



[한국ICT인재개발원] 스프링 프레임워크  
**2. DB 연동하기(OracleSQL)**

前) 광고데이터 분석 1년

前) IT강의 경력 2년 6개월

前) 머신러닝을 활용한 데이터 분석 프로젝트반 운영 1년

前) 리그오브 레전드 데이터 분석 등...

現) 국비반 강의 진행중

1

## ojdbc 설정과 테스트

Ojdbc 설정하기  
JUnit4를 활용한 테스트 코드 작성하기

2

## 커넥션풀 연동하기

Hikari cp 소개  
Oracle sql 로그 콘솔창에 출력하기

Home » com.oracle.database.jdbc » ojdbc8 » 18.15.0.0



## Ojdbc8 » 18.15.0.0

Oracle JDBC Driver compatible with JDK8, JDK11, J

```
<!-- https://mvnrepository.com/artifact/com.orac
<dependency>
  <groupId>com.oracle.database.jdbc</groupId>
  <artifactId>ojdbc8</artifactId>
  <version>18.15.0.0</version>
</dependency>
```

> ojdbc8-18.15.0.0.jar - C:\Users\MYCOM\m2\wr  
> servlet-api-2.5.jar - C:\Users\MYCOM\m2\repc  
> slf4j-api-1.6.6.jar - C:\Users\MYCOM\m2\repo  
> slf4j-log4j12-1.6.6.jar - C:\Users\MYCOM\m2\wr  
> spring-aop-5.0.7.RELEASE.jar - C:\Users\MYCOM  
> spring-beans-5.0.7.RELEASE.jar - C:\Users\MYCC  
> spring-context-5.0.7.RELEASE.jar - C:\Users\MYC

Mvnrepository에서 ojdbc8을 검색해 설치된 버전과 맞는 커넥터 호출

1. Pom.xml에 커넥터 jar파일 세팅하기.
2. Maven Dependencies에서 확인하기.

JSP에서는 DB와 자바가 연결되었는지 확인하기 위해서는

반드시 DAO를 먼저 생성하고

서버를 돌려서 DB와 접속하는 로직을 이용해서

콘솔창에 뜨는 메시지를 보고 연결여부를 확인했었습니다.

그러나 테스트 코드를 작성하면, 서버 작동 없이도 확인할 수 있습니다.

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

```
<!-- https://mvnrepository.com/artifact/org
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.0.7.RELEASE</version>
    <scope>test</scope>
</dependency>
```

1. Pom.xml에서 junit을 찾아 버전 4.7 -> 4.12로 변경하기
2. Mvnrepository에서 spring-test 검색해 스프링 버전과 동일하게 세팅

Home » [org.projectlombok](#) » [lombok](#)



## Project Lombok

Spice up your java: Automatic Resource Managem

License	MIT
Used By	13,157 artifacts

Central (46)	Lombok (7)	EBIPublic (2)	ICM (5)
1.18.20			

```
<!-- https://mvnrepository.com/artifact/
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.20</version>
    <scope>provided</scope>
</dependency>
```

1. Pom.xml에서 롬복 1.18.20버전(2021년 9월 9일 현재) 입력
2. 여기서 jar파일 импорт 이전에 사전 롬복 세팅도 해야함.

jstl-1.2.jar - C:\Users\MYCOM\m2\repository  
junit-4.12.jar - C:\Users\MYCOM\m2\repository  
log4j-1.2.15.jar - C:\Users\MYCOM\m2\repository  
lombok-1.18.20.jar - C:\Users\MYCOM\m2\repository  
ojdbc8-18.15.0.0.jar - C:\Users\MYCOM\m2\repository  
servlet-api-2.5.jar - C:\Users\MYCOM\m2\repository  
slf4j-api-1.6.6.jar - C:\Users\MYCOM\m2\repository  
slf4j-log4j12-1.6.6.jar - C:\Users\MYCOM\m2\repository  
spring-aop-5.0.7.RELEASE.jar - C:\Users\MYCOM\m2\repository  
spring-beans-5.0.7.RELEASE.jar - C:\Users\MYCOM\m2\repository  
spring-context-5.0.7.RELEASE.jar - C:\Users\MYCOM\m2\repository  
spring-core-5.0.7.RELEASE.jar - C:\Users\MYCOM\m2\repository  
spring-expression-5.0.7.RELEASE.jar - C:\Users\MYCOM\m2\repository  
spring-jcl-5.0.7.RELEASE.jar - C:\Users\MYCOM\m2\repository  
spring-test-5.0.7.RELEASE.jar - C:\Users\MYCOM\m2\repository

```
src/test/java
├── org.ict.controller
├── org.ict.dao
│   └── OjdbcConnectionTest.java
```

1. Junit과 spring-test가 제대로 들어왔는지 확인한 후
2. Test코드를 작성하는 src/test/java 하단에 테스트용 파일 생성



```
import lombok.extern.log4j.Log4j;

@Log4j
public class OjdbcConnectionTest {

    static {
        try {
```

```
@Test
public void testConnection()

    try(Connection con =
        DriverManager.get(
            "jdbc:orac
            "mytest",
            "mytest"
        )){
```

1. 롬복의 Log4j 기능을 이용한 로깅을 위해 클래스명 위에 @Log4j 입력
2. 테스트 코드는 @Test가 붙은 메서드를 메인메서드처럼 간주합니다.

```
@Log4j
public class OjdbcConnectionTest {

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

## 1. 드라이버 설정

드라이버는 어떤 DB와 접속할지 컴퓨터에 안내해주는 용도로 지정합니다.

```
@Test
public void testConnection() {

    try(Connection con =
        DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521/XEPDB1",
            "mytest",
            "mytest"
        )){
        Log.info(con);
    } catch (Exception e) {
        fail(e.getMessage());
    }
}
```

## 2. DB주소 설정

Oracle 11g버전 기준은 jdbc:oracle:thin:@localhost:1521:XE

Oracle 18c버전 기준은 jdbc:oracle:thin:@localhost:1521/XEPDB1

입니다.

```
@Test
public void testConnection() {

    try(Connection con =
        DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521/XEPDB1",
            "mytest",
            "mytest"
        )){
        Log.info(con);
    } catch (Exception e) {
        fail(e.getMessage());
    }
}
```

## 3. DB내 계정 설정

getConnection의 2, 3번째 파라미터로 계정id, pw를 순차입력합니다.

```
@Test
public void testConnection() {

    try(Connection con =
        DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521/XEPDB1",
            "mytest",
            "mytest")
    ){
        Log.info(con);
        Log.info("정상적으로 연결되었습니다.");
    } catch (Exception e) {
        fail(e.getMessage());
    }
}
```

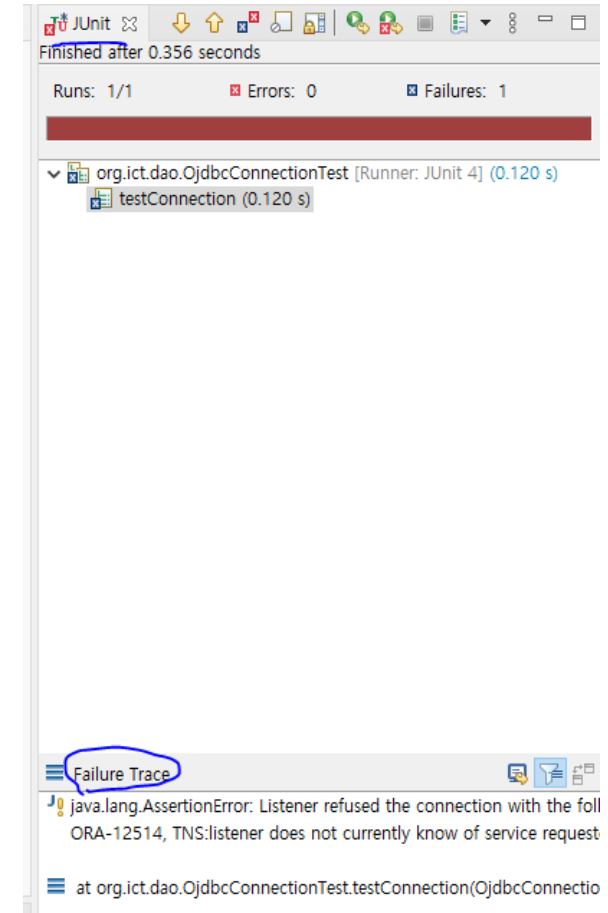
Progress Problems

org.ict.dao.OjdbcConnectionTest - oracle.jdbc.driver.T4CConnecti  
org.ict.dao.OjdbcConnectionTest - 정상적으로 연결되었습니다.

## 4. 테스트 코드 작성

- @Test가 붙은 메서드를 테스트 코드 실행시 실제로 실행합니다.
- 자바 1.7이후로 JDBC 코드는 반드시 try~catch구문을 사용해야합니다.
- Fail()은 따로 import를 해야하고, 연결 실패시 사유를 보여줍니다.

```
ers#MYCOM#.p2#pool#plugins#org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_11.0.2.v20200815-0835#jre#bin#javaw.exe (2021. 9. 9. 오전 12:47:50 - S  
l.OjdbcConnectionTest - oracle.jdbc.driver.T4CConnection@201  
l.OjdbcConnectionTest - 정상적으로 연결되었습니다.
```



6. 정상 연결시 위와같이 커넥터 주소가 나오며, 비정상 연결시 failure trace에 사유가 나옵니다.

커넥션 풀은 DAO가 하나인경우 접속자가 여럿 몰렸을 때  
하나의 DAO가 모든 요청을 처리할 수 없어  
지연이 발생하거나 서버가 터지는 것을 방지하기 위해  
DAO를 요청량에 따라 가변적으로 생성하여  
많은 요청도 여러 개의 DAO로 대응할 수 있도록 도와줍니다.



Hikari-cp는 spring-boot에서 현재 표준으로 채택된 커넥션 풀로 스프링 개발자들이 가장 많이 사용하는 커넥션 풀입니다. 설치를 위해 mvnrepository에서 hikaricp를 검색합니다.



## Artifacts

Java 11+ maven artifact:

```
<dependency>
  <groupId>com.zaxxer</groupId>
  <artifactId>HikariCP</artifactId>
  <version>5.0.0</version>
</dependency>
```

Java 8 maven artifact (maintenance mode):

```
<dependency>
  <groupId>com.zaxxer</groupId>
  <artifactId>HikariCP</artifactId>
  <version>4.0.3</version>
</dependency>
```

✓ Maven Dependencies	18
HikariCP-4.0.3.jar - C:\Users\MYCOM\m2	19
mysql-connector-java-8.0.21.jar - C:\Users\	20
protobuf-java-3.11.4.jar - C:\Users\MYCOM	21
junit-4.12.jar - C:\Users\MYCOM\m2\rep	22
hamcrest-core-1.3.jar - C:\Users\MYCOM\	23
spring-test-5.0.7.RELEASE.jar - C:\Users\MY	24
spring-core-5.0.7.RELEASE.jar - C:\Users\MY	25
spring-jcl-5.0.7.RELEASE.jar - C:\Users\MY	
spring-context-5.0.7.RELEASE.jar - C:\Users\	
spring-aop-5.0.7.RELEASE.jar - C:\Users\MY	
spring-beans-5.0.7.RELEASE.jar - C:\Users\MY	
spring-expression-5.0.7.RELEASE.jar - C:\Us	
spring-webmvc-5.0.7.RELEASE.jar - C:\Users	
spring-web-5.0.7.RELEASE.jar - C:\Users\MY	

```
<!-- https://mvnrepository.com/artifact
<dependency>
  <groupId>com.zaxxer</groupId>
  <artifactId>HikariCP</artifactId>
  <version>4.0.3</version>
</dependency>
```

<https://github.com/brettwooldridge/HikariCP>

제작자 공식 깃허브에 java11버전 이상은 5.0.0을,  
8버전은 4.0.3을 권장하기 때문에  
본인 이클립스에 처리된 버전에 맞게 설치하세요.  
단, 4.0.3버전을 11버전에서 설치해도 큰 문제는 없었습니다.

```
<!-- Root Context: defines shared resources visible to all other web c<
<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/XEPDB1"></property>
  <property name="username" value="mytest"></property>
  <property name="password" value="mytest"></property>
</bean>
```

```
<bean id="datasource"
  class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
  <constructor-arg ref="hikariConfig"></constructor-arg>
</bean>
```

root-context.xml 내부에 bean 설정을 합니다.  
hikariCP에 접속정보를 입력한 다음  
커넥션풀을 담당하는 DataSource 객체에 집어넣어줍니다.

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

    @Autowired
    private DataSource dataSource;

    @Test
    public void testConnection() {
        try(Connection con = dataSource.getConnection()){
            Log.info(con);
            Log.info("hikariCP connection");
        } catch (Exception e) {
            fail(e.getMessage());
        }
    }
}
```

테스트용 코드로 MySQLConnectionPoolTest를 생성합니다.

커넥션 풀 설정이 된 코드는 DataSource 객체가 connection을 호출합니다.

```
org.springframework.test.context.support.DefaultTestContextBootstrapper -  
org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML  
org.springframework.context.support.GenericApplicationContext - Refreshing  
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostPr  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.  
org.ict.dao.MySQLConnectionPoolTest - HikariProxyConnection@1727424614 wr  
org.ict.dao.MySQLConnectionPoolTest - hikariCP connected  
org.springframework.context.support.GenericApplicationContext - Closing o  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

연결이 정확하게 되었다면 위와 같이 HikariPool을 활용한 접속 로그가 나오며, 작업에 따라 HikariPool의 번호가 1번만 나오거나 1~n번까지 나오는 등 시스템 부하에 따른 조절을 합니다.