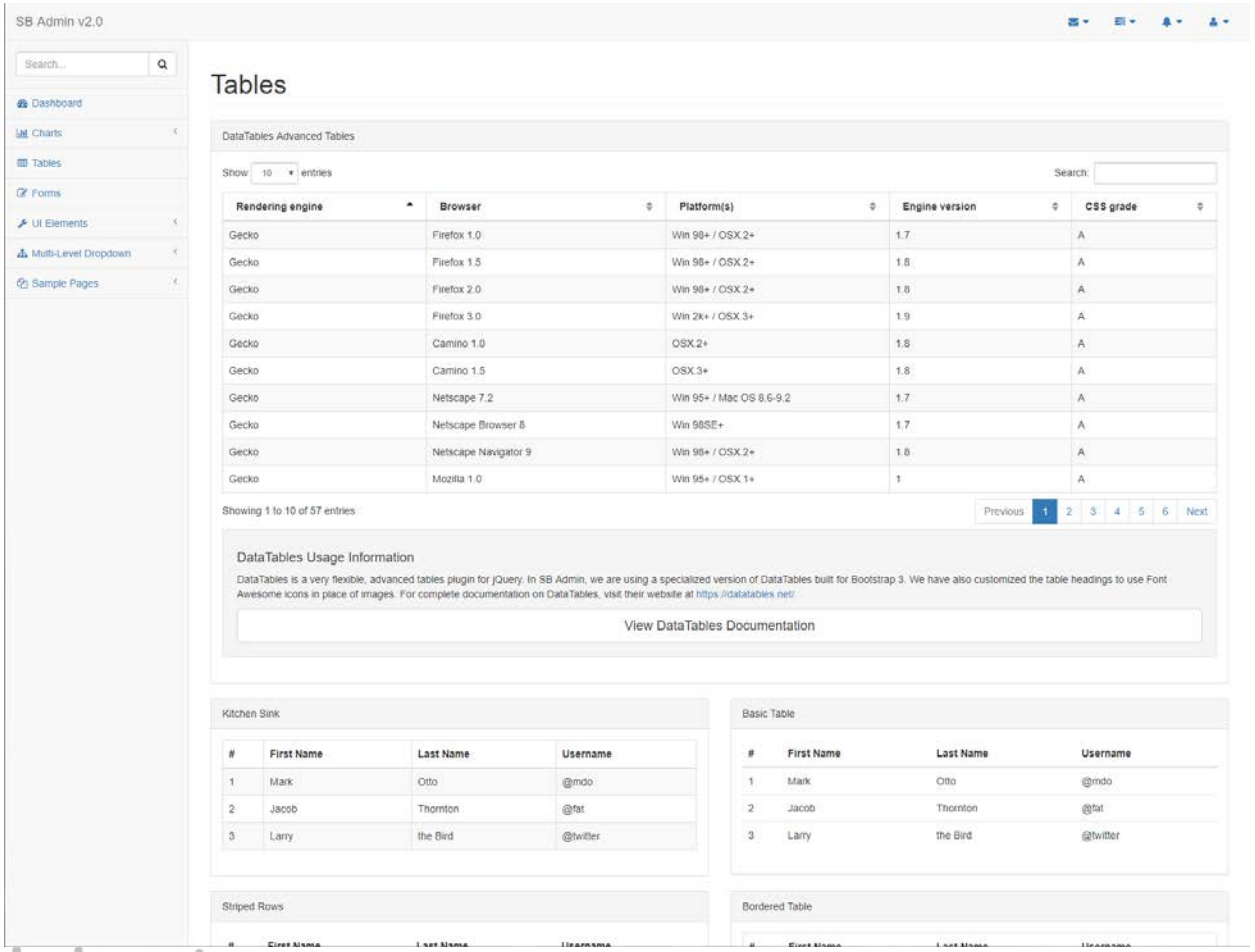


  - list.jsp 파일에서 CSS나 JS 파일의 경로를 '/resources/'로 시작하도록 수정



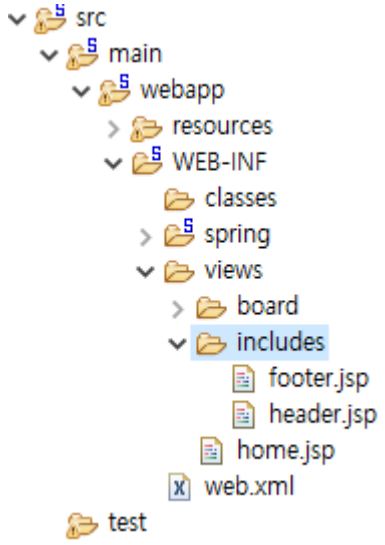
목록 페이지 작업과 includes

- SB Admin2 페이지 적용하기
  - 결과 화면



목록 페이지 작업과 includes

- includes 적용
  - JSP를 작성할 때마다 많은 양의 HTML 코드를 이용하는 것을 피하기 위해 사전에 작업을 해야 함.
  - 현재 프로젝트 views 폴더에 includes 폴더를 작성하고, header.jsp와 footer.jsp를 선언



목록 페이지 작업과 includes

- includes 적용
  - header.jsp
    - header.jsp는 페이지에서 핵심적인 부분이 아닌 영역 중에서 위쪽의 HTML 내용을 처리하기 위해 작성
    - list.jsp파일의 처음 부분에서 라인까지 잘라서 header.jsp의 내용으로 저장
  - footer.jsp
    - 가 끝나는 태그부터 마지막까지 footer.jsp의 내용으로 작성
  - 이후 브라우저를 통해 정상적으로 동작하는지 확인

목록 페이지 작업과 includes

- jQuery 라이브러리 변경
  - JavaScript로 브라우저 내에서 JSP 페이지를 조작하기 위해 수정 필요
    - footer.jsp의 상단에 있는 jquery.min.js 파일의 <script> 태그를 제거

```
    <!-- jQuery -->
<!-- <script src="/resources/vendor/jquery/jquery.min.js"></script> -->
```

    - jQuery는 인터넷을 통해서 다운로드 받을 수 있게 jQuery의 링크를 검색해서 header.jsp 내에 추가해야 함

```
<div id="page-wrapper">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

목록 페이지 작업과 includes

- jQuery 라이브러리 변경
  - 반응형 웹 처리
    - SB Admin2는 반응형으로 설계되어 있어서 브라우저의 크기에 맞게 모바일 용으로 자동으로 변경되지만 jQuery의 최신 버전을 사용한 상태에서는 모바일 크기에서 '새로고침' 시 메뉴가 펼쳐지는 문제가 발생
    - 이 문제를 해결하기 위해 includes 폴더 내 footer.jsp에 아래와 같은 코드를 기존 코드 대신에 추가

```
<!-- Page-Level Demo Scripts - Tables - Use for reference -->
<script>
  $(document).ready(function() {
    $('#dataTables-example').DataTable({
      responsive: true
    });
    $(".sidebar-nav")
      .attr("class","sidebar-nav navbar-collapse collapse")
      .attr("aria-expanded",'false')
      .attr("style","height:1px");
  });
</script>
```

목록 화면 처리

- HTML 내용이 너무 많으므로 아래와 같이 최소한의 태그들만 적용
  - list.jsp

list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>


<%@include file="../includes/header.jsp"%>
<div class="row">
    <div class="col-lg-12">
        <h1 class="page-header">Tables</h1>
    </div>
    <!-- /.col-lg-12 -->
</div>
<!-- /.row -->

<div class="row">
    <div class="col-lg-12">
        <div class="panel panel-default">
            <div class="panel-heading"> Board List Page </div>
            <!-- /.panel-heading -->
            <div class="panel-body">
                <table class="table table-striped table-bordered table-hover">
                    <thead>
                        <tr>
                            <th>#번호</th>
                            <th>제목</th>
                            <th>작성자</th>
                            <th>작성일</th>
                            <th>수정일</th>
                        </tr>
                    </thead>
                </table>
            </div>
            <!--  end panel-body -->
            </div>
    <!-- end panel -->
    </div>
</div>
<!-- /.row -->

    <%@include file="../includes/footer.jsp"%>
```

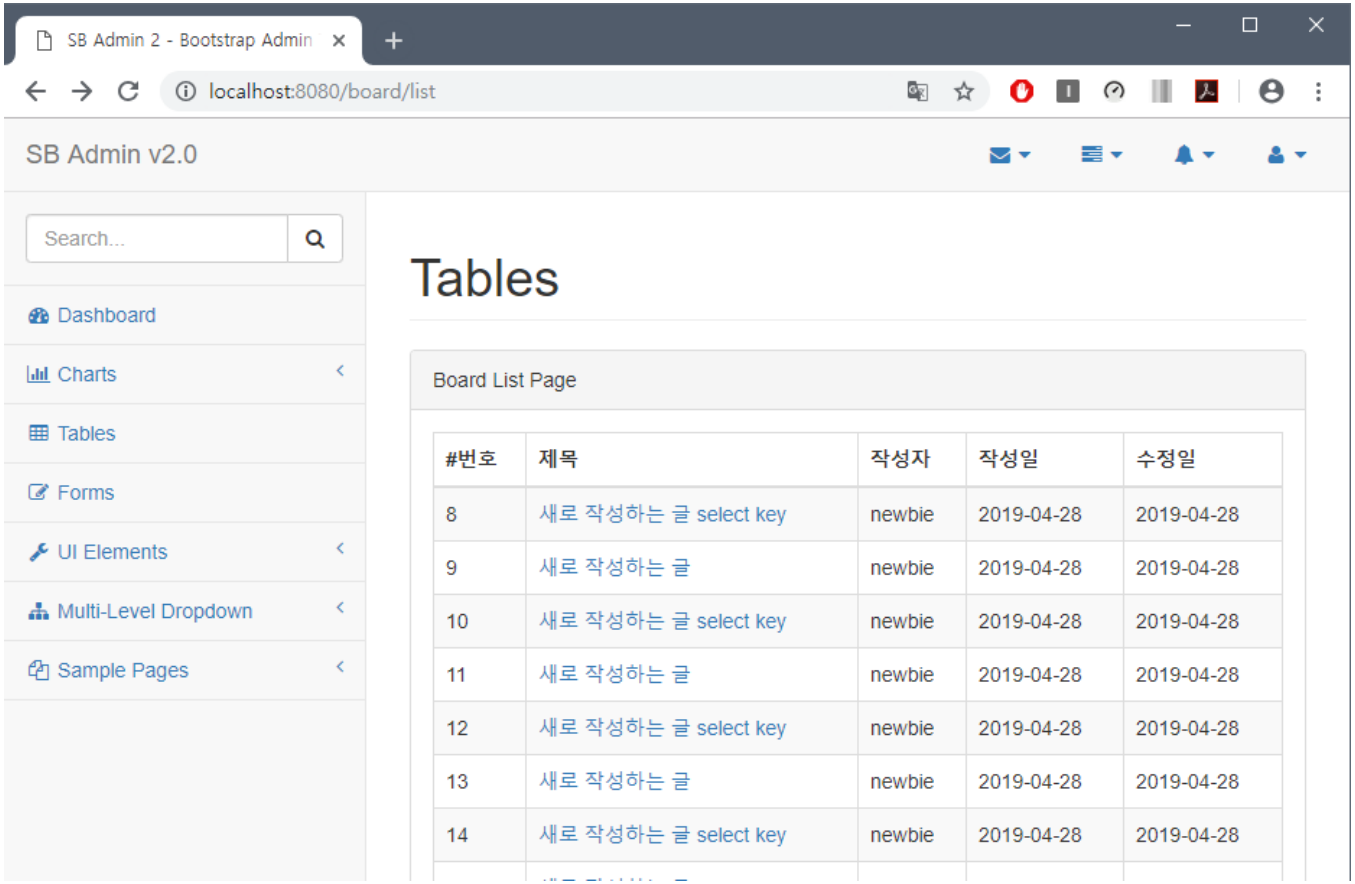
목록 화면 처리

- Model에 담긴 데이터 출력
  - '/board/list'를 실행했을 때 이미 BoardController는 Model을 이용해서 게시물의 목록을 'list'라는 이름으로 전달했으므로 list.jsp에서는 이를 출력함.
  - 출력은 JSTL을 이용해서 처리
  - list.jsp 내에 <tbody> 태그와 각 <tr>을 아래와 같이 작성

```
<c:forEach items="${list}" var="board">
    <tr>
        <td><c:out value="${board.bno}" /></td>
        <%-- <td><a href='/board/get?bno=<c:out value="${board.bno}"/>'><c:out value="${board.title}"/></a></td> --%>

        <td><a class='move' href='<c:out value="${board.bno}"/>'>
                <c:out value="${board.title}" />
        </a></td>

        <td><c:out value="${board.writer}" /></td>
        <td><fmt:formatDate pattern="yyyy-MM-dd"
                value="${board.regdate}" /></td>
        <td><fmt:formatDate pattern="yyyy-MM-dd"
                value="${board.updateDate}" /></td>
    </tr>
</c:forEach>
```

목록 화면 처리

- 결과
  - 브라우저를 통해서 결과를 확인하면 데이터베이스에 있는 전체 목록이 출력됨



광운대학교 KwangWoon University

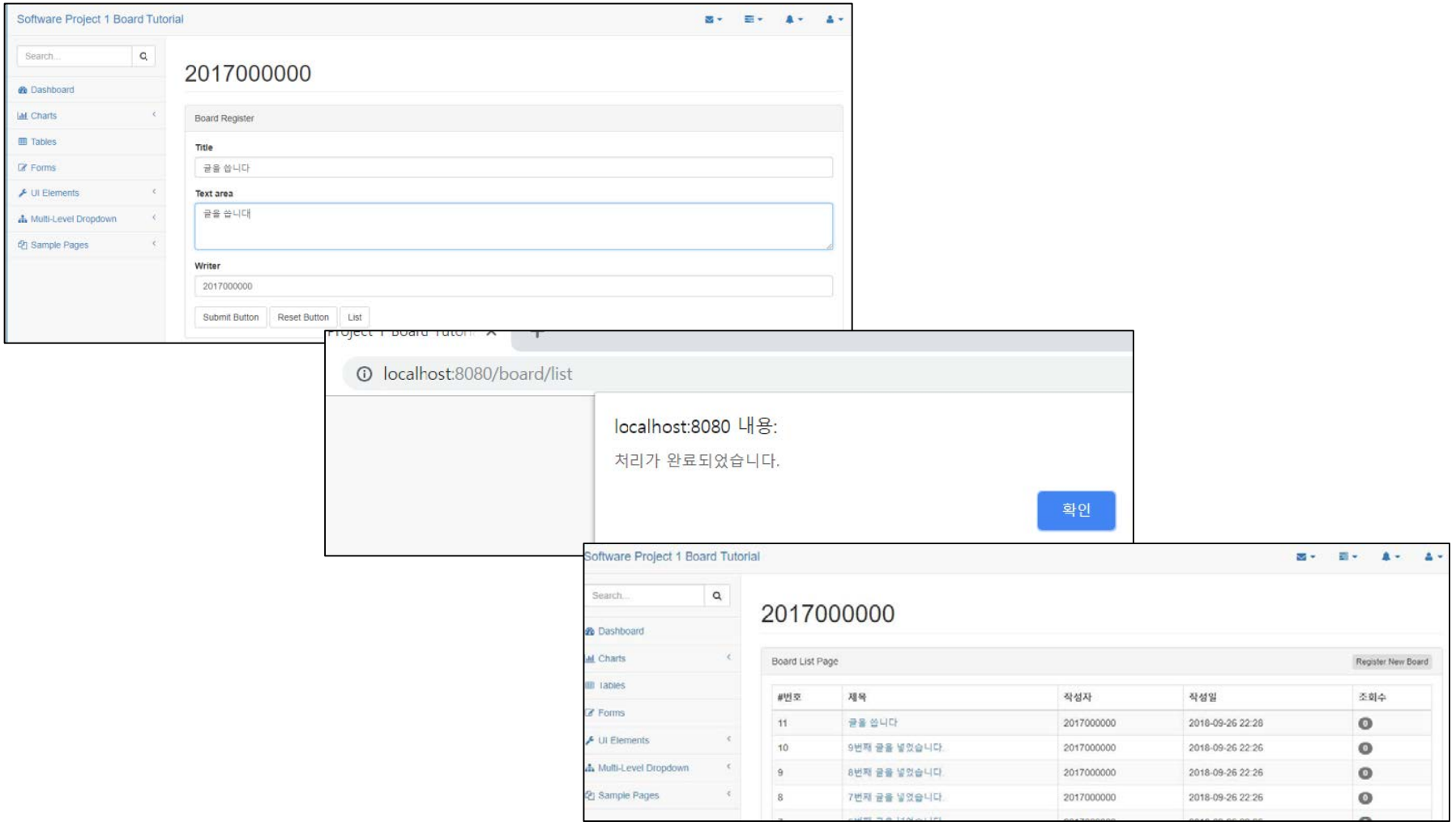
2019 Software Project

CRUD 예시

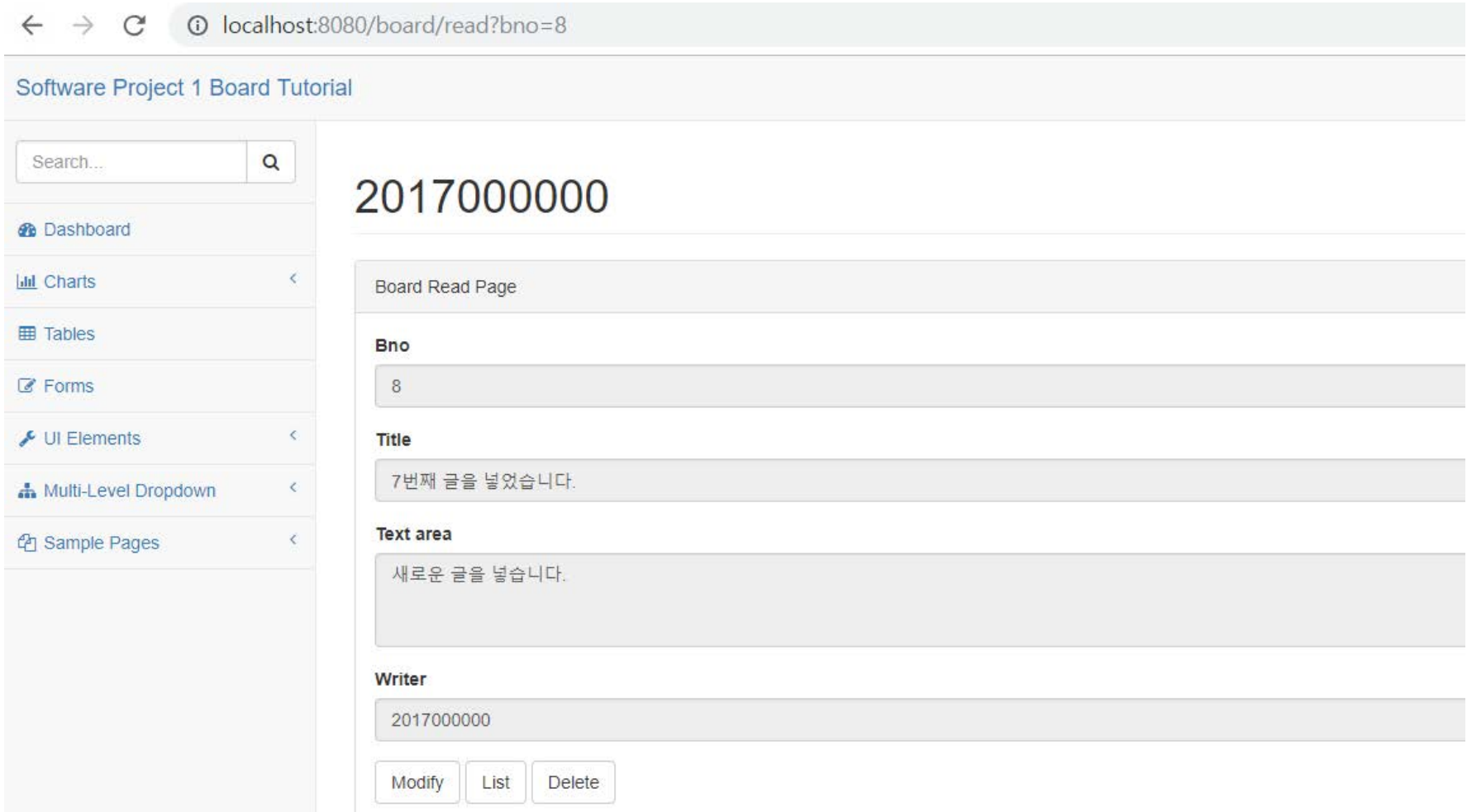
Kwangwoon Univ.
School of Computer and Information Engineering

조교 : 임영범 (ylim@kw.ac.kr)

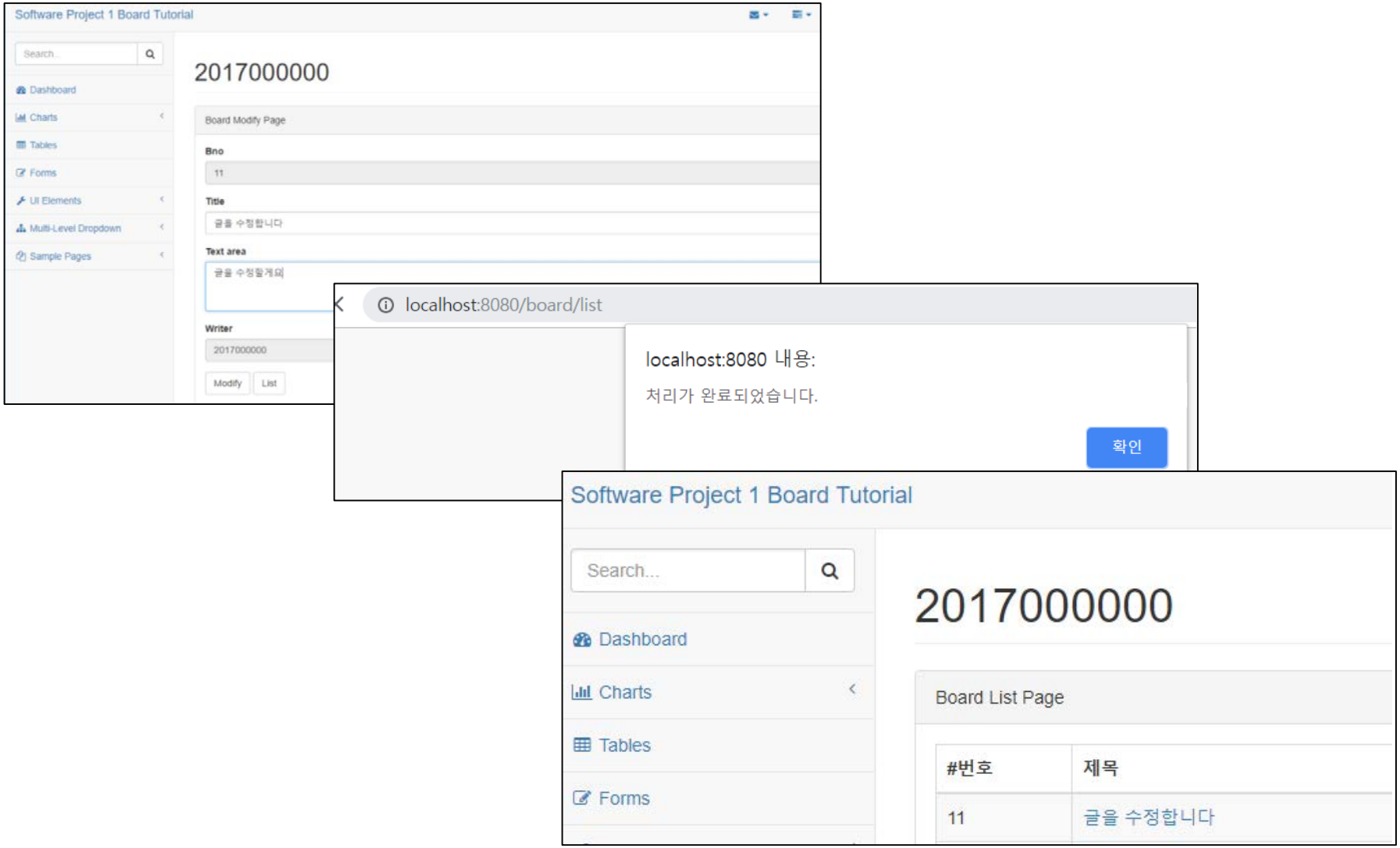
Create


Software Project 1 Board Tutorial
Search...
Dashboard
Charts
Tables
Forms
UI Elements
Multi-Level Dropdown
Sample Pages
2017000000
Board Register
Title
글을 씁니다
Text area
글을 씁니다
Writer
2017000000
Submit Button
Reset Button
List
localhost:8080/board/list
localhost:8080 내용:
처리가 완료되었습니다.
확인
Software Project 1 Board Tutorial
2017000000
Board List Page
Register New Board
#번호
제목
작성자
작성일
조회수
11
글을 씁니다
2017000000
2018-09-26 22:26
10
9번째 글을 넣었습니다.
2017000000
2018-09-26 22:26
9
8번째 글을 넣었습니다.
2017000000
2018-09-26 22:26
8
7번째 글을 넣었습니다.
2017000000
2018-09-26 22:26

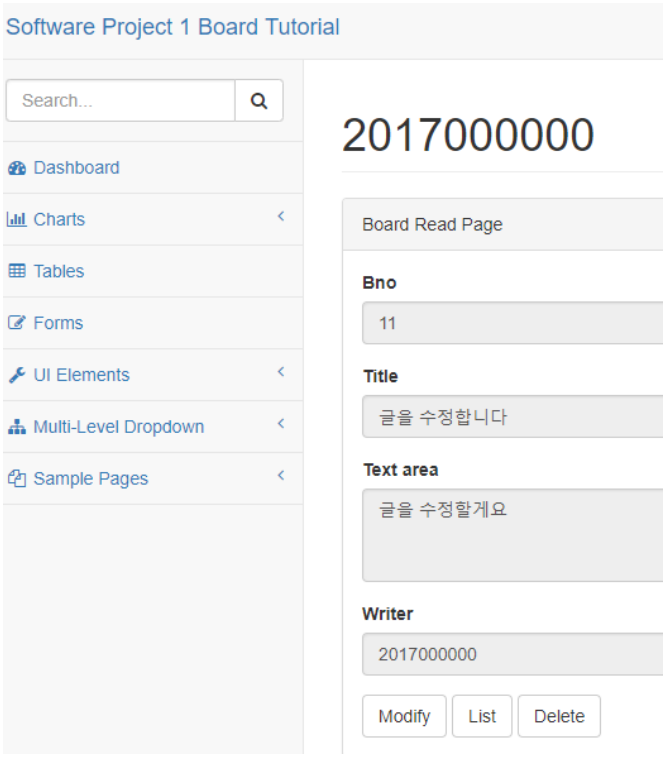
Read

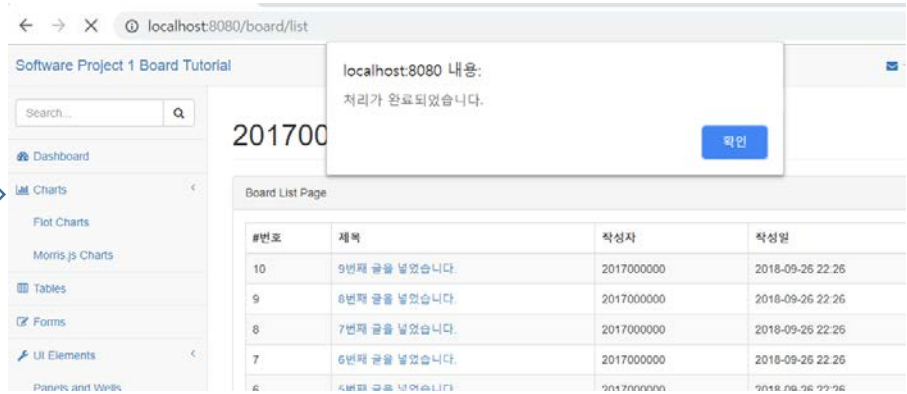

localhost:8080/board/read?bno=8
Software Project 1 Board Tutorial
Search...
Dashboard
Charts
Tables
Forms
UI Elements
Multi-Level Dropdown
Sample Pages
2017000000
Board Read Page
Bno
8
Title
7번째 글을 넣었습니다.
Text area
새로운 글을 넣습니다.
Writer
2017000000
Modify
List
Delete

Update


Software Project 1 Board Tutorial
Search...
Dashboard
Charts
Tables
Forms
UI Elements
Multi-Level Dropdown
Sample Pages
2017000000
Board Modify Page
Bno
11
Title
글을 수정합니다
Text area
글을 수정할게요
Writer
2017000000
Modify
List
localhost:8080/board/list
localhost:8080 내용:
처리가 완료되었습니다.
확인
Software Project 1 Board Tutorial
Search...
Dashboard
Charts
Tables
Forms
2017000000
Board List Page
#번호
제목
11
글을 수정합니다

Delete


Software Project 1 Board Tutorial
Search...
Dashboard
Charts
Tables
Forms
UI Elements
Multi-Level Dropdown
Sample Pages
2017000000
Board Read Page
Bno
11
Title
글을 수정합니다
Text area
글을 수정할게요
Writer
2017000000
Modify
List
Delete


localhost:8080/board/list
Software Project 1 Board Tutorial
localhost:8080 내용:
처리가 완료되었습니다.
확인
Board List Page
#번호
제목
작성자
작성일
10
9번째 글을 넣었습니다.
2017000000
2018-09-26 22:26
9
8번째 글을 넣었습니다.
2017000000
2018-09-26 22:26
8
7번째 글을 넣었습니다.
2017000000
2018-09-26 22:26
7
6번째 글을 넣었습니다.
2017000000
2018-09-26 22:26

References

- 코드로 배우는 스프링 웹 프로젝트