

# 자바 - 오라클DB 테이블 생성

SQL Developer 를 이용하여 본인의 계정으로 접속하여 테이블 생성.

게시물은 각 게시물 마다 고유의 번호가 필요 오라클의 경우 시퀀스를 이용하여 처리

시퀀스를 생성 할 때에는 다른 오브젝트들(테이블 등)과 구별하기 위하여 'seq'로 시작하거나, 't'와 같이 구분이 가능한 단어를 앞에 붙여준다.

tbl\_board 테이블 구성요소

bno : 고유의 번호

title : 제목

content : 내용

writer : 작성자

테이블을 설계할 때에는

가능하면 레코드의 생성 시간과 최종 수정 시간을 같이 기록하는 것이 일반적

regdate : 생성 시간

updatedate : 최종 수정 시간

레코드가 생성 된 시간을 자동으로 기록될 수 있도록 기본 값을 sysdate로 설정

PK(Primary Key)를 지정 할 때에는 'pk\_'로 시작하는 이름을 붙여주는 것이 일반적.

반드시 의미를 구분 할 수 있도록 생성해준다.

테이블 생성

```
create sequence seq_board;
```

```
create table tbl_board (  
  bno number(10,2),  
  title varchar2(200) not null,  
  content varchar2(2000) not null,  
  writer varchar2(50) not null,  
  regdate date default sysdate,  
  updatedate date default sysdate  
);
```

테이블 수정

```
alter table tbl_board  
add constraint pk_board
```

```
primary key(bno);
```

테이블 저장

```
commit;
```

테이블을 생성하고 나면, 테스트를 위한 여러개의 데이터를 추가하게 되는데, 이런 의미 없는 데이터를 'typ data' 혹은 'dummy data' 라고 한다.

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

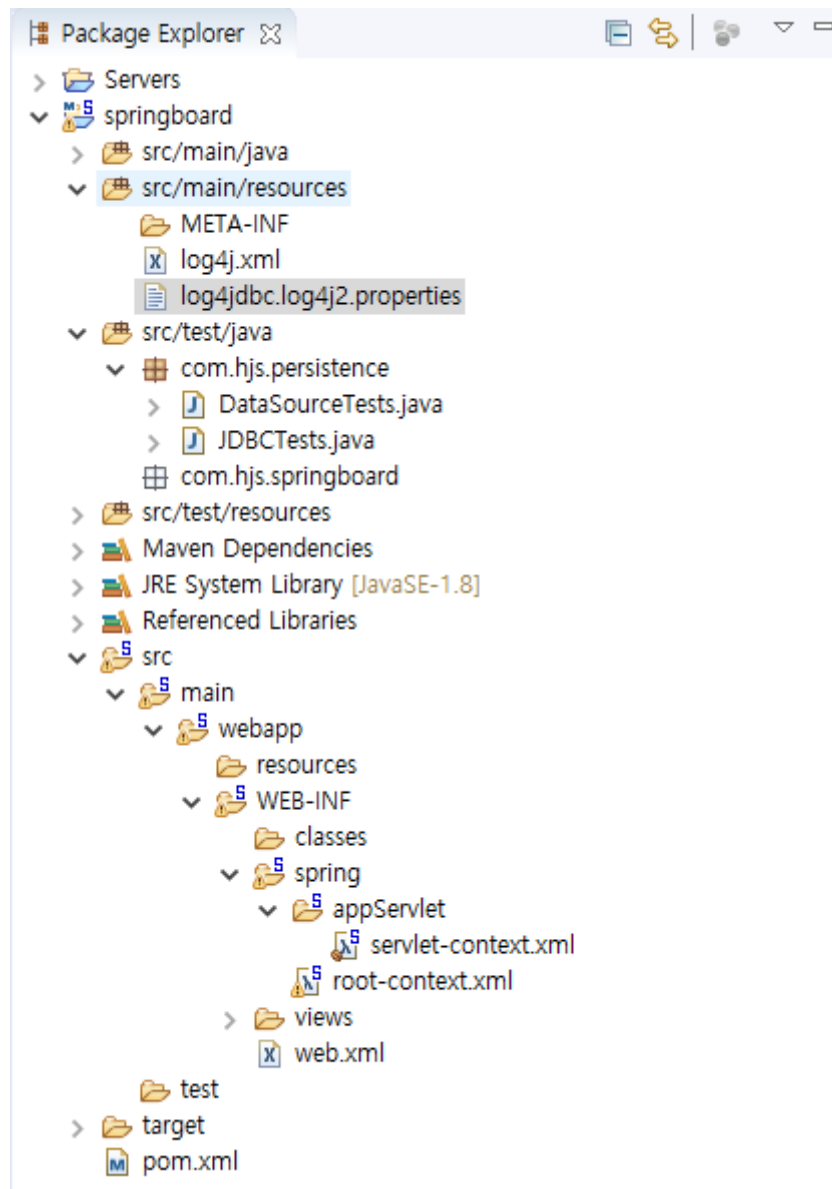
insert into tbl_board (bno, title, content, writer)
values (seq_board.nextval, '테스트제목', '테스트 내용1', 'user01');

insert into tbl_board (bno, title, content, writer)
values (seq_board.nextval, '테스트제목', '테스트 내용2', 'user02');

insert into tbl_board (bno, title, content, writer)
values (seq_board.nextval, '테스트제목', '테스트 내용3', 'user03');

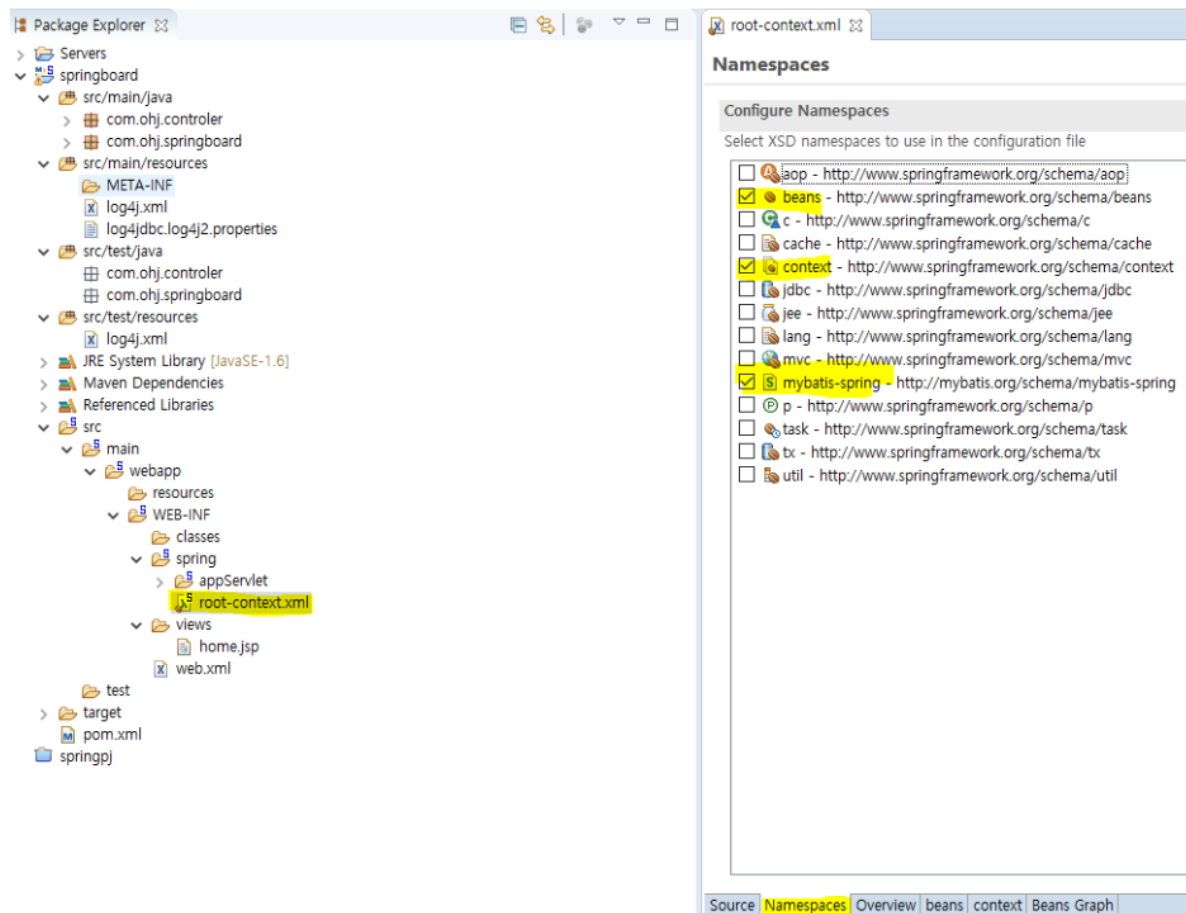
insert into tbl_board (bno, title, content, writer)
values (seq_board.nextval, '테스트제목', '테스트 내용4', 'user04');

commit;
```



#log4jdbc.log4j2.properties 파일 내부 코드(test 로그 값 찍어주는 것/콘솔창에 찍어주는 것 처럼 나옴)

```
log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
spring.datasource.hikari.connection-test-query=SELECT 1 FROM DUAL
```



## 네임스페이스 체크

프로젝트가 정상적으로 실행되기 위한 조건 :

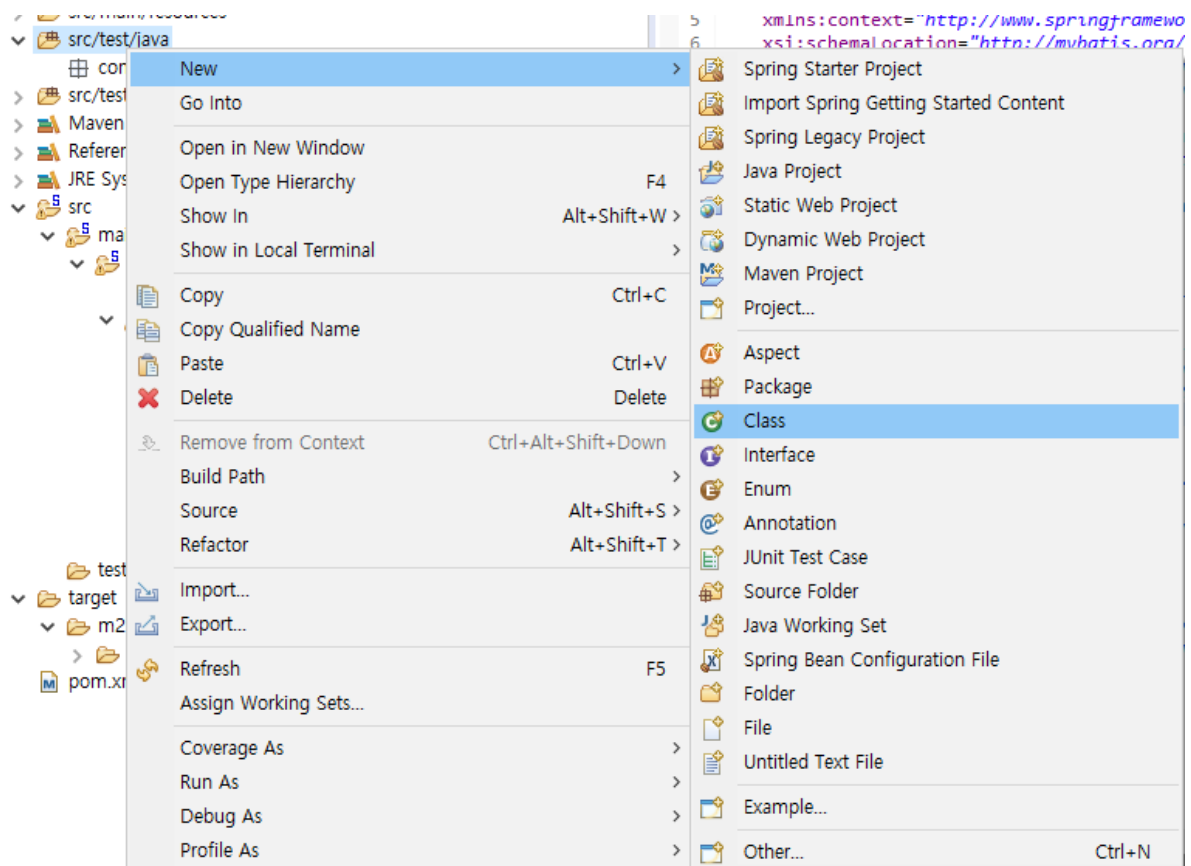
DataSource와 Mybatis 의 연결이 반드시 필요.


DataSourceTests.java

JDBCTests.java

위의 두 클래스는 반드시 웹 개발 이전에 테스트를 통해서 확인한다.


DataSourceTest




New Java Class

## Java Class

Create a new Java class.



Source folder:  Browse...

Package:  Browse...

☐ Enclosing type:  Browse...

Name:

Modifiers:
☒ public
☐ package
☐ private
☐ protected  
☐ abstract
☐ final
☐ static

Superclass:  Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?
☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

?

Finish

Cancel

Package Explorer

Servers
springboard
src/main/java
com.hjs.springboard2
HomeController.java
src/main/resources
META-INF
log4j.xml
log4jdbc.log4j2.properties
src/test/java
com.hjs.persistence
DataSourceTests.java
JDBCTests.java
com.hjs.springboard2
src/test/resources
Maven Dependencies
Referenced Libraries
JRE System Library [JavaSE-1.8]
src
main
webapp
resources
WEB-INF
classes
spring
appServlet
servlet-context.xml
root-context.xml
views
web.xml

DataSourceTests.java
JDBCTests.java

```

1 package com.hjs.persistence;
2
3 import static org.junit.Assert.fail;
19
20 @RunWith(SpringJUnit4ClassRunner.class)
21 @ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
22 @Log4j
23 public class DataSourceTests {
24
25     @Setter(onMethod_ = { @Autowired })
26     private DataSource dataSource;
27
28     @Setter(onMethod_ = { @Autowired })
29     private SqlSessionFactory sqlSessionFactory;
30
31     @Test
32     public void testMyBatis() {
33         //세션팩토리에 접속하기 위한 코드
34         try(SqlSession session = sqlSessionFactory.openSession();
35             Connection con = session.getConnection()) {
36             Log.info(session);
37             Log.info(con);
38
39         } catch (Exception e) {
40             fail(e.getMessage());
41         } // try catch END
42     } // testMyBatis() END
43
44     public void testConnection() {
45         try(Connection con = dataSource.getConnection()){
46             Log.info(con);
47         } catch (Exception e) {
48             fail(e.getMessage());
49         } // try catch END
50     } // testConnection() END
51 } // CLASS END

```

```

package com.hjs.persistence;

import static org.junit.Assert.fail;

```

```

import java.sql.Connection;

import javax.sql.DataSource;

import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory; // SqlSession 객체를 얻어내야
하기 때문에 импорт
import org.junit.Test; //단계별 테스트를 위한 импорт
import org.junit.runner.RunWith; // 실행 과정을 봐야 하기 위한 импорт
import org.springframework.beans.factory.annotation.Autowired; // 어노테이션을 이
용한 주입방식(오토와이어드 방식)
import org.springframework.test.context.ContextConfiguration; // 환경설정(테스트용
도) 구조
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;

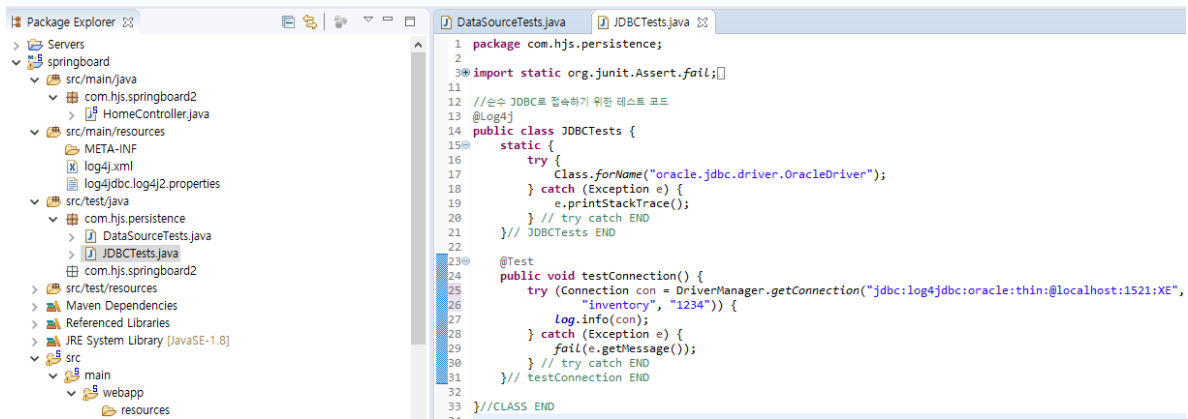
    @Setter(onMethod_ = { @Autowired })
    private SqlSessionFactory sqlSessionFactory;

    @Test
    public void testMyBatis() {
        //세션팩토리에 접속하기 위한 코드
        try(SqlSession session = sqlSessionFactory.openSession();
            Connection con = session.getConnection();) {
            log.info(session);
            log.info(con);

        } catch(Exception e) {
            fail(e.getMessage());
        } // try catch END
    } // testMyBatis() END

    public void testConnection() {
        try(Connection con = dataSource.getConnection()){
            log.info(con);
        } catch(Exception e) {
            fail(e.getMessage());
        } // try catch END
    } // testConnection() END
} // CLASS END

```



```
package com.hjs.persistence;

import static org.junit.Assert.fail;

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

import org.junit.Test;

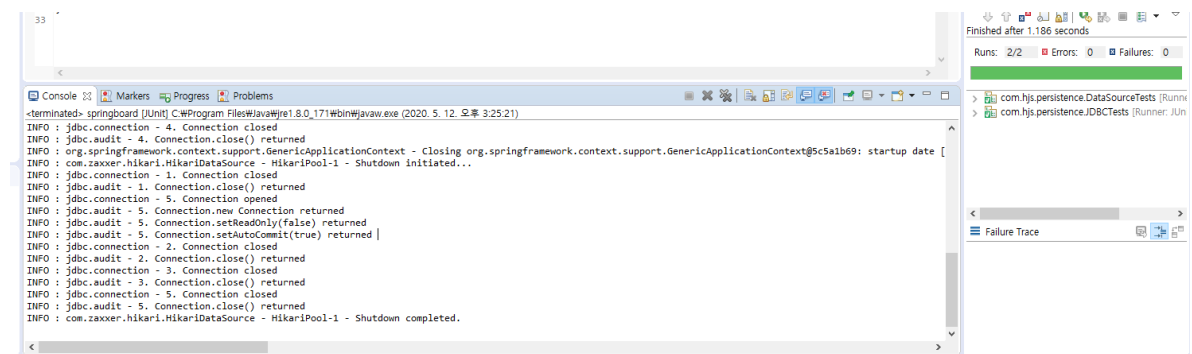
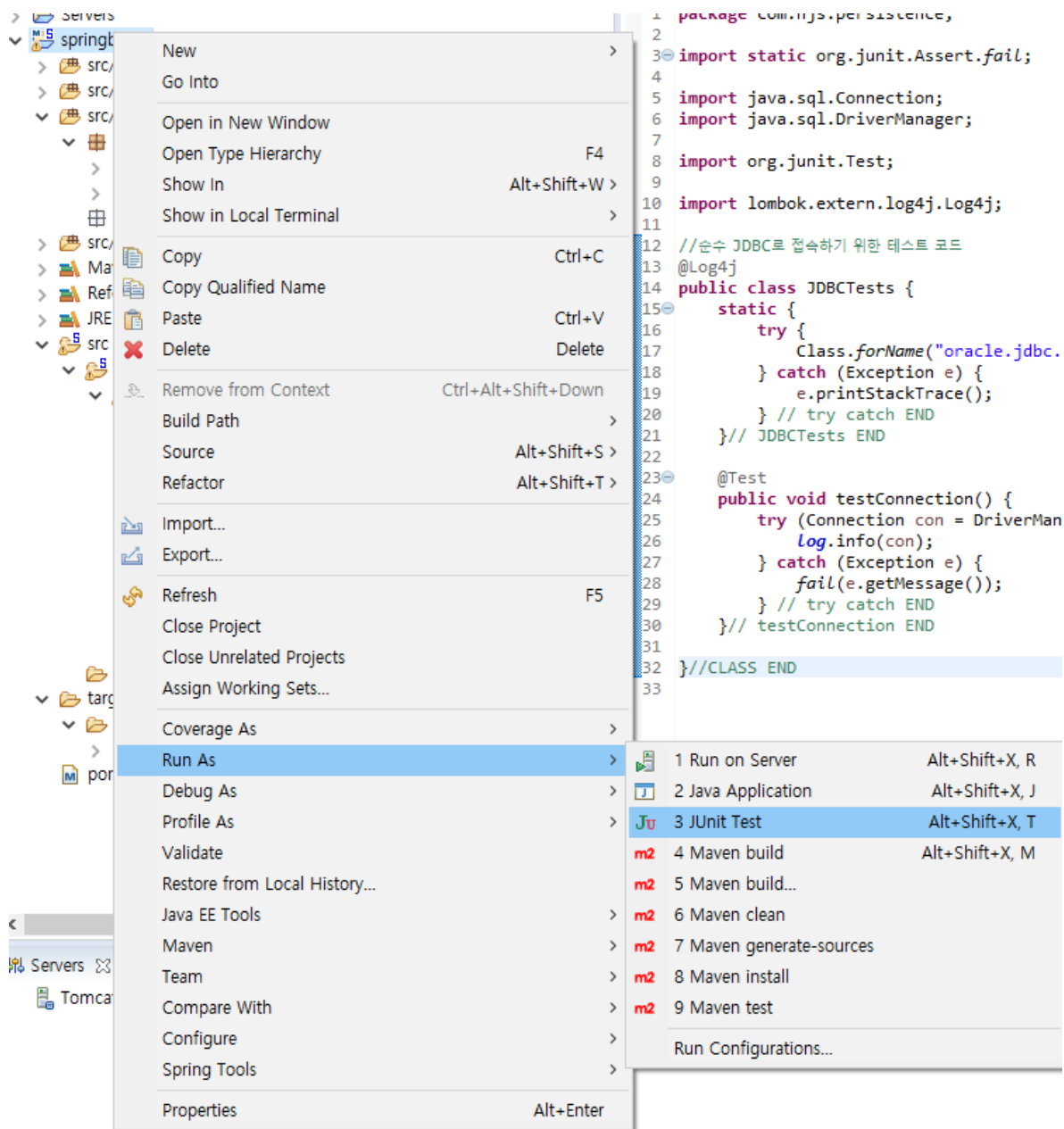
import lombok.extern.log4j.Log4j;

//순수 JDBC로 접속하기 위한 테스트 코드
@Log4j
public class JDBCTests {
    static {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (Exception e) {
            e.printStackTrace();
        } // try catch END
    } // JDBCTests END

    @Test
    public void testConnection() {
        try (Connection con =
DriverManager.getConnection("jdbc:log4jdbc:oracle:thin:@localhost:1521:XE",
"inventory", "1234")) {
            log.info(con);
        } catch (Exception e) {
            fail(e.getMessage());
        } // try catch END
    } // testConnection END

} //CLASS END
```





성공화면