

7장. MyBatis와 스프링 연동



MyBatis



Oracle Database 연동

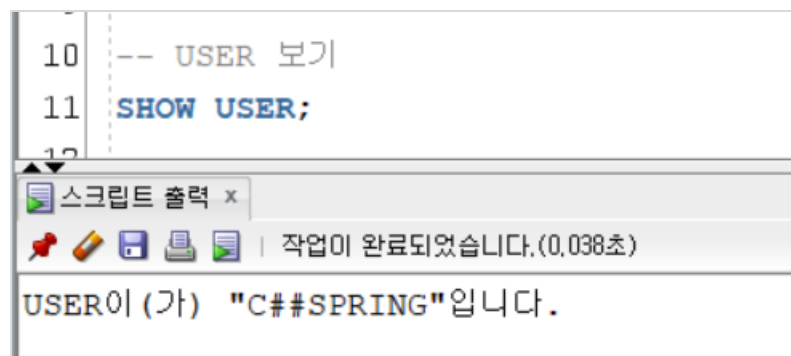
◆ 오라클 스키마 계정 생성

```
-- c##spring 데이터베이스 생성
-- DB이름, 비밀번호, 테이블 공간 생성

CREATE USER c##spring IDENTIFIED BY spring
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP;

-- 권한 설정
GRANT CONNECT, DBA TO c##spring;
```

```
10 -- USER 보기
11 SHOW USER;
12
```



스크립트 출력 x

작업이 완료되었습니다.(0.038초)

USER이 (가) "C##SPRING"입니다.



Oracle Database 연동

◆ 사용자 등록

사용자이름 : c##spring , 비밀번호: spring

새로 만들기/데이터베이스 접속 선택

접속 이름	접속 세부정보
JWEBDB	C##JWEB@//l...
SPRINGDB	c##spring@//l...
SYSTEM	system@//loc...

Name: SPRINGDB Color

데이터베이스 유형: Oracle

사용자 정보 프록시 사용자

인증 유형: 기본값

사용자 이름(U): c##spring 롤(L): 기본값

비밀번호(P): ☐ 비밀번호 저장(Y)

접속 유형(Y): 기본

세부정보 고급

호스트 이름(A): localhost

포트(B): 1521

☒ SID(I): xe

☐ 서비스 이름(E):

상태: 성공

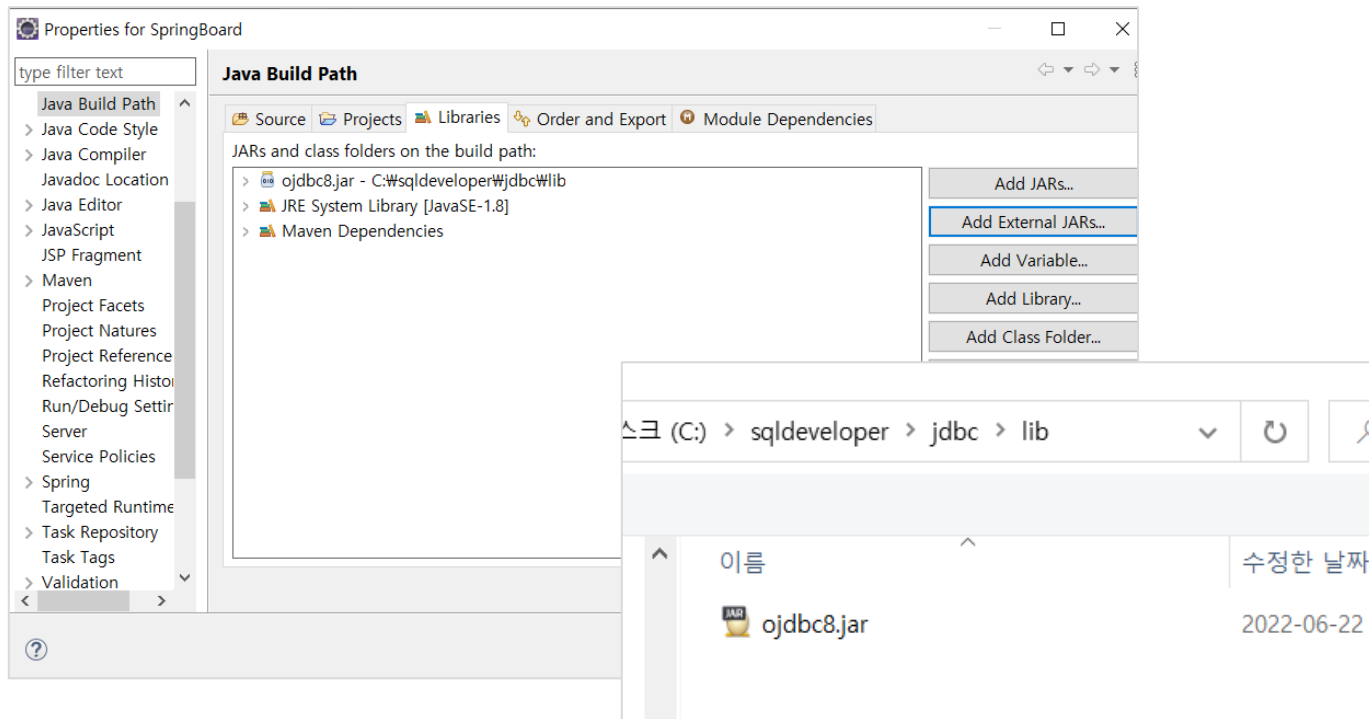
도움말(H) 저장(S) 지우기(C) 테스트(T) 접속(O) 취소



오라클 드라이버 연결

- 스프링 프로젝트 - 오라클 드라이버 추가하기

프로젝트 -> properties -> build path -> Add External Jars -> ojdbc8.jar 선택



Oracle Database 연동

◆ pom.xml에 라이브러리 추가하기

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

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.18.16</version>
  <scope>provided</scope>
</dependency>
```

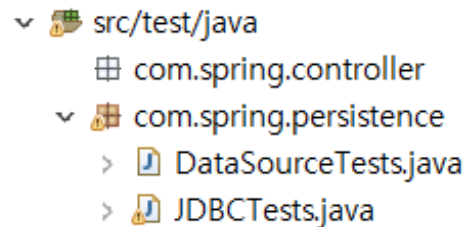


Oracle Database 연동

◆ pom.xml에 라이브러리 추가하기

runtime 부분 삭제하고 1.2.15 -> 1.2.17로 변경

```
<dependency>  
  <groupId>log4j</groupId>  
  <artifactId>log4j</artifactId>  
  <version>1.2.17</version>  
</dependency>
```



src/test/java

- com.spring.controller
- com.spring.persistence
 - DataSourceTests.java
 - JDBCTests.java



Oracle Database 연동

◆ JDBC 테스트 – Junit

```
@Log4j
public class JDBCTests {

    @Test
    public void testConnection() throws Exception {

        Class clz = Class.forName("oracle.jdbc.OracleDriver");

        System.out.println(clz); //클래스 이름 출력

        Connection con =
            DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe",
                "c##spring", "spring");

        Log.info(con); //연결 객체 출력

        con.close();
    }
}
```

Coverage As	>	1 Run on Server	Alt+Shi
Run As	>	2 JUnit Test	Alt+Shi
Debug As	>		

Runs: 1/1 Errors: 0 Failures: 0

> com.spring.persistence.JDBCTests [Runner: J] Failure Trace

class oracle.jdbc.OracleDriver

INFO : com.spring.persistence.JDBCTests - oracle.jdbc.driver.T4CConnection@5e7cd6cc



데이터베이스 커넥션 풀(DBCP)

◆ 커넥션 풀(DBCP) 설정 – Junit

```
<dependency>  
  <groupId>com.zaxxer</groupId>  
  <artifactId>HikariCP</artifactId>  
  <version>3.4.5</version>  
</dependency>
```

▼ Maven Dependencies

- > aspectjrt-1.6.10.jar - C:\Users\kiyon\m2\repository\org\aspectj\aspectjrt\1.6.10
- > aspectjweaver-1.9.2.jar - C:\Users\kiyon\m2\repository\org\aspectj\aspectjweaver\1.9.2
- > hamcrest-core-1.3.jar - C:\Users\kiyon\m2\repository\org\hamcrest\hamcrest-core\1.3
- > HikariCP-3.4.5.jar - C:\Users\kiyon\m2\repository\com\zaxxer\HikariCP\3.4.5



데이터베이스 커넥션 풀(DBCP)

◆ 커넥션 풀(DBCP) 설정 – DataSource 빈 등록

```
<!-- DataSource 설정 - hikari 빈 등록 -->
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
    <property name="driverClassName" value="oracle.jdbc.OracleDriver" />
    <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe" />
    <property name="username" value="c##spring" />
    <property name="password" value="spring" />
</bean>
<!-- 생성자 주입(DI) -->
<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource"
    destroy-method="close">
    <constructor-arg ref="hikariConfig" />
</bean>
```



데이터베이스 커넥션 풀(DBCP)

◆ 커넥션 풀(DBCP) 설정 - Junit

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

    @Autowired
    private DataSource ds;

    @Test
    public void testConnection() {
        try(Connection con = ds.getConnection()){

            Log.info(con); //연결 객체 생성
        }catch(Exception e) {
            e.printStackTrace();
            Log.error(e.getMessage());
        }
    }
}
```

```
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
INFO : com.spring.persistence.DataSourceTests - HikariProxyConnection@764358458 wrapping oracle.jdbc.driver.T4
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```



MyBatis와 스프링 연동

◆ Mybatis

MyBatis는 SQL 매핑 프레임워크로 분류되는데 JDBC의 복잡하고 반복되는 코드를 줄일 수 있어서 JPA(Java Persistence API)와 함께 많이 사용된다.

JDBC 프로그램	MyBatis
직접 connection을 맺고 마지막에 close() PreparedStatement 직접 생성 및 처리 Select의 경우 직접 ResultSet 처리	자동으로 connection close() MyBatis 내부적으로 PreparedStatement 처리 리턴 타입을 지정하는 경우 자동으로 객체 생성 및 ResultSet 처리



MyBatis와 스프링 연동

◆ Mybatis 관련 라이브러리 추가

```
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.6</version>
</dependency>
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.6</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>
```



MyBatis와 스프링 연동

◆ SQLSessionFactory

MaBatis의 가장 핵심적인 객체는 SQLSession과 SQLSessionFactory이다.

SQLSessionFactory는 Connection을 생성하거나 원하는 SQL을 전달하고 결과를 리턴받는 구조로 작동한다.

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

<!-- sqlSessionFactory 빈 등록 -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
</bean>
```



MyBatis와 스프링 연동

◆ DataSource Test

```
@Autowired
private SqlSessionFactory sqlSessionFactory;

@Test
public void testMyBatis() {
    try(SqlSession session = sqlSessionFactory.openSession();
        Connection con = session.getConnection()){
        Log.info(session); //session 객체
        Log.info(con);     //connect 객체
    }catch(Exception e) {
        e.getMessage();
    }
}
```

```
INFO : com.spring.persistence.DataSourceTests - org.apache.ibatis.session.defaults.DefaultSqlSession@602c
INFO : com.spring.persistence.DataSourceTests - HikariProxyConnection@1610372241 wrapping oracle.jdbc.dri
```



MyBatis와 스프링 연동

◆ Mapper 인터페이스

```
package com.spring.mapper;

import org.apache.ibatis.annotations.Select;

public interface TimeMapper {

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

}
```

root-context.xml – 객체 스캔

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



MyBatis와 스프링 연동

◆ Mapper 테스트 – 1. TimeMapper.java 사용

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

    @Autowired
    private TimeMapper mapper;

    @Test
    public void testGetTime() { //TimeMapper.java를 사용
        Log.info(mapper.getTime());
    }
}
```

```
INFO : com.spring.persistence.TimeMapperTests - 2022-08-28 11:24:55
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
```



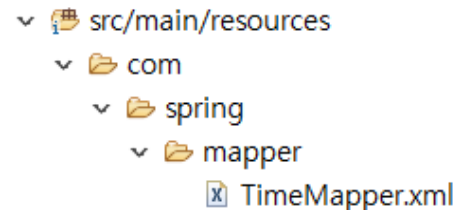
MyBatis와 스프링 연동

◆ Mapper 테스트 – 2. TimeMapper.xml 사용

src/main/resource에 com > spring > mapper 폴더를 생성함

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

<mapper namespace="com.spring.mapper.TimeMapper">
  <select id="getTime2" resultType="String">
    SELECT sysdate FROM dual
  </select>
  <!-- id는 getTime2()와 같음. 반환자료형(resultType) -->
</mapper>
```



```
src/main/resources
├── com
│   └── spring
│       └── mapper
│           └── TimeMapper.xml
```



MyBatis와 스프링 연동

◆ Mapper 테스트 – 2. TimeMapper.xml 사용

```
public class TimeMapperTests {  
  
    @Autowired  
    private TimeMapper mapper;  
  
    @Test  
    public void testGetTime() { //TimeMapper.java를 사용  
        Log.info(mapper.getTime());  
    }  
  
    @Test  
    public void testGetTime2() { //TimeMapper.xml을 사용  
        Log.info(mapper.getTime2());  
    }  
}
```

```
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.  
INFO : com.spring.persistence.TimeMapperTests - 2022-08-28 11:33:04  
INFO : com.spring.persistence.TimeMapperTests - 2022-08-28 11:33:04  
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
```

