Spring & MyBatis

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy/Spring5

MyBatis ?

- Persistence
  - **데이터의 지속성**을 의미
  - Application이 종료하고 다시 실행하더라도 이전에 저장한 데이터를 다시 불러올 수 있는 기술을 말한다.
- Framework
  - Library가 개발에 필요한 도구들을 단순히 나열한 반면,
  - Framework은 동작에 필요한 구조를 어느 정도 완성해 놓은 **반제품** 형태의 도구
- Persistence Framework
  - 데이터의 저장, 조회, 변경, 삭제를 다루는 **클래스 및 설정 파일들의 집합**
  - JDBC 프로그래밍의 복잡함이나 번거로움 없이 간단한 작업만으로 Database와 연동되는 시스템을 빠르게, 안정적으로 개발 가능

MyBatis ? (Cont.)

- Persistence Framework 종류
  - SQL Mapper
  - Object-Relational Mapper
- SQL Mapper
  - SQL 문장을 통해 Database의 데이터 다루는 방법
  - iBATIS / MyBatis
- Object-Relational Mapper(ORM)
  - Java 객체를 통해 간접적으로 Database의 데이터를 다루는 방법
  - Hibernate / JPA / TopLink

MyBatis ? (Cont.)

- http://www.mybatis.org/mybatis-3
- Java Object와 SQL 문 사이의 자동 Mapping 기능을 지원하는 Semi-ORM Framework.
- SQL을 별도의 File로 분리해서 관리하게 해주며, 객체-SQL 사이의 Parameter Mapping 작업을 자동으로 해주기 때문에 많은 인기를 얻고 있는 기술
- 단순하고 반복적인 JDBC 코드를 Capsul化하여 DB Programming을 간결하게 만듦
- Hibernate나 JPA(Java Persistence API)처럼 새로운 DB Programming Paradigm을 익혀야 하는 부담 없이, 개발자가 익숙한 SQL을 그대로 이용하면서 JDBC Code 작성의 불편함도 제거해주고, Domain 객체나 VO 객체를 중심으로 개발이 가능하다는 장점.

MyBatis 특징

- 쉬운 접근성과 Code의 간결함
  - 가장 간단한 Persistence Framework.
  - XML 형태로 서술된 JDBC Code라고 생각해도 될 만큼 JDBC의 모든 기능을 MyBatis가 대부분 제공.
  - 복잡한 JDBC Code를 걷어내며 깔끔한 Source Code 유지.
  - 수동적인 Parameter 설정과 Query 결과에 대한 Mapping 구문 제거.
- SQL 문과 Programming Code의 분리
  - SQL에 변경이 있을 때마다 Java Code를 수정하거나 Compile 하지 않아도 된다.
  - SQL 작성과 관리 또는 검토를 DBA와 같은 개발자가 아닌 다른 사람에게 맡길 수도.
- 다양한 programming 언어로 구현가능
  - Java, C#, .NET, Ruby

MyBatis 작동 흐름

1. 데이터 처리를 위해 DAO는 MyBatis에서 제공하는 객체의 Method를 호출한다.
2. MyBatis는 SQL이 저장된 Mapper 파일에서 데이터 처리에 필요한 SQL문을 찾는다.
3. MyBatis는 Mapper 파일에서 찾은 SQL을 서버에 보내고자 JDBC Driver를 사용
4. JDBC Driver는 SQL문을 Database Server로 보낸다.
5. MyBatis는 Select 문의 실행 결과를 값 객체에 담아서 반환하고, Insert, Update, Delete문인 경우 입력, 변경, 삭제된 레코드의 개수를 반환한다.

DAO에서 MyBatis를 사용하는 시나리오

1. DAO는 **SqlSessionFactory**에게 SQL을 실행할 객체를 요구한다.
2. **SqlSessionFactory**는 **SqlSession** 객체를 생성하여 반환한다.
3. DAO는 **SqlSession** 객체에게 SQL 실행을 요청한다.
4. **SqlSession** 객체는 SQL이 저장된 **Mapper 파일(XML)**에서 SQL을 찾는다.
5. **SqlSession**은 JDBC Driver를 통해 Database Server에게 질의를 실행한다.
6. **SqlSession**은 Database Server로부터 가져온 질의 결과를 객체로 생성하여 반환한다.
7. DAO는 사용이 끝난 **SqlSession**을 닫는다.

MyBatis와 MyBatis-Spring을 사용한 DB Access Architecture

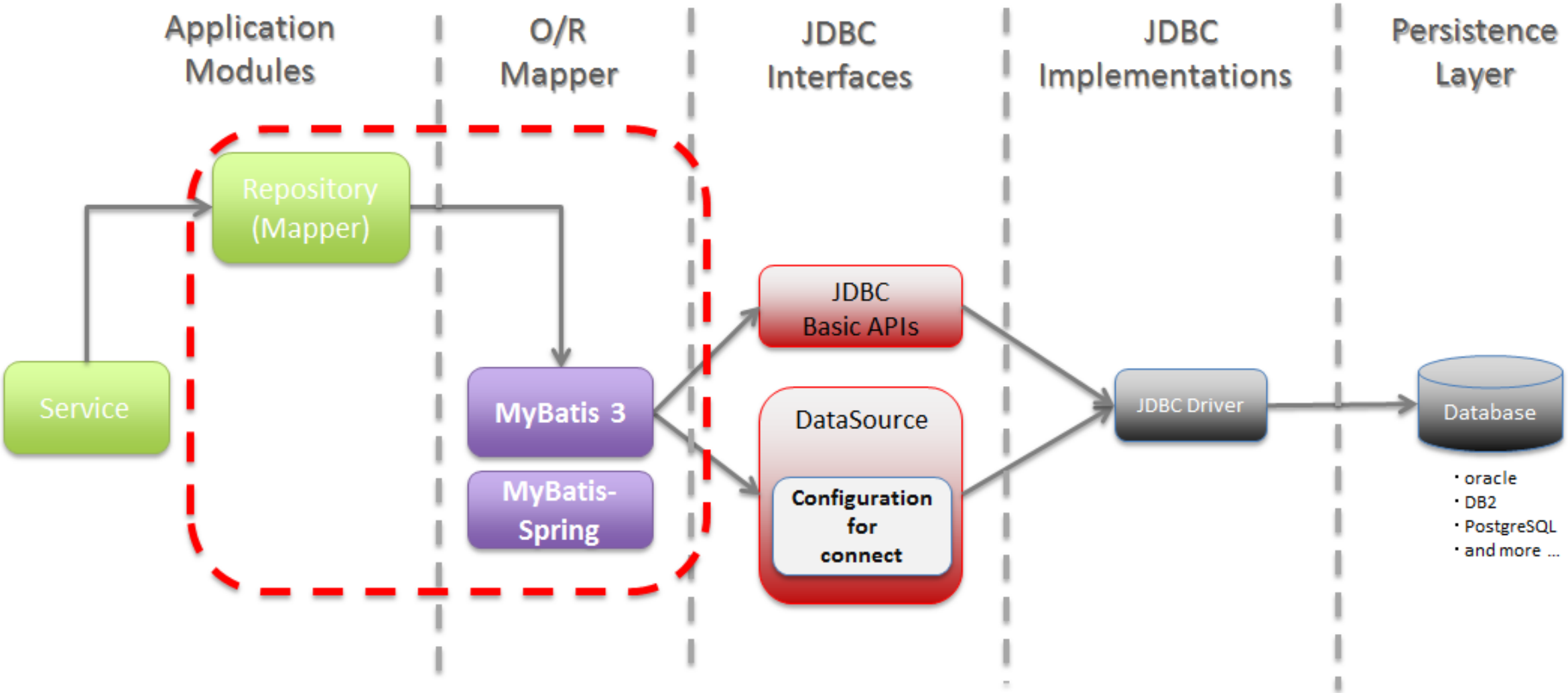

Image from http://terasolunaorg.github.io/guideline/5.0.0.RELEASE/en/ArchitectureInDetail/DataAccessMyBatis3.html

MyBatis를 사용하는 데이터 액세스 계층

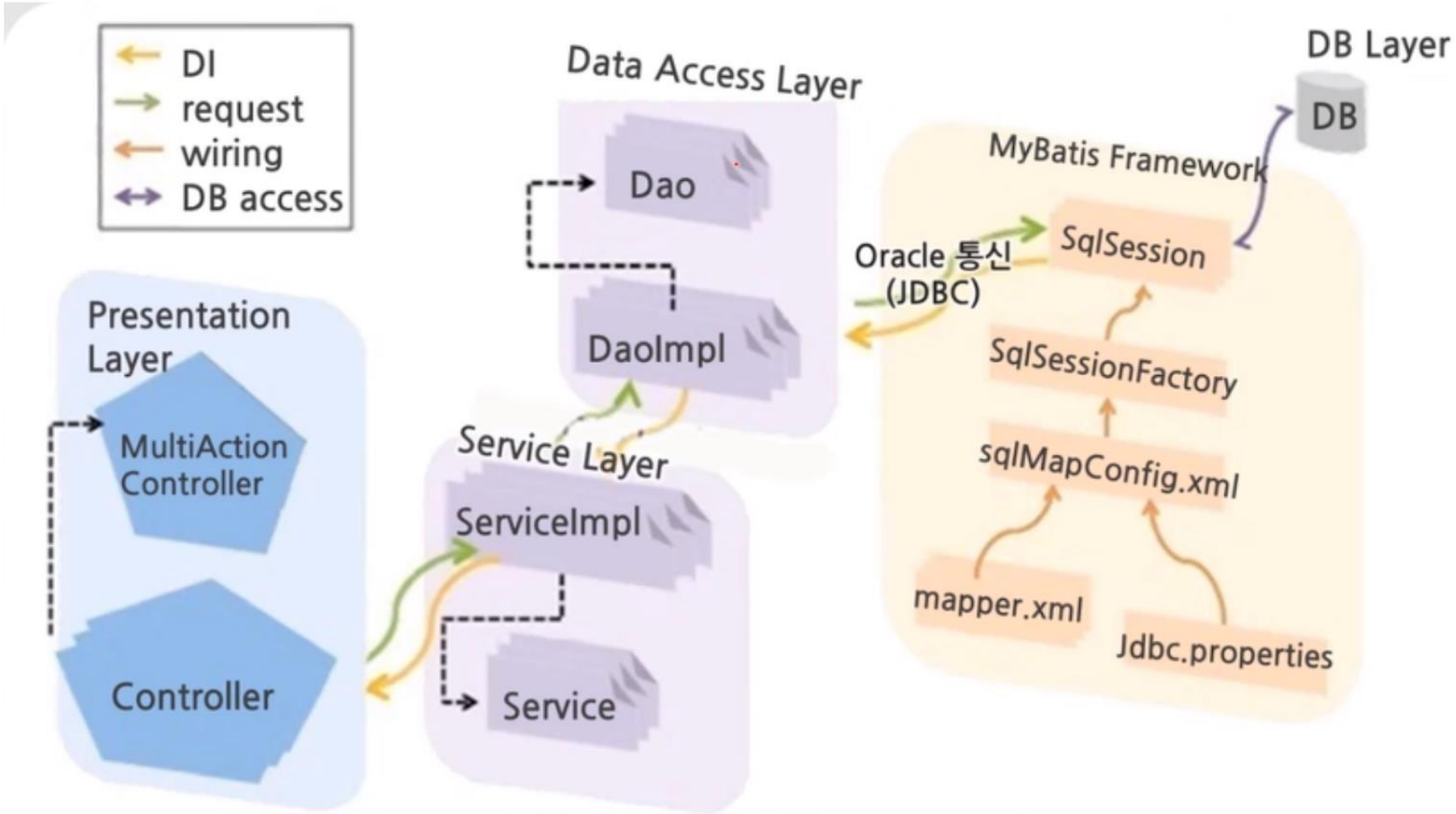

Image from https://velog.io/@cil05265/Mybatis%EB%9E%80-%EC%84%A4%EB%AA%85-%EC%82%AC%EC%A7%84-%EC%B2%A8%EB%B6%80

MyBatis의 주요 Components

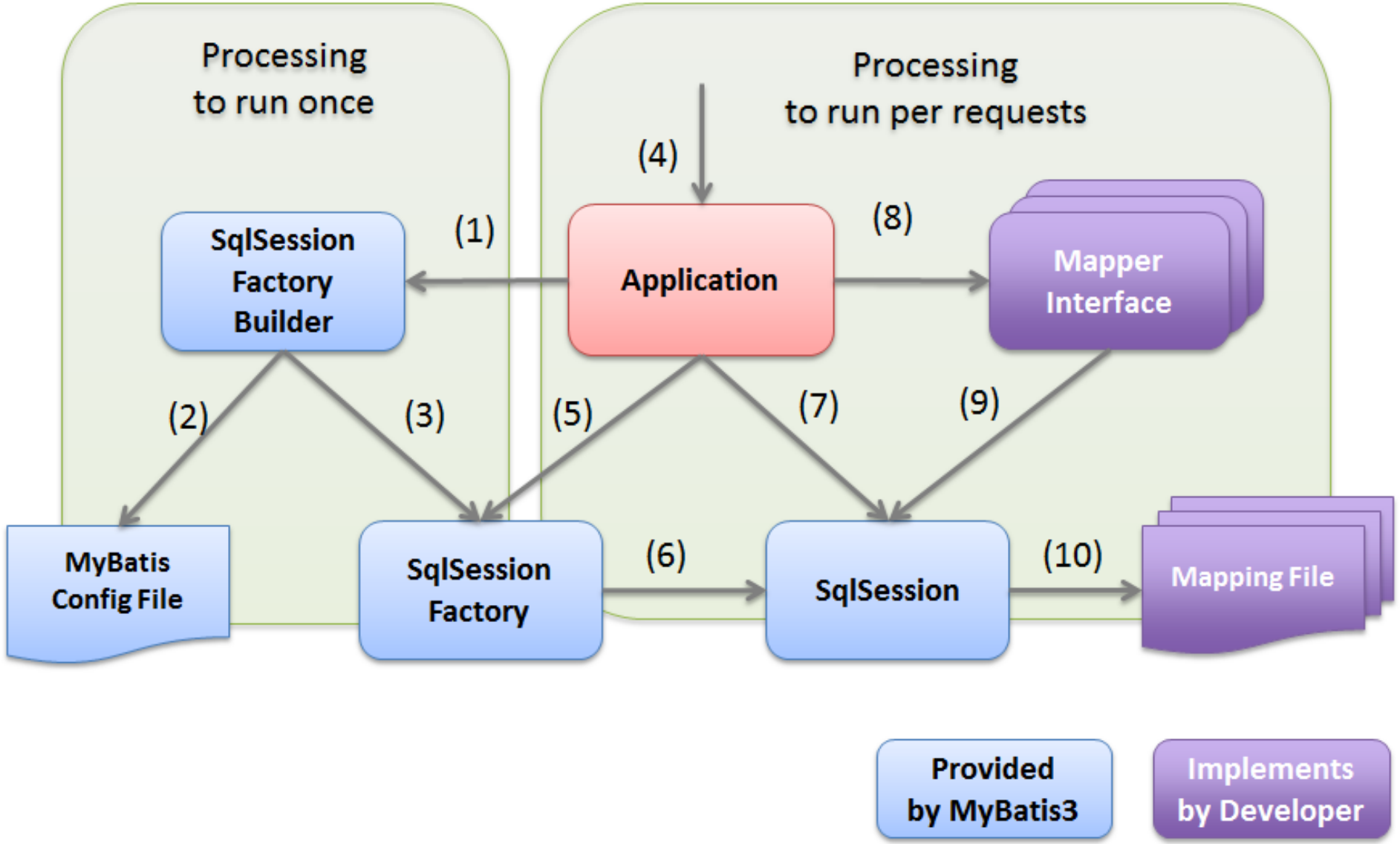

Image from http://terasolunaorg.github.io/guideline/5.0.0.RELEASE/en/ArchitectureInDetail/DataAccessMyBatis3.html

흐름

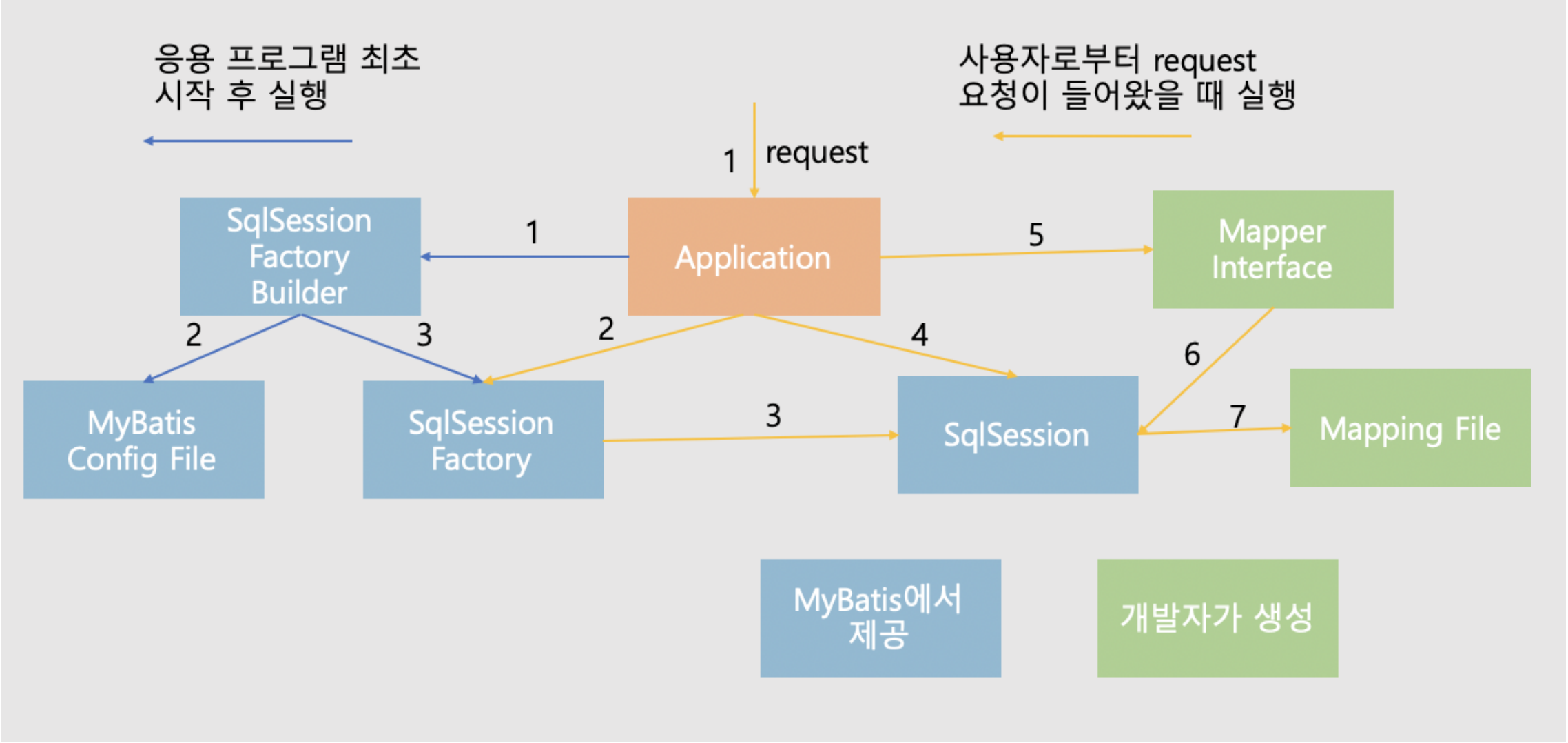

Image from https://ksshlee.github.io/spring/springmybatis/

MyBatis의 주요 Components (Cont.)

- MyBatis 설정 File(i.e. SqlMapconfig.xml)
  - Database의 접속 주소 정보나 Mapping File의 경로 등의 고정된 환경정보를 설정
- **SqlSessionFactoryBuilder**
  - MyBatis 설정 File을 바탕으로 **SqlSessionFactory**를 생성
- **SqlSessionFactory**
  - **SqlSession**을 생성
- **SqlSession**
  - 핵심적인 역할을 하는 Class로서 SQL 실행이나 Transaction 관리를 실행
  - **SqlSession** Object는 Thread-Safe하지 않으므로 Thread마다 필요에 따라 생성
- Mapping File(i.e. Users.xml)
  - SQL문과 O-R Mapping을 설정

MyBatis-Spring의 주요 컴포넌트

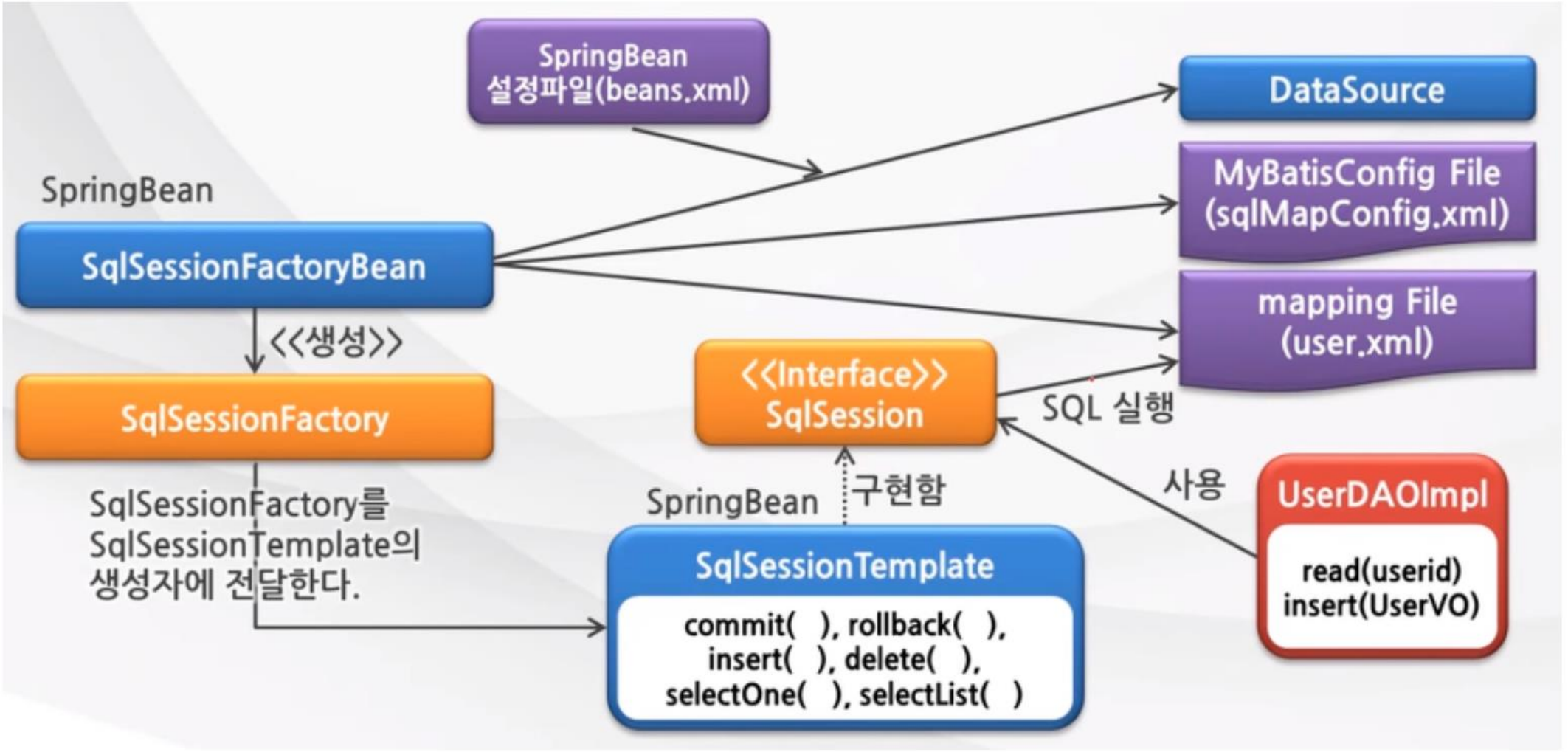

Image from https://velog.io/@cil05265/Mybatis%EB%9E%80-%EC%84%A4%EB%AA%85-%EC%82%AC%EC%A7%84-%EC%B2%A8%EB%B6%80

MyBatis-Spring의 주요 Component의 역할

- MyBatis 설정 File(i.e. sqlMapConfig.xml)
  - VO 객체의 정보를 설정
- **SqlSessionFactoryBean**
  - MyBatis 설정 File을 바탕으로 **SqlSessionFactory**를 생성
  - Spring Bean으로 등록해야 함.
- **SqlSessionTemplate**
  - 핵심적인 역할을 하는 Class로서 SQL 실행이나 Transaction 관리를 실행한다.
  - **SqlSession** Interface를 구현하며, Thread-Safe하다.
  - Spring Bean으로 등록해야 함.
- Mapping File(i.e. mybatis-mapper.xml)
  - SQL문과 OR Mapping을 설정
- Spring Bean 설정 File(i.e. beans.xml)
  - **SqlSessionFactoryBean**을 Bean 등록할 때 **DataSource** 정보와 MyBatis Config file정보, Mapping file의 정보를 함께 설정한다.
  - **SqlSessionTemplate**을 Bean으로 등록한다.

설치하기

- Maven 사용시 pom.xml에 다음과 같은 설정 추가

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.7</version>
</dependency>
```

- Spring 에서 사용하려면

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>2.0.6</version>
</dependency>
```

XML에서 SqlSessionFactory 빌드하기

- 모든 MyBatis Application은 **SqlSessionFactory** Instance를 사용한다.
- **SqlSessionFactory** Instance는 **SqlSessionFactoryBuilder**를 사용하여 만들 수 있다.
- **SqlSessionFactoryBuilder**는 XML설정파일에서 **SqlSessionFactory** Instance를 빌드할 수 있다.

```
String resource = "org/mybatis/example/mybatis-config.xml";
Reader reader = Resources.getResourceAsReader(resource);
SqlSessionFactory sqlSessionFactory =
                        new SqlSessionFactoryBuilder().build(reader);
```

DAO에서 SqlSessionFactory 사용

- **SqlSessionFactory**는 SQL을 실행할 때 사용할 도구를 만들어줌
- DAO class에서 **SqlSessionFactory**를 저장할 Instance 변수와 Setter를 선언

```
SqlSessionFactory sqlSessionFactory;

public void setSqlSessionFactory(SqlSessionFactory sqlSessionFactory) {
      this.sqlSessionFactory = sqlSessionFactory;
}
```

SqlSession 사용

- **SqlSession**은 SQL을 실행하는 도구
- 이 객체는 직접 생성할 수 없고, **SqlSessionFactory**를 통해서만 얻을 수 있음.

```
SqlSession sqlSession = sqlSessionFactory.openSession();

try {
    return sqlSession.selectList("Employee.selectAll");
    //List<E> selectList(String sqlId)
    //sqlId = SQL 맵퍼의 네임스페이스 이름 + 맵퍼파일에 있는 SQL문 ID
    //List<E> selectList(String sqlId, Object parameter)
    //SELECT문을 실행하는데 값이 필요하다면 위와 같이 두번째 매개변수로 값을 전달
} finally {
    sqlSession.close();
}
```

SqlSession의 주요 Method

- **selectList()**
  - SELECT문 실행. 값 객체(Value Object) List 반환
- **selectOne()**
  - SELECT문 실행. 하나의 값 객체 반환
- **insert()**
  - INSERT문 실행. 반환값은 입력한 데이터의 개수
- **Update()**
  - UPDATE문 실행. 반환값은 입력한 데이터의 개수
- **Delete()**
  - DELETE문 실행. 반환값은 입력한 데이터의 개수

SqlSession의 Method와 Parameter 값 전달

- DAO 에서

```
sqlSession.insert("Employee.insert", newEmployee);
```

- Mapper 파일에서

```
<insert id="insert" parameterType="newEmployee">
    insert into mytable(col1, col2, col3[,…])
    values (#{val1}, #{val2}, #{val3}[,…])
</insert>
```

#{property_name}
Instance의 이름이 아니라 Getter/Setter를 말함, property의 이름은 Getter/Setter에서 추출
예)**#{title}**은 객체의 **getTitle()**의 반환값

MyBatis에서 TCL 다루기

- commit() / rollback()

```
int count = sqlSession.insert("Employee.insert", newEmployee);
sqlSession.commit();
return count;
```

- 자동 커밋(Auto-commit)
  - INSERT, UPDATE, DELETE 을 실행할 때 자동으로 커밋하려면 SqlSession객체를 생성할 때 true 전달.

```
SqlSession sqlSession = sqlSessionFactory.openSession(true);
```

  - 자동 커밋으로 설정해 놓고 쓰면 편리하지만 Transaction을 다룰 수 없음.

SQL Mapper

Bbs.xml

```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="com.example.vo.BbsVO">

    <resultMap type="bbsVo" id="BbsResultMap">
        <result property="idx" javaType="java.lang.Integer"
              column="idx" jdbcType="INTEGER"/>
        <result property="username" javaType="java.lang.String"
              column="username" jdbcType="VARCHAR"/>
        <result property="email" javaType="java.lang.String"
              column="email" jdbcType="VARCHAR"/>
        <result property="title" javaType="java.lang.String"
              column="title" jdbcType="VARCHAR"/>
        <result property="contents" javaType="java.lang.String"
              column="contents" jdbcType="VARCHAR"/>
        <result property="readnum" javaType="java.lang.Integer"
              column="readnum" jdbcType="INTEGER"/>
        <result property="writeday" javaType="java.util.Date"
              column="writeday" jdbcType="VARCHAR"/>
    </resultMap>

    <parameterMap type="bbsVo" id="selectAllParameterMap">
        <parameter property="results" javaType="ResultSet" jdbcType="CURSOR"
          mode="OUT" resultMap="BbsResultMap"/>
    </parameterMap>

    <select id="selectAll" parameterMap="selectAllParameterMap" statementType="CALLABLE">
        { call sp_select_all(?) }
    </select>
```

MyBatis Configuration File

mybatis-config.xml

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

<configuration>
    <properties resource="dbinfo.properties"/>
    <typeAliases>
        <typeAlias type="com.javasoft.libs.models.StudentVO" alias="StudentVO"/>
    </typeAliases>

    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="${db.driver}"/>
                <property name="url" value="${db.url}"/>
                <property name="username" value="${db.username}"/>
                <property name="password" value="${db.password}"/>
            </dataSource>
        </environment>
    </environments>
    <mappers>
        <mapper resource="com/javasoft/libs/models/Student.xml"/>
    </mappers>
</configuration>
```

Spring Bean Configuration File

applicationContext.xml

```
11          <property name="driverClass" value="${db.driver}" />
12          <property name="url" value="${db.url}" />
13          <property name="username" value="${db.username}" />
14          <property name="password" value="${db.password}" />
15       </bean>
16
17       <context:component-scan base-package="com.javasoft" />
18
19       <!-- Spring-Mybatis Setting -->
20       <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
21          <property name="dataSource" ref="dataSource" />
22          <property name="configLocation" value="classpath:mybatis-config.xml" />
23          <property name="mapperLocations">
24             <list>
25                <value>classpath:mapper.xml</value>
26             </list>
27          </property>
28       </bean>
29
30       <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
31          <constructor-arg index="0" ref="sqlSessionFactory" />
32       </bean>
33    </beans>
```

Task 1. MySQL World Database의 City Table 가져오기

Task 2. SungjukMgmt Project with MyBatis