스프링(Spring) 개발

간단한 게시판을 하나 만들어 보면서 스프링의 MVC 구조와 DB, 트랜잭션, 파일 업로드, jQuery등을 살펴봅니다.

(9) Mybatis (마이바티스) 연동하기

1. Mybatis

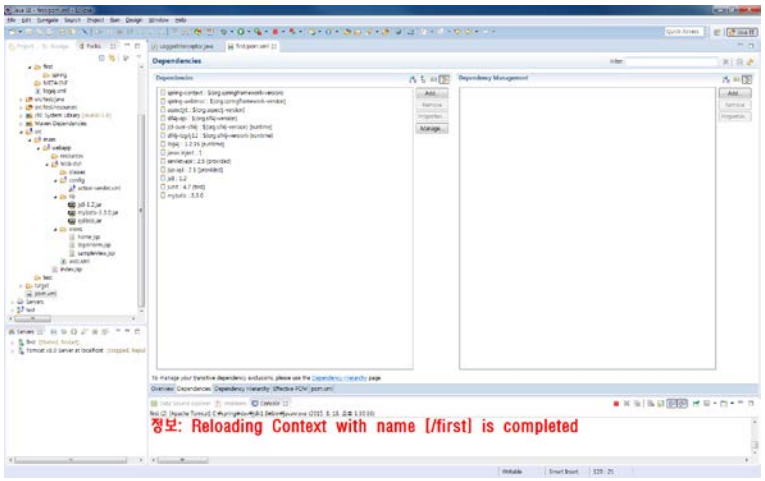
1. MyBatis(마이바티스)란?

MyBatis는 개발자가 지정한 SQL, 저장 프로시저, 트랜잭션 등 몇가지 고급 매핑을 지원하는 퍼시스턴스 프레임워크이다. MyBatis는 데이터베이스 레코드의 원시타입과 Map 인터페이스 등에 대해 자바 POJO를 설정하고 매핑하기 위해 XML과 애노테이션을 사용할 수 있다.
기존에 JDBC를 이용하여 프로그래밍을 하는 방식에 비해서 MyBatis는 개발자의 부담을 굉장히 많이 덜어주고, 생산성 향상에도 도움이 된다.
기존에 JDBC를 이용하여 프로그래밍을 하는 방식은 프로그램 소스 안에 SQL문을 작성하는 방식이었다. 커넥션을 연결하고, SQL 구문을 전송해서 결과셋 받아오고, 그걸 rs.next()등을 이용하여 한행씩 컬럼별 데이터를 DTO 객체에 저장하고 컬렉션에 추가하는 방식이었다.
따라서 SQL의 변경 등이 발생할 경우, 프로그램(java 파일)을 수정하기 때문에 그 유연성이 좋지 못했는데, MyBatis에서는 SQL구문을 따로 mapper.xml 파일에 작성하기 때문에, SQL의 변환이 자유롭고, 가독성이 좋다는 장점이 있다.

2. MyBatis 라이브러리

스프링에서 MyBatis를 사용하려면 라이브러리가 필요하다.
우리는 이미 Maven을 이용하여 라이브러리를 관리하기 때문에, 라이브러리의 추가가 굉장히 쉽다.

pom.xml을 열고, 아래의 '**Dependencies' 탭**을 클릭하고, 왼쪽 'Dependenices 영역의 'Add...' 버튼을 눌러 추가할 라이브러리에 대한 정보를 입력한다




```
1       <dependency>
2           <groupId>org.mybatis</groupId>
3           <artifactId>mybatis</artifactId>
4           <version>3.3.0</version>
5       </dependency>
6
7       <dependency>
8           <groupId>org.mybatis</groupId>
9           <artifactId>mybatis-spring</artifactId>
10          <version>1.2.2</version>
11      </dependency>
12
13      <dependency>
14          <groupId>org.springframework</groupId>
15          <artifactId>spring-jdbc</artifactId>
16          <version>${org.springframework-version}</version>
17      </dependency>
18
19      <dependency>
20          <groupId>commons-dbcp</groupId>
21          <artifactId>commons-dbcp</artifactId>
22          <version>1.4</version>
23      </dependency>
```

여기까지가 공통으로 들어가는 라이브러리이고, 사용하는 DB에 따라서 추가할 라이브러리가 약간 다르다.

1) MySql을 사용할 경우

```
1          <dependency>
2              <groupId>mysql</groupId>
3              <artifactId>mysql-connector-java</artifactId>
4              <version>5.1.31</version>
5          </dependency>
```

이것만 추가하면 된다.

2) Oracle의 경우

오라클의 경우 ojdbc.jar 파일을 검색할 경우, 다음과 같은 dependency가 나오는데, 실제로는 다운받을 수가 없다.

```
1    <dependency>
2            <groupId>ojdbc</groupId>
3            <artifactId>ojdbc</artifac
4             <version>14</version>
5    </dependency>
```

따라서 다음의 dependency로 바꿔준다.

```
1    <dependency>
2        <groupId>com.oracle</groupId>
3        <artifactId>ojdbc14</artifactId>
4        <version>10.2.0.4.0</version>
5    </dependency>
```

이렇게 작성하여도 아직은 에러가 나온다.
따라서 상단의 <properties> 다음에 아래의 코드를 추가한다. 키보드로 입력하면 자동 팁이 나타날 것이므로 선택하여 입력되게 처리한다.

```
1   <repositories>
2       <repository>
3           <id>mvn2</id>
4           <url>http://repo1.maven.org/maven2/</url>
5           <releases>
6               <enabled>true</enabled>
7           </releases>
8           <snapshots>
```

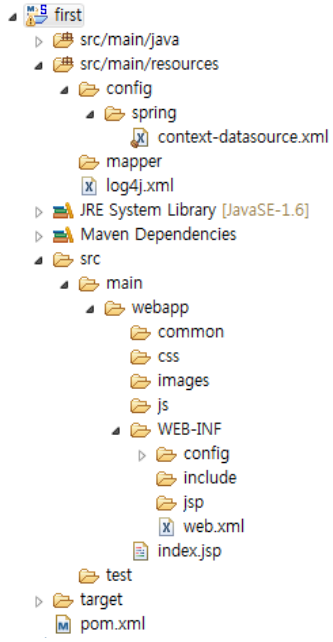
```
9               <enabled>true</enabled>
10          </snapshots>
11      </repository>
12
13      <repository>
14          <id>oracle</id>
15          <name>ORACLE JDBC Repository</name>
16          <url>http://mesir.googlecode.com/svn/trunk/mavenrepo</url>
17      </repository>
18  </repositories>
```

ojdbc를 다운받을 수 있는 Repository를 수동으로 등록한 것이다.
Overview 탭에서 상단의 에러 표시가 없는지 확인한다.

3. MyBatis와 DB(데이터베이스) 연결 설정

마이바티스와 데이터베이스를 연결하는 방법을 살펴보자.

1) src/main/resource 폴더 밑에 이전에 추가했던 first/spring 폴더는 지우고, config/spring 폴더를 만든다.
2) spring 폴더에서 -> 마우스 우클릭 -> New -> Spring Bean Configuration File 선택 -> 파일명 : context-datasource.xml 파일을 만든다.



위와 같은 구조로 만들어 준다.

3) web.xml에서 위의 설정 파일을 읽을 수 있도록 요소를 추가한다.
web.xml의 하단에 다음의 코드가 있어야 한다.

```
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath*:config/spring/context-*.xml</param-value>
    </context-param>
```

이는 최초 서버가 시작될 때, 해당 위치에 있는 context 파일을 모조리 읽어들이는 것을 뜻한다. xml 태그에서 알 수 있듯이, contextConfigLocation을 설정하고, 그 위치는 위에서 만들었던 config / spring 폴더에 있는 context-로 시작하는 모든 xml 파일을 의미한다.
앞으로 스프링 관련 여러가지 설정파일이 있기 때문에, 위와 같이 설정파일을 모두 읽어올 수 있도록 하였다.

4) MyBatis 연결 설정을 하자.

4-1) MySQL 의 경우

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                        http://www.springframework.org/schema/jdbc  http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd">

    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://주소/스키마"/>
        <property name="username" value="아이디"/>
        <property name="password" value="비밀번호"/>
    </bean>
</beans>
```

달리 어려운 설정은 없을 것이라고 생각한다.
url에서 MySQL이 설치된 서버의 주소와 사용할 DB 스키마를 적어주면 된다. (개인PC에서 개발할 경우 localhost/스키마 또는 127.0.0.1/스키마 로 적어주면 된다.)

4-2) Oracle의 경우

context-datasource.xml 파일 아래의 'beans' 탭 선택 -> 'New Bean...' 누름 -> 각 속성에 값 입력한다. Class 항목은 오른쪽 'Browse...' 를 눌러 클래스명을 검색해서 선택한다.
추가된 dataSource 위에서 마우스 우클릭 -> Insert <property> element 선택함 -> 화면 오른쪽 Element Details 뷰에서 각 속성에 값 입력함 -> 입력이 완료되면 저장한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                        http://www.springframework.org/schema/jdbc  http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd">

    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"/>
        <property name="url" value="jdbc:oracle:thin:@localhost:1521:XE"/>
        <property name="username" value="아이디"/>
        <property name="password" value="비밀번호"/>
    </bean>
</beans>
```

저장 후에 파일 윗부분에 에러 표시가 없는지 확인한다.

4. MyBatis와 Spring의 연결

1) src/main/resource 폴더에 mapper 폴더를 생성한다.
2) src/main/resource 의 config / spring 폴더에 'Spring Bean Configuration File' 을 이용하여 context-mapper.xml 파일을 생성한다.
3) context-mapper.xml 파일에 다음의 내용을 작성한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://www.springframework.org/schema/context
```

```
8 http://www.springframework.org/schema/context/spring-context.xsd">
9
10    <bean id="sqlSession" class="org.mybatis.spring.SqlSessionFactoryBean">
11        <property name="dataSource" ref="dataSource" />
12        <property name="mapperLocations" value="classpath:/mapper/**/*_SQL.xml" />
13    </bean>
14
15    <bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
16        <constructor-arg index="0" ref="sqlSession"/>
      </bean>
  </beans>
```

소스 코드를 살펴보면,.
11번째 줄의 property의 name과 ref가 dataSource로 정의되어있다.
이 두가지는 같은 것을 의미하지 않는다. name은 위에서 등록한 sqlSession 빈(bean)에서 사용할 이름이 dataSource이고, ref의 dataSource는 우리가 context-datasource.xml에서 정의한 빈(bean)을 참조하는 것을 의미한다.
12번째 줄의 mapperLocations는 앞으로 우리가 작성할 SQL문이 위치할 장소이다. 여기서 classpath:/mapper/**/*_SQL.xml 이라는 정의를 살펴보자.
앞에서 web.xml에서 spring context 설정파일을 읽어오기 위해서 classpath*:config/spring/context-*.xml 라고 정의했던 것을 기억하자.
앞으로 우리는 다양한 SQL 파일을 만들 것인데, 그것을 일일이 등록해서 사용할 수는 없다. (원래는 xml 파일에 XML도 bean으로 설정해야한다.) 그렇지만 다양한 사람들이 작업하는 프로젝트의 특성상, 그것을 일일이 등록할 수 없고, 할 수 있더라도 귀찮은 일이다. 따라서 서버가 시작될 때 자동으로 SQL이 정의되어 있는 XML 파일도 읽어 오도록 하는 것이 필요하다.
따라서 SQL이 위치할 mapper 폴더를 잡아주고, 그 안에 모든 폴더를 의미하는 **를 붙여준 후, 마지막으로 _SQL로 끝나는 모든 xml 파일을 읽어오도록 설정한다.
여기서 중간에 ** 를 붙인 이유는, 유연한 폴더구조의 변경을 위해서 이렇게 작성했다. 예를 들어 mapper / first / *_SQL.xml 와 mapper / first / second / *_SQL.xml 은 다른 경로이다. (mapper 폴더 밑에 first, second 라는 이름의 폴더가 있는 경우를 의미)
우리는 앞으로 게시판에 관련된 쿼리만 작성하겠지만, 실제 프로젝트에서는 굉장히 많은 패키지가 생성되기 때문에 폴더가 2, 3단계로 구분된다. 그것을 유연성있게 대처하도록 해 줬다.
마지막으로 15번째 줄의 sqlSessionTemplate은 마이바티스 스프링 연동모듈의 핵심이다. SQLSessionTemplate은 SqlSession을 구현하고, 코드에서 SqlSessoin을 대체하는 역할을 한다.

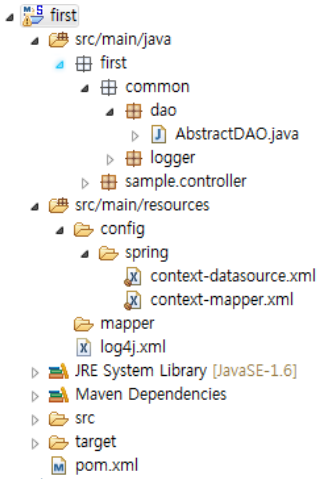
5. DAO 작성

마지막으로 실제 소스에서 위에서 선언한 SqlSessionTemplate을 사용할 DAO를 만들어주자.
DAO는 Data Access Object의 약자로, Data Access Object를 Factory 패턴화 시키는 것으로, 비즈니스 로직을 모듈화 하는 방법이다. Factory 패턴이 무엇인지 잘 모르면, 디자인 패턴에 대해서 공

부하면 좋다.

1) src/main/java 폴더의 first > common 패키지 밑에 dao 패키지를 생성한다.

2) dao 패키지 안에 AbstractDAO.java를 생성한다.



위와 같은 구조를 가지게 된다.

3) AbstractDAO에 다음 소스를 작성한다.

```
1	public class AbstractDAO {
2		protected Log log = LogFactory.getLog(AbstractDAO.class);
3
4		@Autowired
5		private SqlSessionTemplate sqlSession;
6
7		protected void printQueryId(String queryId) {
8			if(log.isDebugEnabled()){
9				log.debug("₩t QueryId  ₩t:  " + queryId);
10			}
11		}
12
13		public Object insert(String queryId, Object params){
14			printQueryId(queryId);
15			return sqlSession.insert(queryId, params);
16		}
17
18		public Object update(String queryId, Object params){
19			printQueryId(queryId);
```

```
20                  return sqlSession.update(queryId, params);
21              }
22
23              public Object delete(String queryId, Object params){
24                  printQueryId(queryId);
25                  return sqlSession.delete(queryId, params);
26              }
27
28              public Object selectOne(String queryId){
29                  printQueryId(queryId);
30                  return sqlSession.selectOne(queryId);
31              }
32
33              public Object selectOne(String queryId, Object params){
34                  printQueryId(queryId);
35                  return sqlSession.selectOne(queryId, params);
36              }
37
38              @SuppressWarnings("rawtypes")
39              public List selectList(String queryId){
40                  printQueryId(queryId);
41                  return sqlSession.selectList(queryId);
42              }
43
44              @SuppressWarnings("rawtypes")
45              public List selectList(String queryId, Object params){
46                  printQueryId(queryId);
47                  return sqlSession.selectList(queryId,params);
48              }
49          }
```

간단히 살펴보자.
우리가 앞에서 SqlSessionTemplate을 설정하였고, 이는 SqlSession을 대체한다고 이야기 했었다.
5번째 줄에 SqlSessionTemplate을 선언하고 여기에 Autowired 어노테이션(Annotation)을 통해서 xml에 선언했던 의존관계를 자동으로 주입하도록 하였다.
쿼리는 sqlSession.메서드를 호출하면 되는데, 여기서는 앞으로 개발할때, 좀 더 보기편하게 로그를 남기기위해서 AbstractDAO를 만들어서 insert, delete, update, select 메서드를 재정의 하였다.
실제 개발에서는 각 비즈니스 로직을 담당할 DAO를 생성하여 AbstractDAO를 상속받도록 할 계획이다.

여기까지 작성하고 서버를 실행시켜서 에러가 나지 않으면, 설정에 문제가 없는것으로 판단할 수 있다.
여기까지 MyBatis의 설정이 완료되었습니다.

first1.zip