4장. MyBatis와 스프링 연동

동의과학대학교
컴퓨터정보과
김진숙

학습내용

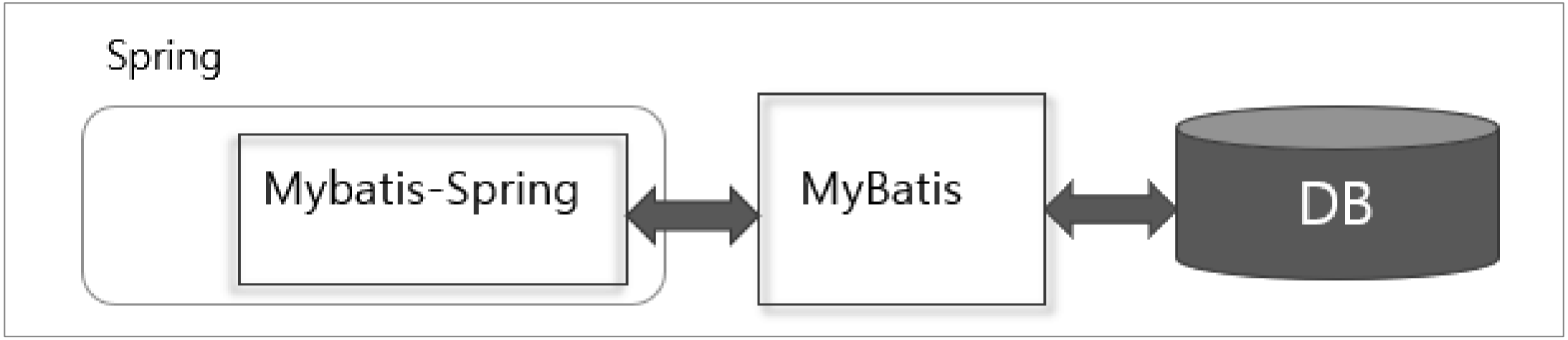
- MyBatis 세팅
- spring-mybatis 라이브러리 설정
- Mapper 인터페이스 설정 확인
- XML Mapper 파일
- log4jdbc-log4j2 설정

MyBatis

JPA(Java Persistence Api)
- 관계지향 -> 객체지향을 접목
- ORM(Object-Relational Mapping) 기술 표준

- SQL Mapping 프레임워크
  - SQL과 Object간의 관계를 매핑해주는 유틸리티
  - SQL을 그대로 사용
  - JDBC코드에 비해 처리하는 부분이 간결해지고, close처리등이 지원

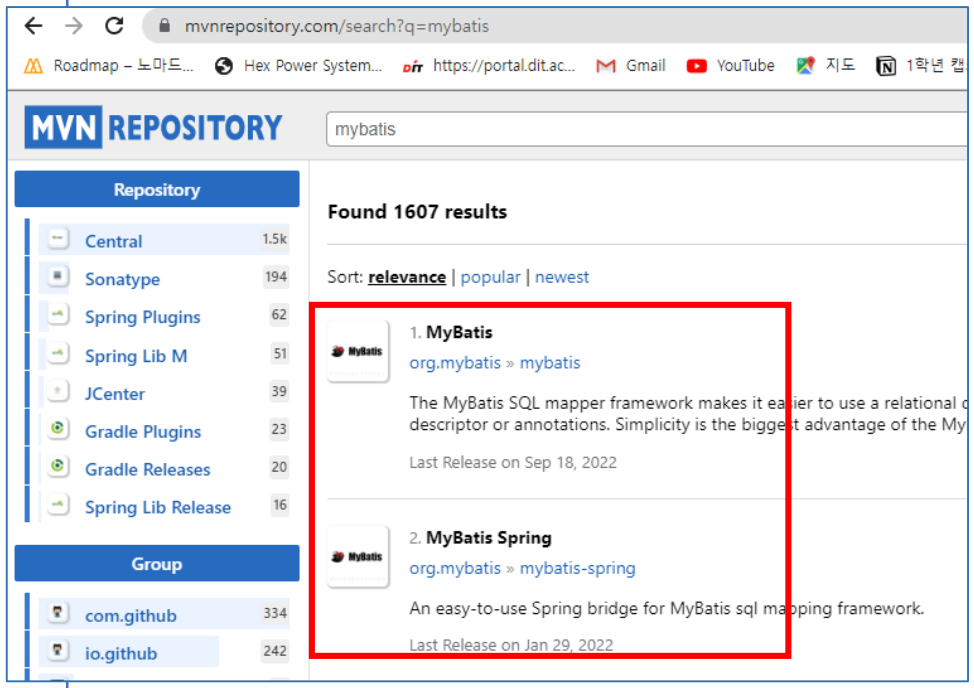
- Spring에서의 사용
  - 스프링은 MyBatis와의 연결을 위한 mybatis-spring 라이브러리을 이용해서 연동 처리



Pom.xml 추가

- mybatis-spring의 설정

```
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.5.6</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>2.0.6</version>
</dependency>
```



Pom.xml 추가

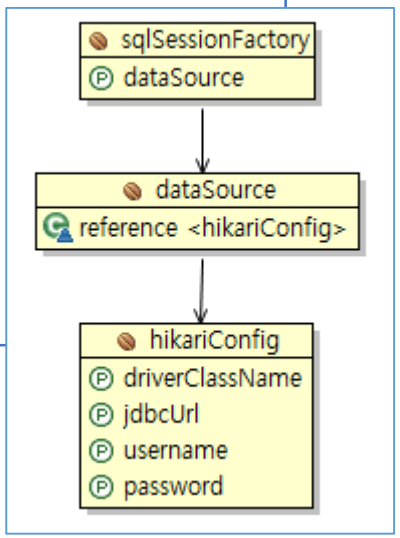
- 관련 spring 라이브러리 추가
  - spring-jdbc : 데이터베이스 연동 관련 스프링 라이브러리
  - spring-tx : 스프링 트랜잭션 라이브러리

```
<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>
```

SqlSessionFactory의 설정

- **root-context.xml**에 MyBatis설정
- MyBatis의 핵심 객체는 SqlSessionFactory타입의 객체 **Connection 객체의 역할**
- SqlSessionFactoryBean은 내부적으로 MyBatis의 SqlSessionFactory를 생성 (mybatis-spring 라이브러리의 클래스)

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

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



Connection을 Session으로 표기하기도 함

Mybatis 설정 테스트

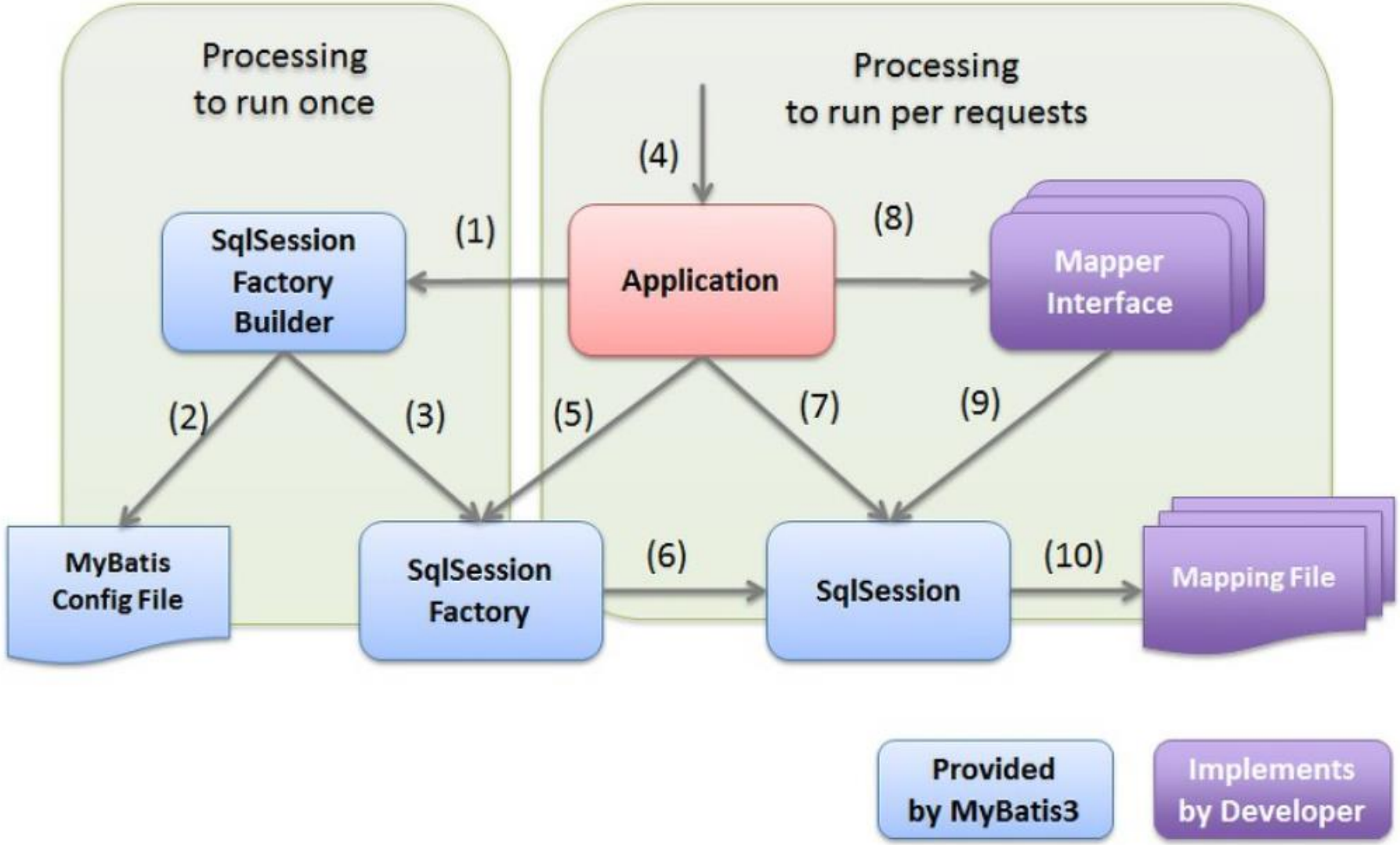
```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
@Log4j
public class DataSourceTests {
        ……
        @Autowired
        private SqlSessionFactory sessionFactory;

        @Test
        public void testMybatis() {

                try(   SqlSession session = sessionFactory.openSession();
                        Connection con = session.getConnection() )
                {       log.info("sqlSession----------------------------");
                        log.info(session);
                        log.info(con);
                }catch(Exception e){
                        e.printStackTrace();
                }
        }
}
```

```
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
INFO : cs.dit.persistency.JDBCTests - testMybatis
INFO : cs.dit.persistency.JDBCTests - org.apache.ibatis.session.defaults.DefaultSqlSession@32193bea
INFO : cs.dit.persistency.JDBCTests - HikariProxyConnection@1804441305 wrapping org.mariadb.jdbc.Connection@69653e16
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

MyBatis

커넥션풀의 커넥션

Mybatis 구조


Processing
to run once
Processing
to run per requests
(4)
SqlSession
Factory
Builder
(1)
Application
(8)
Mapper
Interface
(2)
(3)
(5)
(7)
(9)
MyBatis
Config File
SqlSession
Factory
(6)
SqlSession
(10)
Mapping File
Provided
by MyBatis3
Implements
by Developer

Mybatis 동작 과정

(1) 응용 프로그램이 SqlSessionFactoryBuilder를 위해 SqlSessionFactory를 빌드하도록 요청

(2) SqlSessionFactoryBuilder는 SqlSessionFactory를 생성하기 위한 Mybatis 구성 파일을 읽음

(3) SqlSessionFactoryBuilder는 Mybatis 구성 파일의 정의에 따라 SqlSessionFactory를 생성

(4) 클라이언트가 응용 프로그램에 대한 프로세스를 요청

(5) 응용프로그램은 SqlSessionFactoryBuilder로 빌드된 SqlSessionFactory에서 SqlSession을 가져옴

(6) SqlSessionFactory는 SqlSession을 생성하고 이를 애플리케이션에 반환

(7) 응용 프로그램이 SqlSession에서 매퍼 인터페이스의 구현 개체를 가져옴

(8) 응용 프로그램이 매퍼 인터페이스 메서드를 호출

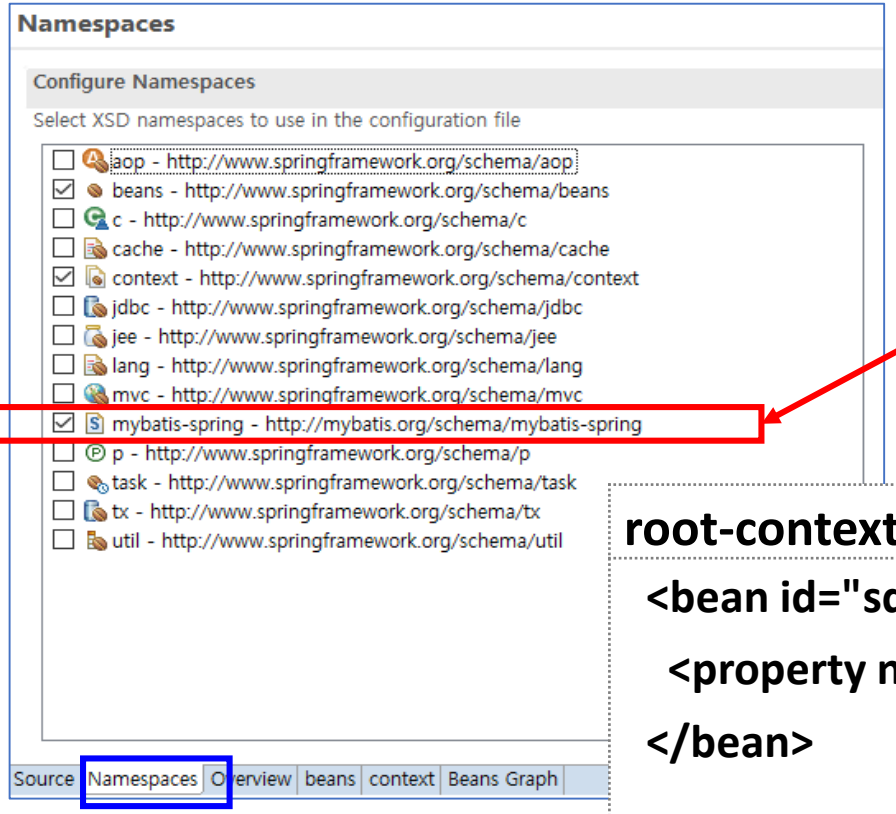
(9) 매퍼 인터페이스의 구현 개체가 SqlSession 메서드를 호출하고 SQL 실행을 요청

(10) SqlSession은 매핑 파일에서 실행할 SQL을 가져와 SQL을 실행

(1) ~ (3)은 응용 프로그램 시작시 수행되는 프로세스
(4) ~ (10)은 클라이언트의 각 요청에 대해 수행되는 프로세스

Mapper인터페이스의 설정/인식

- MyBatis를 사용하는 방식으로는 Mapper인터페이스를 작성하고 자동으로 생성되는 클래스를 사용하는 방식을 이용



해당 패키지에 대한 자동완성 기능 제공됨

root-context.xml의 일부

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource“ />
</bean>

<mybatis-spring:scan base-package="cs.dit.mapper"/>
<context:component-scan  base-package="cs.dit.sample"/>
```

Mapper인터페이스

- MyBatis를 사용하는 방식으로 Mapper인터페이스를 작성하고 자동으로 생성되는 클래스를 사용하는 방식을 이용
- Mapping 에 기재된 SQL을 호출하기 위한 인터페이스
  - 어노테이션 사용
  - XML 파일 사용

TimeMapper 인터페이스

```
package cs.dit.mapper;

import org.apache.ibatis.annotations.Select;

public interface TimeMapper {
    @Select("SELECT now()")
    public String getTime01();
}
```

어노테이션을 이용하여 SQL 문 작성

- 장점 : 쉽다
- 단점 : 긴 문장의 SQL 작성이 어렵다
  동적 SQL문 작성이 어렵다

Mapper인터페이스의 설정 확인

- spring-test를 이용해서 테스트 코드 작성

```
@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
public class TimeMapperTests {

 @Autowired
 private TimeMapper timeMapper;

 @Test
 public void testGetTime01() {
        log.info("mybatis-test------------------------");
        log.info(timeMapper.getClass().getName());
        log.info(timeMapper.getTime());
 }

}
```

스프링이 인터페이스를 이용하여 객체 생성

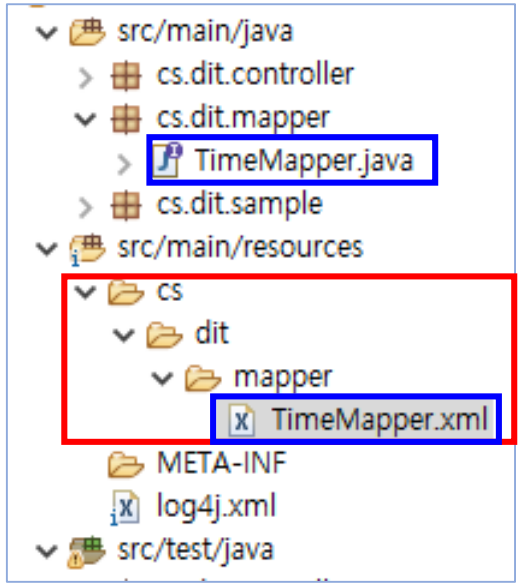
```
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
INFO : cs.dit.mapper.TimeTests - TimeMapper test.....
INFO : cs.dit.mapper.TimeTests - com.sun.proxy.$Proxy21
INFO : cs.dit.mapper.TimeTests - 2022-11-08 21:06:00
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
INFO : com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

실제 동작하는 클래스 명

시간

XML Mapper + Mapper 인터페이스

- xml 파일로 매퍼 작성하기
  - src/main/resources 구조에 xml을 저장할 폴더 생성
  - 폴더 구조를 다음과 같이 하나하나 만들고, 가능한 **Mapper 인터페이스와 xml 매퍼는 같은 이름**을 사용하면 가독성이 증가함



TimeMapper.xml

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


<mapper namespace="cs.dit.mapper.TimeMapper">
  <select id="getTime02" resultType="string">
        SELECT now()
  </select>
</mapper>
```

Java.lang.String의 alias

TimeMapper 인터페이스

```
package org.zerock.mapper;
import org.apache.ibatis.annotations.Select;

public interface TimeMapper {

  @Select("SELECT now()")
  public String getTime01();

  public String getTime02();
}
```

Mapper인터페이스의 설정 확인

- spring-test를 이용해서 테스트 코드 작성

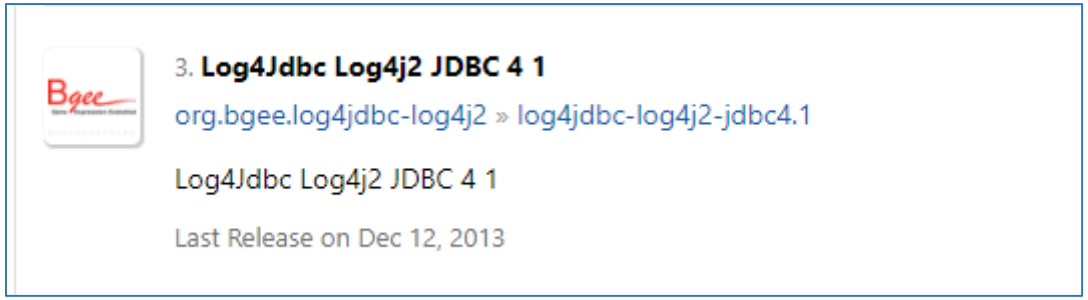
```
@RunWith(SpringJUnit4ClassRunner.class) //현재 테스트 코드가 스프링 실행 역할을 할 것이라고 알림
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml") //설정파일 읽어내기
@Log4j
public class TimeMapperTests {

 @Autowired
 private TimeMapper timeMapper;

 @Test
 public void testGetTime02() {
        log.info("mybatis-test-------------------------");
        log.info(timeMapper.getClass().getName());
        log.info(timeMapper.getTime());
 }
 @Test
 public void testGetTime01() {
        log.info("mybatis-test-------------------------");
        log.info(timeMapper.getClass().getName());
        log.info(timeMapper.getTime());
 }
}
```

SQL 로그 : log4jdbc-log4j2 설정

- MyBatis는 내부적으로 PreparedStatement를 이용하기 때문에 좀 더 쉽게 **자세한 SQL의 로그**를 보기 위한 설정
  - log4jdbc-log4j2는 **DB, JDBC 버전에 따라 지원 여부가 결정**되기 때문에 버전 선택에 신중해야 함 ➔ 동작하지 않을 수 있음
  - PreparedStatement의 ?가 어떤 값으로 처리되었는지 확인하는 기능이 추가됨
- 설정 순서
  1. 라이브러리 추가(pom.xml)
  2. 프로퍼티 파일 추가
  3. DataSource 설정 변경(root-context.xml)

SQL 로그 : log4jdbc-log4j2 설정

1. 라이브러리 추가(maven repository에서)



pom.xml 일부

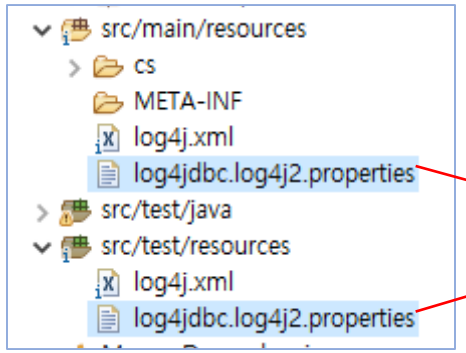
```
<!--  logging -->
<dependency>
   <groupId>org.bgee.log4jdbc-log4j2</groupId>
   <artifactId>log4jdbc-log4j2-jdbc4.1</artifactId>
   <version>1.16</version>
</dependency>
```

SQL 로그 : log4jdbc-log4j2 설정

2. log4jdbc.log4j2.properties 작성(파일)
   - 파일 내용을 복사하여 파일에 붙여둔다.
   - 테스트를 위해 src/test/resource에도 복사해 둔다.

파일명

log4jdbc-log4j2.properties



파일내용

log4jdbc.spylogdelegator.name = net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator

SQL 로그 : log4jdbc-log4j2 설정

3. DataSource 설정 변경
   - 아래의 value 값을 복사하여 붙여 넣는다..

root-contest.xml 일부 변경

```
<property name="driverClassName"   value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"/></property>
<property name="jdbcUrl"    value="jdbc:log4jdbc:mariadb://localhost:3306/jinsookdb"></property>
```

```
10    <!-- Root Context: defines shared resources visible to all other web components -->
11    <bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
12  <!--     <property name="driverClassName" value="org.mariadb.jdbc.Driver"/> -->
13  <!--     <property name="jdbcUrl" value="jdbc:mariadb://localhost:3306/jinsookdb"/> -->
14
15      <property name="driverClassName"  value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"/>
16      <property name="jdbcUrl"    value="jdbc:log4jdbc:mariadb://localhost:3306/jinsookdb"/>
17
18      <property name="username" value= "jinsook"/>
19      <property name="password" value= "1111"/>
20    </bean>
```

TimeMapperTests()

```java
package cs.dit.mapper;

import org.apache.ibatis.annotations.Select;

public interface TimeMapper {

    @Select("select now()")
    public String getTime();

    public String getTime2();
}
```

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

    @Autowired
    private TimeMapper timeMapper;

    @Test
    public void testGetTime() {//SQL어노테이션 사용
        Log.info("mybatis-test--------------------------");
        Log.info(timeMapper.getClass().getName());//실제 동작하는 클래스의 이름 확인
        Log.info(timeMapper.getTime());//
    }
    @Test
    public void testGetTime2() {//SQL 매퍼 사용
        Log.info("mybatis-test--------------------------");
        Log.info(timeMapper.getClass().getName());//실제 동작하는 클래스의 이름 확인
        Log.info(timeMapper.getTime2());//
    }
}
```

```
INFO : jdbc.resultset - 1. ResultSet.getMetaData() returned org.mariad
INFO : jdbc.resultset - 1. ResultSet.getType() returned 1003
INFO : jdbc.resultset - 1. ResultSet.isClosed() returned false
INFO : jdbc.resultset - 1. ResultSet.next() returned true
INFO : jdbc.resultset - 1. ResultSet.getString(now()) returned 2022-11
INFO : jdbc.resultset - 1. ResultSet.isClosed() returned false
INFO : jdbc.resultsettable -
|---------------------|
|now()                |
|---------------------|
|2022-11-09 12:42:11.0 |
|---------------------|

INFO : jdbc.resultset - 1. ResultSet.next() returned false
INFO : jdbc.resultset - 1. ResultSet.close() returned void
INFO : jdbc.audit - 1. Connection.getMetaData() returned org.mariadb.j
INFO : jdbc.audit - 1. PreparedStatement.getMoreResults() returned fal
```

자세한 로그가 출력됨