KB금융그룹 | 국민의 평생 금융파트너

JDBC 알죠

그거하고

연결해볼거에요

2025년 상반기 K-디지털 트레이닝

스프링과 MySQL Database 연동

[KB] IT's Your Life

KB 국민은행

데이터베이스 생성 및 계정 생성, 권한 설정

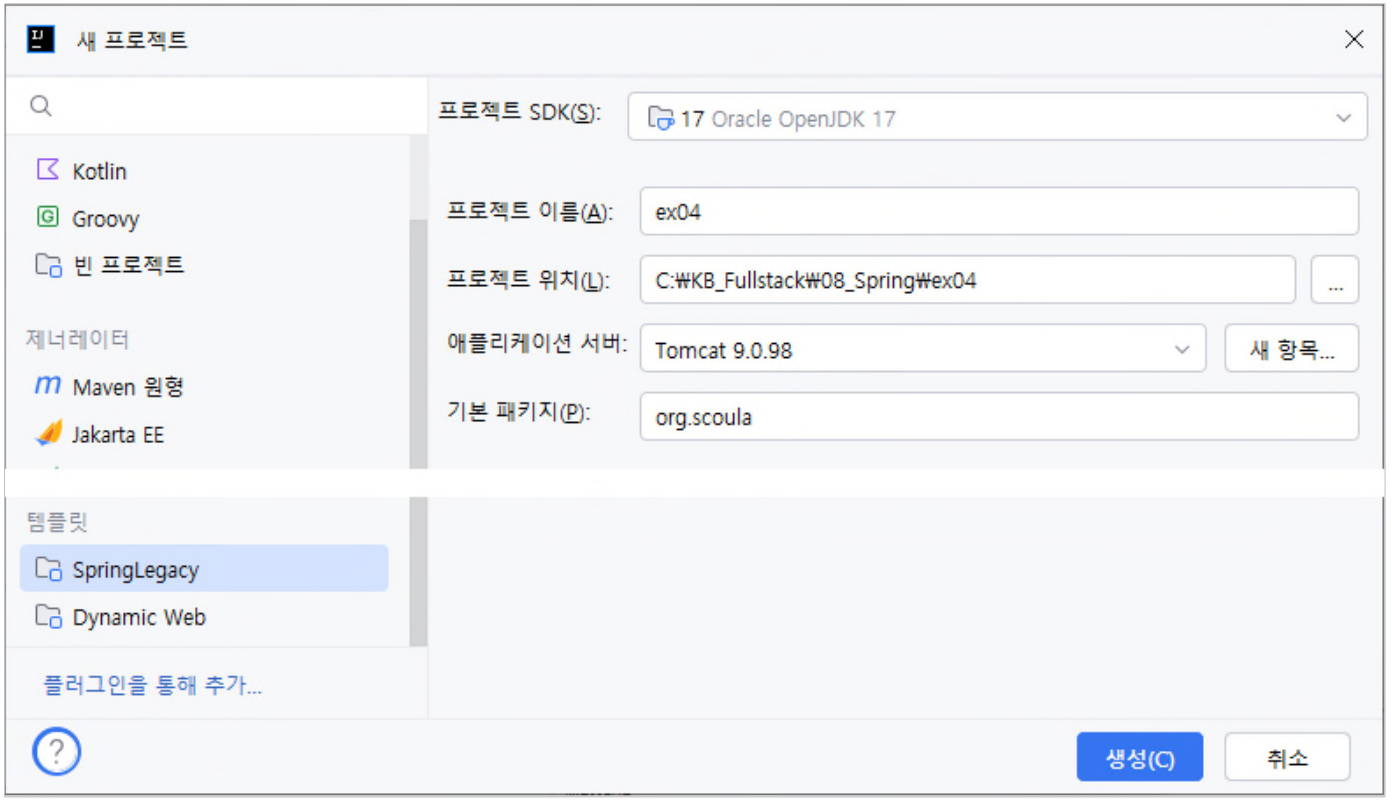
```
create database scoula_db;

create user 'scoula'@'%' identified by '1234';

grant all privileges on scoula_db.* to 'scoula'@'%';
```

프로젝트 만들기

- Templates: SpringLegacy
- Project name: ex04

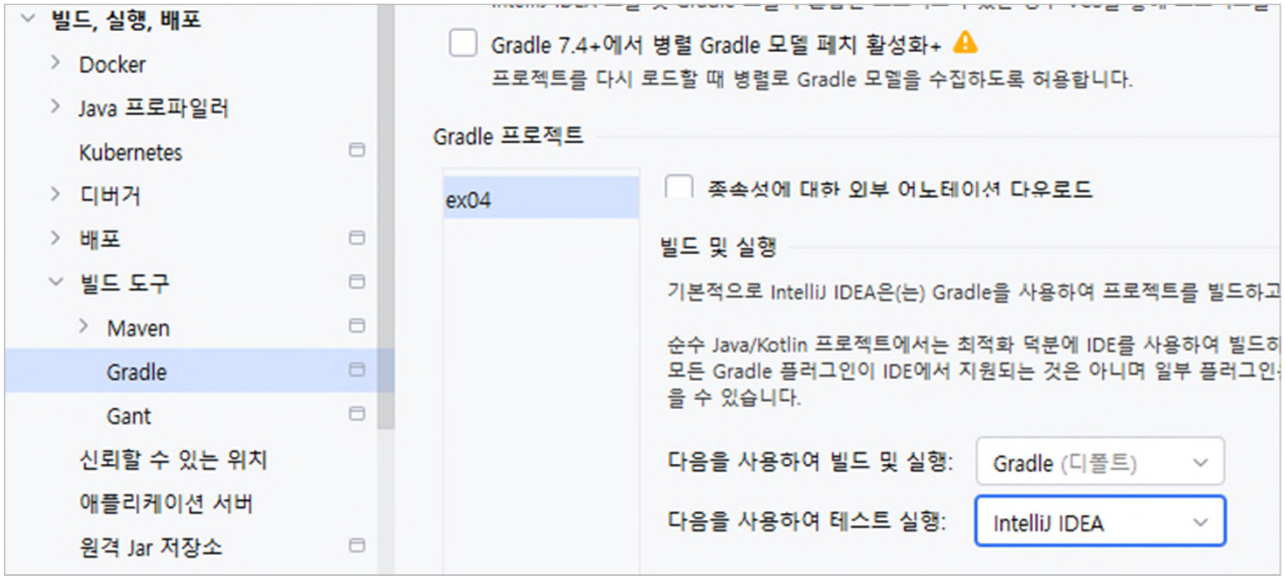


settings.gradle

```
rootProject.name = 'ex04'
```

설정

- Enable Annotation Processor 활성화
- 빌드, 실행, 배포 > 빌드 도구 > Gradle
  - 다음을 사용하여 테스트 실행: IntelliJ IDEA 선택



프로젝트의 JDBC 연결

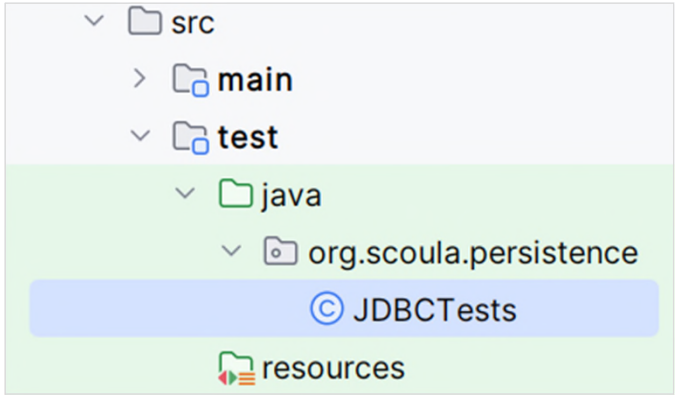
build.gradle

```
// 데이터베이스
implementation 'com.mysql:mysql-connector-j:8.1.0'
```

→ gradle sync

JDBC 테스트 코드

- src/test/java
  - org.scoula.persistence
    - JDBCTests.java



test :: JDBCTest.java

```
package org.scoula.persistence;

import lombok.extern.log4j.Log4j2;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

import java.sql.Connection;
import java.sql.DriverManager;
import static org.junit.jupiter.api.Assertions.fail;

@Log4j2
public class JDBCTest {
    @BeforeAll
    public static void setup() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
```

test :: JDBCTests.java

커넥션도 쓰레드 수만큼
준비시켜서
하나씩 쓰레드에 배정해줘야한다.

접속하는데 오버헤드가 많이
걸린다. 시간이 젤
많이 걸린다
쓰레드 풀 기억하는가

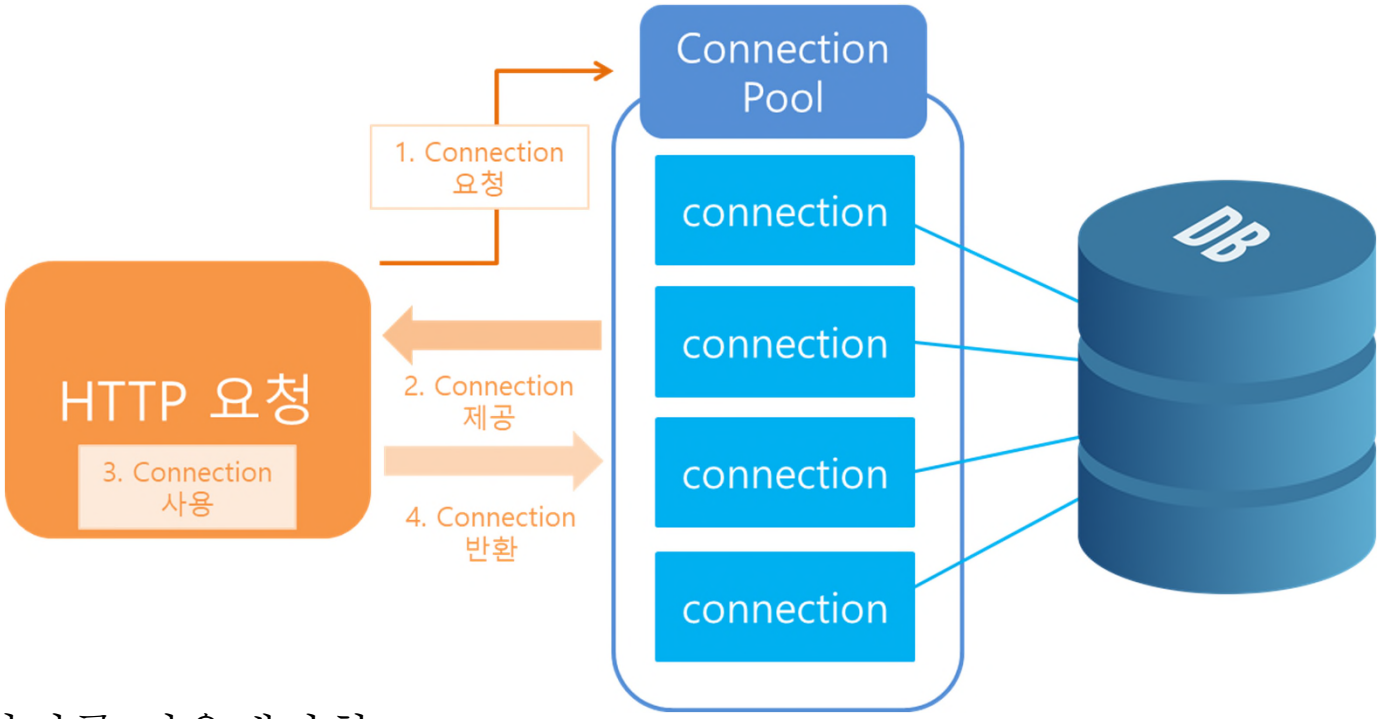
디비커넥션 풀을 만들어
운용해보자

```
    @Test
    @DisplayName("JDBC 드라이버 연결이 된다.")
    public void testConnection() {
        String url = "jdbc:mysql://localhost:3306/scoula_db";
        try(Connection con = DriverManager.getConnection(url, "scoula", "1234")) {
            log.info(con);
        } catch(Exception e) {
            fail(e.getMessage());
        }
    }

}
```

DataSource

- Connection Pool



라이브러리를 이용해야함
히카리커넥션풀

이라는 라이브러리
c3po라는 것도 있다

라이브러리 추가와 DataSource 설정

build.gradle

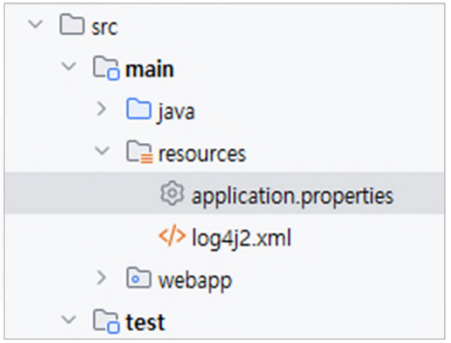
```
// 데이터베이스
implementation 'com.mysql:mysql-connector-j:8.1.0'
implementation 'com.zaxxer:HikariCP:2.7.4'
```

→ gradle sync

resources/application.properties

```
jdbc.driver=com.mysql.cj.jdbc.Driver
jdbc.url=jdbc:mysql://127.0.0.1:3306/scoula_db
jdbc.username=scoula
jdbc.password=1234
```

설정 .properties파일에
하드코딩된 접속 정보를
분리시키자.



.properties파일을 핸들링하는 어노테이션으로
관리, 읽어오기 등 동작을 진행할거야

번외 클래스패스 ㅁ== 클래스 서치 패스 == 클래스 찾을 때 쓰는 기준이 되는 패스

4 커넥션 풀 설정

- **.properties에서 값 읽어 오기**
  - @PropertySource({"properties 경로 문자열"})
    - 클래스 레벨 어노테이션
    - 사용할 .properties 경로를 지정
    - 예
      @PropertySource({"classpath:/application.properties"})
  - @Value("${키:기본값}")
    - 필드 레벨 어노테이션
    - 기본값은 생략 가능
    - 예
      @Value("${jdbc.driver}") String driver;

파일시스템패스말고
클래스패스로 보라고 명시

파일 여러개 지정할 수있음.
정보의 종류별로 분리된 파일을
지정할 수 잇다. 중괄호는 배열이죠?

내 전에 말했지? 디비 관련 연결 설정은 rootconfig 클래스에서 진행한다고

RootConfig.java

```
package org.scoula.config;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.jdbc.datasource.DataSourceTransactionManager;

import javax.sql.DataSource;

@Configuration
@PropertySource({"classpath:/application.properties"})
public class RootConfig {
    @Value("${jdbc.driver}") String driver;
    @Value("${jdbc.url}") String url;
    @Value("${jdbc.username}") String username;
    @Value("${jdbc.password}") String password;
```

다른 영역에서도 활용할 수 있는것. properties파일 정보를 키에 추출

이제 커넥션 풀을 빈으로 등록시킬거임

RootConfig.java

```
    @Bean
    public DataSource dataSource() {
        HikariConfig config = new HikariConfig();

        config.setDriverClassName(driver);
        config.setJdbcUrl(url);
        config.setUsername(username);
        config.setPassword(password);

        HikariDataSource dataSource = new HikariDataSource(config);
        return dataSource;
    }
}
```

히카리 라이브러리에 있는
객체를 이용해서

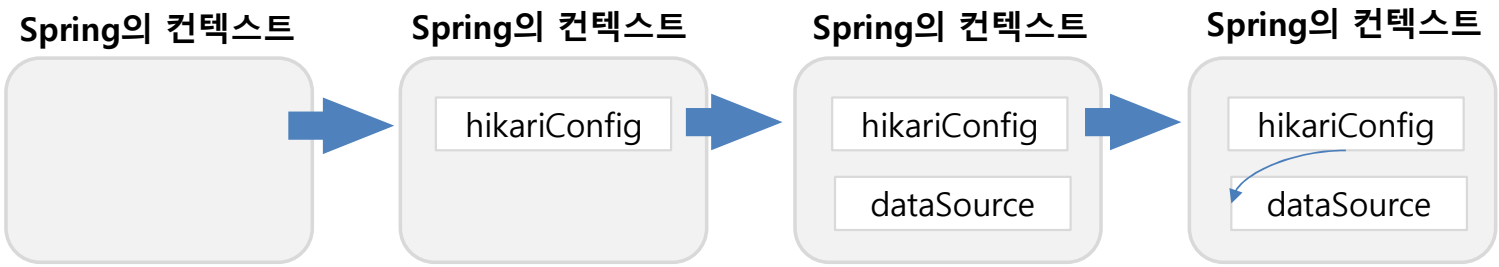
DBCpool을 생성해서
반환!

언제 빈을 이용해서 등록시키라했지
라이브러리에 있는 객체를 등록할때는
우리가 소스 핸들링을 못하니까

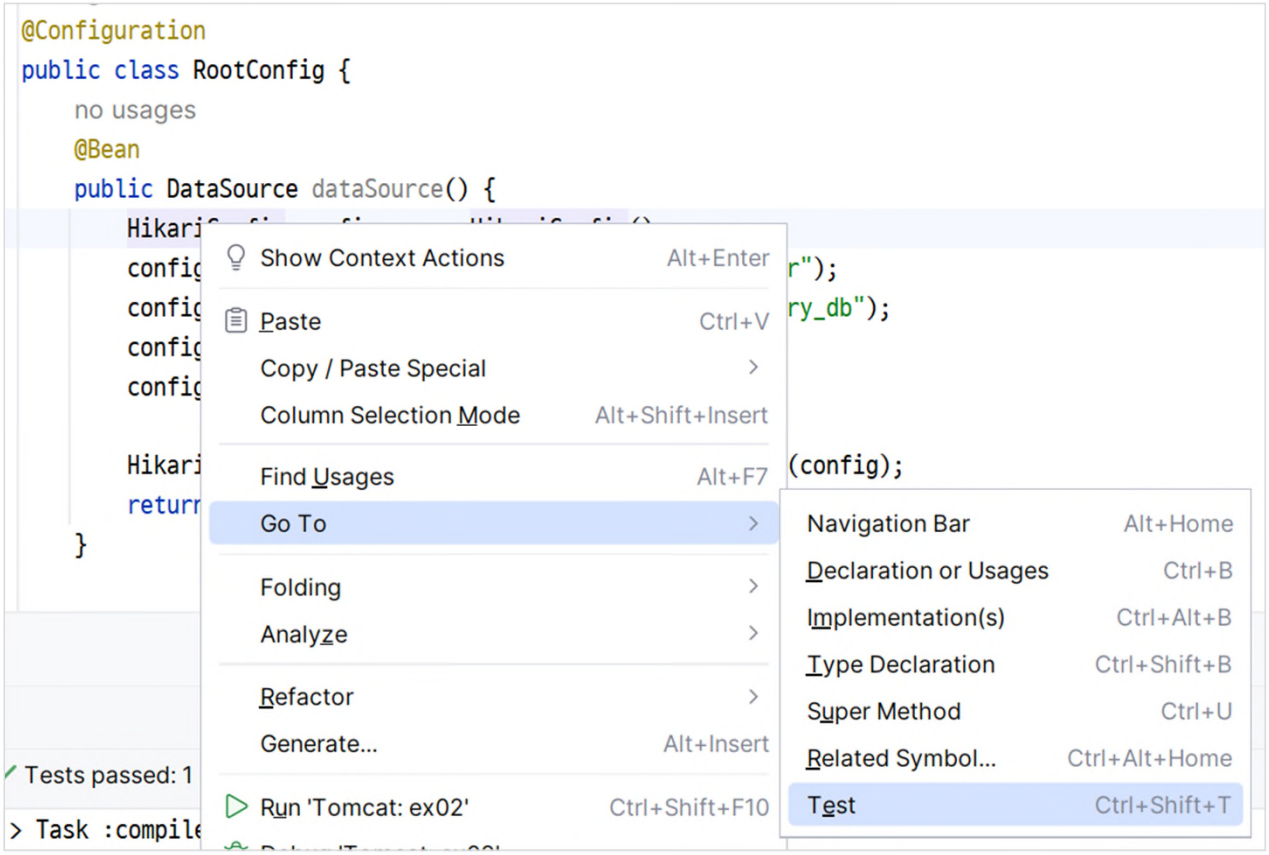
반환된 객체를 빈으로 등록시킴
빈의 이름은 메서드명인거 잊지말고

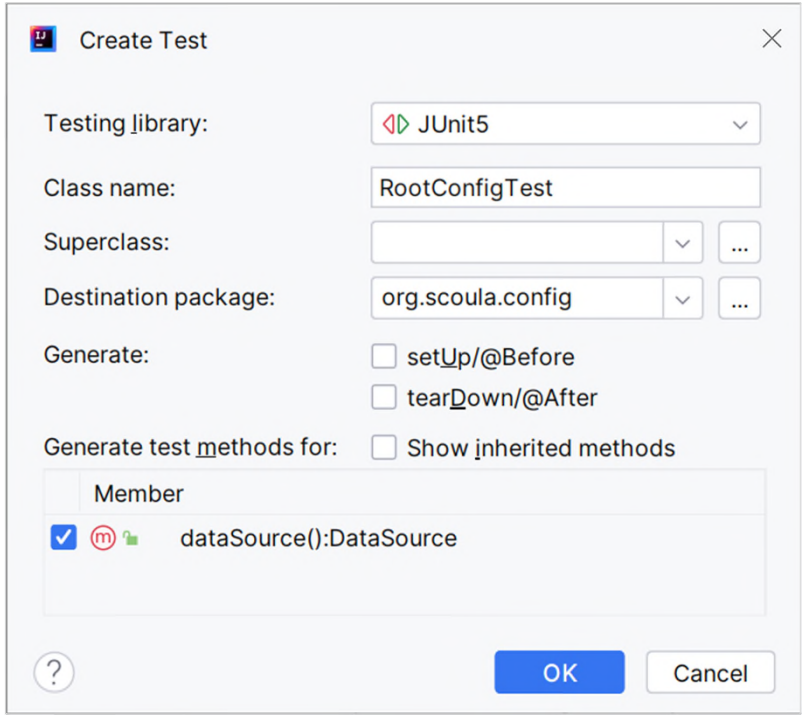
우리가 정의하고 만들수 있고 소스핸들링이
되는 객체 타입을 등록시킬땐 @Component

라이브러리 추가와 DataSource 설정



RootConfig 테스트 클래스 만들기





test :: config.DataSourceTests.java

```
@ExtendWith(SpringExtension.class)
@ContextConfiguration(classes= {RootConfig.class})
@Log4j2
class RootConfigTest {

  @Autowired
  private DataSource dataSource;

  @Test
  @DisplayName("DataSource 연결이 된다.")
  public void dataSource() throws SQLException {
    try(Connection con = dataSource.getConnection()){
      log.info("DataSource 준비 완료");
      log.info(con);
    }
  }
}
```

스프링 기반 테스트 진행시 항상 사용하는 어노테이션

DI요청했구

테스트는 눈으로 보고 성공 실패 구분하면 안돼.
테스트는 일일이 확인하면 안돼

테스트는 자동화되는 것이 원칙

지금은 학습이니까 로그 찍는거야

원래는 모든 테스트가 저 초록색 마크가 뜨느냐 마느냐만 본다. 모든 단위테스트 글자가 아닌 빨간색이 뜨는지.