



[한국ICT인재개발원] 스프링 프레임워크

3. Mybatis 연동하기

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

마이바티스 세팅하기

Mybatis, mybatis-spring에 대한 이해
Mapper 인터페이스로 SQL 추가하기

2

Mapper xml 분리 이해하기

XML 파일에 쿼리 작성하기
테스트하기

3

콘솔창에 DB조회 결과 출력하기

Log4jdbc-log4j2 설정하기
테스트하기

4

실제 문제 풀어보기

JSP의 DAO를 살펴보면

자바 코드 내에 SQL구문을 String형태로 작성해서

쿼리문을 완성시켜 전달하는 방식이었습니다.

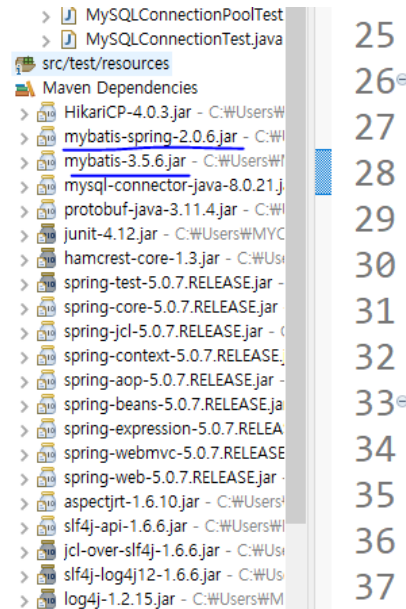
이 방식의 최대 단점은 자바 코드와 SQL 구문이라는

이질적인 두 개의 구문이 하나의 로직에 섞여서

가독성을 해치는 문제가 있었고

이를 해결하기 위해 자바 로직과 SQL 구문을 분리할 목적으로

마이바티스를 사용합니다.



```
25 <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring/2.0.6 -->
26 <dependency>
27     <groupId>org.mybatis</groupId>
28     <artifactId>mybatis-spring</artifactId>
29     <version>2.0.6</version>
30 </dependency>
31
32 <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis/3.5.6 -->
33 <dependency>
34     <groupId>org.mybatis</groupId>
35     <artifactId>mybatis</artifactId>
36     <version>3.5.6</version>
37 </dependency>
```

Mvnrepository.com 에 접속한 다음

Mybatis, mybatis-spring 을 세팅합니다.

Mybatis-spring이 스프링의 실행문을 받아 mybatis에 전달하는 역할을 합니다.



The screenshot shows an IDE with two panels. The left panel displays the 'Maven Dependencies' tree, listing various JAR files including HikariCP, mybatis-spring, mybatis, spring-jdbc, spring-beans, spring-core, spring-jcl, spring-tx, mysql-connector-java, protobuf, junit, hamcrest, spring-test, spring-context, spring-aop, spring-expression, spring-webmvc, and spring-web. The right panel shows the corresponding XML configuration in the pom.xml file, with lines 32 through 44. The configuration includes a dependency for mybatis (version 3.5.6) and a dependency for spring-jdbc (version 5.0.7.RELEASE). The mybatis dependency is highlighted in blue.

```
32 <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis/3.5.6 -->
33 <dependency>
34     <groupId>org.mybatis</groupId>
35     <artifactId>mybatis</artifactId>
36     <version>3.5.6</version>
37 </dependency>
38
39 <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
40 <dependency>
41     <groupId>org.springframework</groupId>
42     <artifactId>spring-jdbc</artifactId>
43     <version>5.0.7.RELEASE</version>
44 </dependency>
```

다음 다시 Mvnrepository.com 에 접속한 다음

Spring-tx, spring-jdbc를 5.0.7버전에 맞게 세팅합니다.

위 jar파일들도 mybatis 실행에 영향을 주기 때문에 반드시 입력합니다.

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

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

sqlSessionFactory가 바로 스프링 구문을 받아 mybatis로 전달하는 객체입니다.

이 객체는 DB연결을 위해 dataSource를 필요로 하니 주입해줍니다.

```
@Autowired
private SqlSessionFactory sqlSessionFactory;

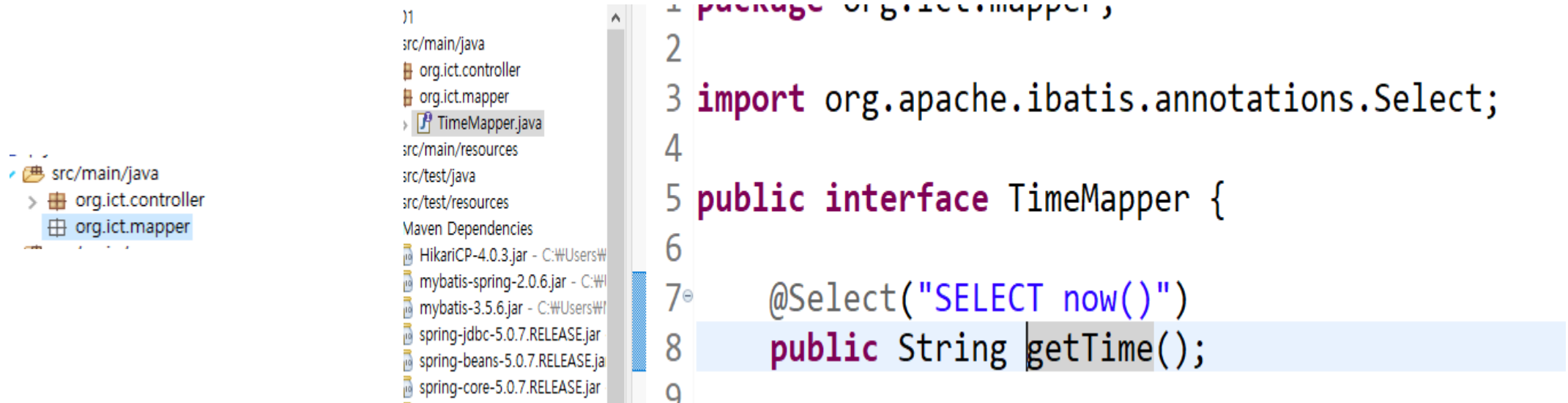
@Test
public void testMyBatis() {
    try(SqlSession session = sqlSessionFactory.openSession();
        Connection con = session.getConnection();){
        Log.info(session);
        Log.info(con);
    } catch (Exception e) {
        fail(e.getMessage());
    }
}
```

테스트 코드는 아까 썼던 MySQLConnectionPoolTest.java를 재활용합니다.

위와 같이 SqlSessionFactory를 이용해 마이바티스에 명령이 들어가는지
체크합니다.


```
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 - org.apache.ibatis.session.defaults.D  
org.ict.dao.MySQLConnectionPoolTest - HikariProxyConnection@1664165134 wra  
org.springframework.context.support.GenericApplicationContext - Closing or  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

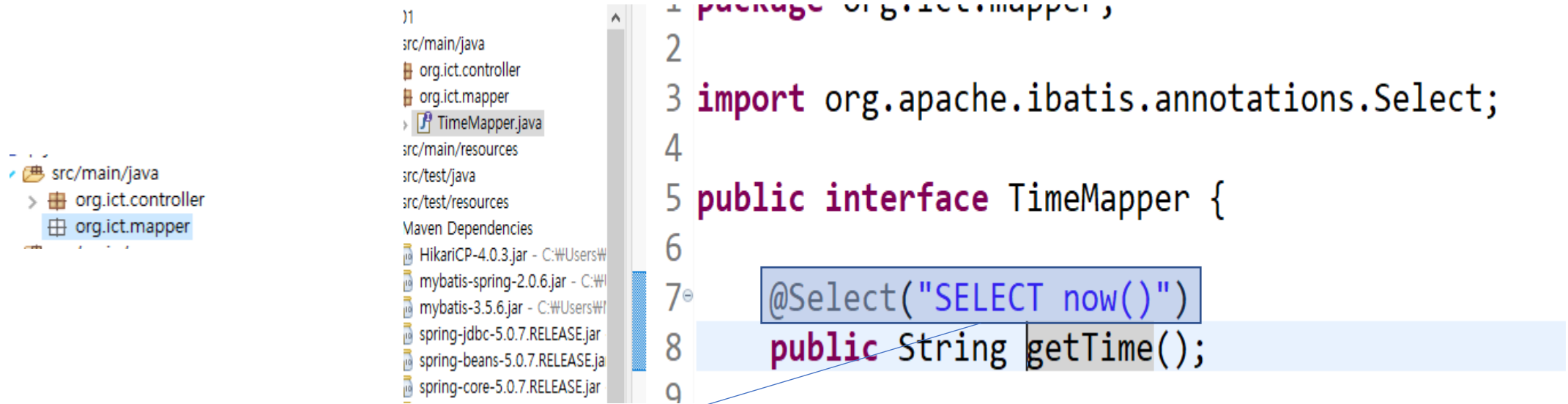
명령이 제대로 들어가는 경우 위와 같이 ibatis 관련 구문이 함께 출력됩니다.



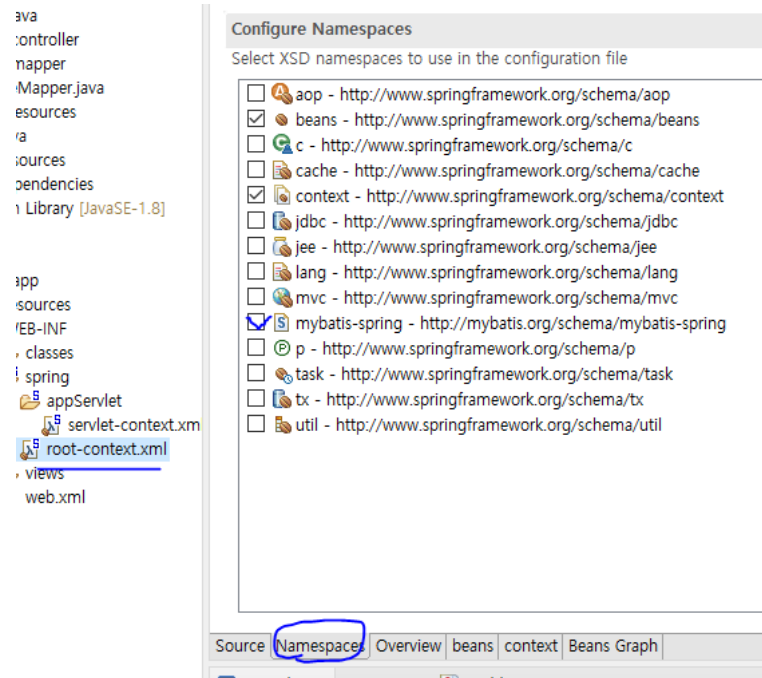
Org.ict.mapper 패키지를 생성하고, TimeMapper 인터페이스를 만듭니다.

마이바티스는 메서드 선언을 특이하게 인터페이스로 하며

선언된 메서드의 구현은 xml파일로 진행합니다.



상단의 @Select 어노테이션 안에는 SELECT 구문을 작성할 수 있으며
getTime() 메서드가 호출될 때 현재 시간을 출력하도록 쿼리문을 작성했습니다.

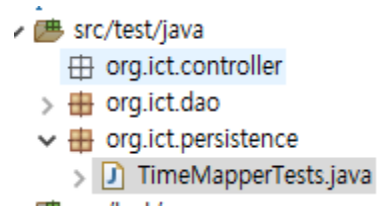


</update>

```
<mybatis-spring:scan base-package="org.ict.mapper"/>
```

Root-context.xml 하단의 namespace 탭으로 가 mybatis-sprin을 체크합니다.

그리고 내부에는 아까 작성한 org.ict.mapper 패키지를 스캔하도록 합니다.



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

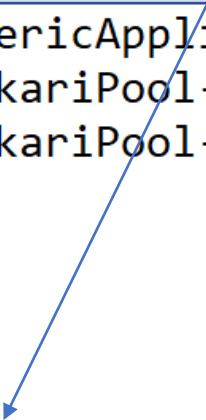
    @Autowired
    private TimeMapper timeMapper;

    @Test
    public void testGetTime() {
        Log.info("현재 시간 조회중...");
        Log.info(timeMapper.getTime());
    }
}
```

테스트 코드 작성을 위해 src/test/java 하위에 org.ict.persistence를 생성하고
테스트 코드를 작성합니다.

인터페이스 TimeMapper 자료형을 그대로 선언하고 의존성 주입을 하면
자동으로 마이바티스가 구현클래스화 해줍니다.

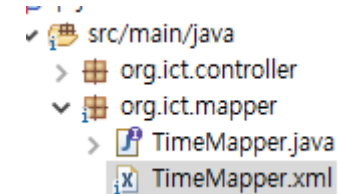
```
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcess  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.  
org.ict.persistence.TimeMapperTests - 현재 시간 조회중...  
org.ict.persistence.TimeMapperTests - 2021-03-11 01:26:59  
org.springframework.context.support.GenericApplicationContext - Closing org.spr  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...  
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```



실행 결과창에는 보시는 바와 같이 `SELECT now()` 구문이 그대로 실행됩니다.

구현한 적 없는 `TimeMapper` 인터페이스의 구현클래스가 활용된것입니다.

```
public interface TimeMapper {  
  
    @Select("SELECT now()")  
    public String getTime();  
  
    public String getTime2();  
  
}
```



이번엔 TimeMapper에 @Select도 활용하지 않은 메서드를 선언하겠습니다.

그리고 getTime2() 메서드가 사용할 쿼리문을 저장할 XML파일을 생성하겠습니다.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
4 <mapper namespace="org.ict.mapper.TimeMapper">
5
6     <select id="getTime2" resultType="string">
7         SELECT now()
8     </select>
9
10 </mapper>
```

XML 파일의 1~3번라인의 스키마는 복사해서 쓰시면 됩니다.
4번 라인에서는 어떤 인터페이스를 이 xml파일이 구현할지 경로를 적고

4~10번라인의 여닫는 부분 사이에 구문 종류에 따라 태그를 정하고
id에는 구현할 메서드 이름을, resultType에는 리턴할 자료형을 적은뒤
여닫는 태그 사이에 쿼리문을 ;없이 작성합니다.


```
public class TimeMapperTests {  
  
    @Autowired  
    private TimeMapper timeMapper;  
  
    @Test  
    public void testGetTime2() {  
        Log.info("getTime2가 얻어온 시간");  
        Log.info(timeMapper.getTime2());  
    }  
  
    // @Test  
    public void testGetTime() {
```

```
story.annotation.AutowiredAnnotationBe  
source - HikariPool-1 - Starting...  
source - HikariPool-1 - Start complete  
erTests - getTime2가 얻어온 시간  
erTests - 2021-03-11 01:39:33  
support.GenericApplicationContext - C]  
source - HikariPool-1 - Shutdown initi  
source - HikariPool-1 - Shutdown comp]
```

테스트 코드를 간단히 짜서 돌려보면 역시 실행이 잘 됩니다.

앞으로 위와 같이 인터페이스에 메서드를 정의하고

Xml파일에 쿼리문을 작성해 자바 코드와 SQL 쿼리문을 분리해 관리합니다.

Log4jdbc-log4j2 도입으로 콘솔창에 sql로그 찍기

콘솔창에 DB조회 결과 출력하기 - 채종훈 강사

log4jdbc-log4j2-jdbc4

Found 337 results

Sort: **relevance** | popular | newest

 **1. Log4Jdbc Log4j2 JDBC 4**
org.bgee.log4jdbc-log4j2 » log4jdbc-log4j2-jdbc4
Log4Jdbc Log4j2 JDBC 4
Last Release on Dec 12, 2013

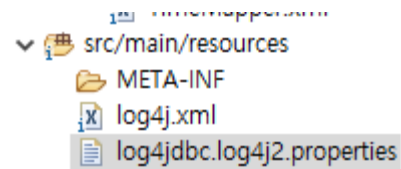
Maven Dependencies

- HikariCP-4.0.3.jar - C:\Users\W
- mybatis-spring-2.0.6.jar - C:\W
- mybatis-3.5.6.jar - C:\Users\W
- log4jdbc-log4j2-jdbc4-1.16.jar
- spring-jdbc-5.0.7.RELEASE.jar
- spring-beans-5.0.7.RELEASE.jar
- spring-core-5.0.7.RELEASE.jar
- spring-jcl-5.0.7.RELEASE.jar - C
- spring-tx-5.0.7.RELEASE.jar - C
- mysql-connector-java-8.0.21.j
- protobuf-java-3.11.4.jar - C:\W
- junit-4.12.jar - C:\Users\W\MYC
- hamcrest-core-1.3.jar - C:\Us
- spring-test-5.0.7.RELEASE.jar -
- spring-context-5.0.7.RELEASE
- spring-aop-5.0.7.RELEASE.jar -
- spring-expression-5.0.7.RELEA
- spring-webmvc-5.0.7.RELEASE
- spring-web-5.0.7.RELEASE.jar
- aspectjrt-1.6.10.jar - C:\Users\

```
</dependency>  
  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
  
<!-- https://mvnrepository.com/artifact/  
<dependency>  
    <groupId>org.bgee.log4jdbc-log4j2</g  
    <artifactId>log4jdbc-log4j2-jdbc4</a  
    <version>1.16</version>  
</dependency>
```

현재 마이바티스의 결과물은 실제 sql로그와는 형태가 약간 다릅니다.
이제 sql창에서 나오는 로그와 유사한 구문을 콘솔에 찍기 위해
추가설정을 하겠습니다.

Mvnrepository에서 log4jdbc-log4j2-jdbc4를 검색해 1.16버전을
Pom.xml에 입력합니다.



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

로그 설정을 추가하기 위해 src/main/resource에
Log4jdbc.log4j2.properties 파일을 생성하고

화면과 같이 문구를 입력합니다.

```
<!--  
<property name="driverClassName"  
    value="com.mysql.cj.jdbc.Driver"></property>  
<property name="jdbcUrl"  
    value="jdbc:mysql://localhost:3306/mysql?serverTimezone=UTC"></property> -->  
<property name="driverClassName"  
    value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></property>  
<property name="jdbcUrl"  
    value="jdbc:Log4jdbc:mysql://localhost:3306/mysql?serverTimezone=UTC"></property>  
<property name="username" value="root"></property>
```

그리고 root-context.xml의 hikariConfig를 설정하는 곳에서

DriverClassName과 jdbcUrl을 log4jdbc에 맞게 변경합니다.

```
INFO : jdbc.resultset - 1. ResultSet.isClosed
INFO : jdbc.resultsettable -
|-----|
|now()   |
|-----|
|2021-03-11 01:55:39|
|-----|

INFO : jdbc.resultset - 1. ResultSet.next() r
INFO : jdbc.resultset - 1. ResultSet.close()
INFO : jdbc.audit - 1. Connection.getMetaData
INFO : jdbc.audit - 1. PreparedStatement.getM
```

바꾼 뒤 다시 TimeMapperTests.java를 테스트 실행하면 콘솔창에
이제 sql 커맨드라인에서 봤을법한 로그가 나옵니다.