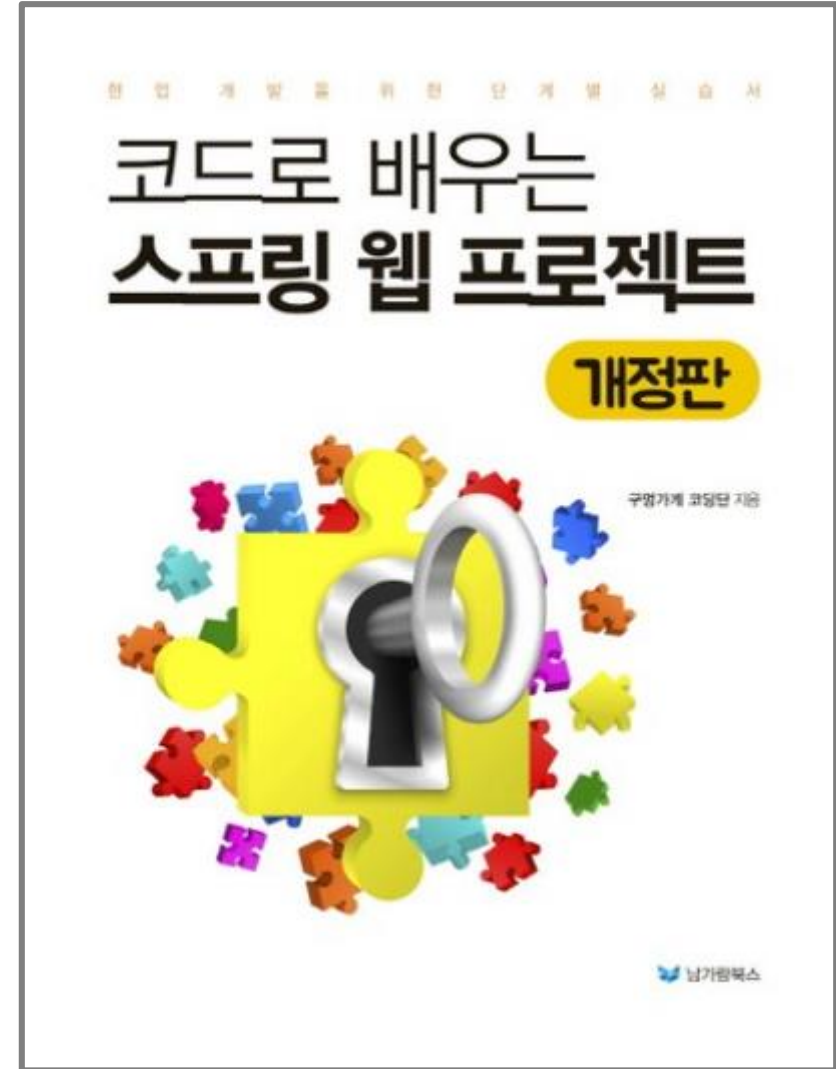


3장. 스프링과 Mariadb연동

동의과학대학교
컴퓨터정보과
김진숙

학습 목표

- 스프링의 개발 환경 (STS(혹은 Eclipse), Lombok 등)
- Mariadb 데이터베이스 설치 및 계정 설정
- 스프링과 MyBatis의 연동 설정
- 스프링 MVC의 구성 설정 및 테스트



pom.xml 추가 – spring-jdbc/spring-tx

```
<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>
```

pom.xml 추가 – mariadb 2.7.5

- mvnrepository에서 적당한 버전을 골라 Pom.xml 추가

The screenshot shows the Maven Repository search results for 'mariadb'. The search bar contains 'mariadb' and the results show 'Found 110 results'. The first result is '1. MariaDB Java Client' by 'org.mariadb.jdbc', with '660 usages' and '660 usages' (repeated). The license is 'LGPL'. The last release is on 'Sep 20, 2022'. The left sidebar shows various repository categories like Central, Sonatype, Spring Plugins, etc.

The screenshot shows the artifact page for 'org.mariadb.jdbc:mariadb-java-client:2.7.5'. The page includes a graph of indexed artifacts (30.4M), a list of popular categories, and detailed information about the artifact. The license is 'LGPL 2.1', and the categories are 'JDBC Drivers'. The tags include 'database', 'sql', 'jdbc', 'driver', 'client', and 'mysql'. The organization is 'mariadb.com', and the homepage is 'https://mariadb.com/kb/en/mariadb/about-mariadb-connector-j/'. The date is 'Jan 19, 2022'. The files are 'pom (13 KB)' and 'jar (608 KB)'. The repositories are 'Central'. The ranking is '#616 in MvnRepository (See Top Artifacts)' and '#7 in JDBC Drivers'. The artifact is used by '660 artifacts'. A note indicates there is a new version for this artifact.

```
<!-- https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client -->
<dependency>
  <groupId>org.mariadb.jdbc</groupId>
  <artifactId>mariadb-java-client</artifactId>
  <version>2.7.5</version>
</dependency>
```

```
<dependency>
  <groupId>org.mariadb.jdbc</groupId>
  <artifactId>mariadb-java-client</artifactId>
  <version>2.7.5</version>
</dependency>
```

JDBC연결 및 테스트

```
10 @Log4j
11 public class JDBCTests {
12
13     @Test
14     public void testConnection() {
15         try {
16             Class class1 = Class.forName("org.mariadb.jdbc.Driver");
17
18             Log.info(class1);
19
20             Connection con = DriverManager.getConnection(
21                 "jdbc:mariadb://localhost:3306/jinsookdb", "jinsook", "1111");
22             Log.info(con);
23
24             con.close();
25
26         } catch (Exception e) {
27             e.printStackTrace();
28         }
29
30     }
31 }
```

프로토콜 :
"jdbc:mariadb://localhost:3306/jinsookdb"

테스트 성공 - 커넥션 생성

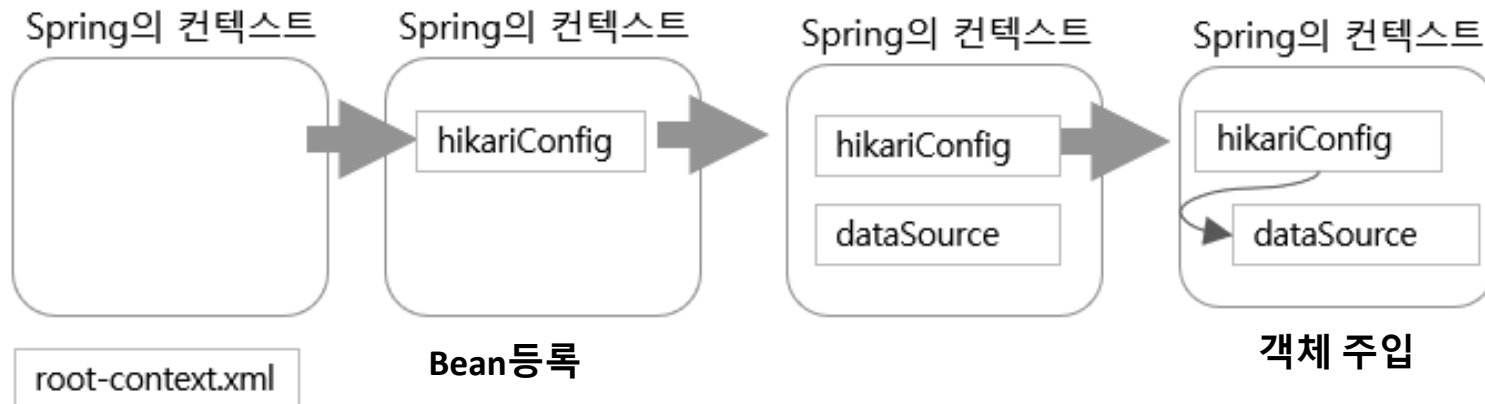
Console Progress Problems
<terminated> JDBCTests.testConnection [JUnit] C:\JSP\jdk-14.0.2\bin\javaw.exe (2022. 10. 23. 오전 12:40:58 - 오전 12:40:59)
INFO : cs.dit.persistence.JDBCTests - class org.mariadb.jdbc.Driver
INFO : cs.dit.persistence.JDBCTests - org.mariadb.jdbc.MariaDbConnection@382db087

pom.xml – HikariCP 3.4.5

- DB와 Connection을 맺고 끊는 작업은 리소스의 소모가 많은 작업
- **Pooling**이라는 기법을 통해서 객체를 미리 생성하고 빌려 쓰는 방식으로 이용해서 연결 시간을 단축(Connection Pool)
- Commons DBCP나 HikariCP등을 활용(HikariCP가 성능이 더 좋음)

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

Pom.xml



DataSource 설정 – root-context.xml

```
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">  
  <property name="driverClassName" value="org.mariadb.jdbc.Driver" />  
  <property name="jdbcUrl" value="jdbc:mariadb://localhost:3306/jinsookdb"/>  
  <property name="username" value="jinsook"/>  
  <property name="password" value="1111" />  
</bean>
```

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

- Hikari CP 도움말 사이트

<https://github.com/brettwooldridge/HikariCP#configuration-knobs-baby>

root-context.xml

- 스프링이 로딩되면서 읽어 들이는 문서이므로, 주로 이미 만들어진 클래스를 스프링 빈으로 등록할 때 사용됨
- 일반적으로 프로젝트에서 **직접 작성하는 클래스**들은 **어노테이션**을 사용
- **외부 jar** 파일의 클래스들은 **<bean> 태그**를 사용(소스 코드를 접근할 수 없으므로)

Hikari CP 연결 테스트

```
@RunWith(SpringJUnit4ClassRunner.class) //현재 테스트 코드가 스프링 실행 역할을 할 것이라고 알림
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml") //설정파일 읽어내기
@Log4j //lombok을 이용해 로그를 기록하는 Logger를 변수로 생성
public class JDBCTests {

    @Autowired
    private DataSource ds;

    @Test
    public void testCP() {

        try(Connection con = ds.getConnection());
        {
            Log.info("testCP.....");
            Log.info(con);

        } catch (Exception e) {
            e.printStackTrace();
            Log.error(e.getMessage());
        }

    }
}
```

테스트 성공 - 커넥션 생성

```
WARN : com.zaxxer.hikari.util.DriverDataSource - Registered driver with driverClassN
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
INFO : cs.dit.persistence.JDBCTests - con : -----
INFO : cs.dit.persistence.JDBCTests - HikariProxyConnection@1470966439 wrapping orac
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```