

Practice #2

Annotation을 사용한 Spring IoC 컨테이너 내용 정리

Spring의 IoC 컨테이너 저장 조건 (가장 많이 쓰이는 조건 4가지+@)

1. **@Component**가 붙은 클래스 자체
(@Configuration이 붙은 설정 클래스가 @ComponentScan으로 스캔을 먼저 해줘야함)
2. **@Controller**가 붙은 클래스
(해당 클래스안에 있는 메소드에 @RequestMapping을 붙이고 URL을 주면
그 URL을 가지고 해당 메소드를 실행할 수 있고, 이는 @Component의 경우도 유효하다.)
3. **@Configuration**이 붙은 클래스 안의 **@Bean**이 붙은 메소드가 리턴하는 객체
4. 클래스 안에 다른 클래스가 인스턴스 멤버로서 포함되어 사용될 때 @AutoWired를 붙여서
객체를 주입해준다.
(해당 인스턴스 멤버의 클래스 파일에도 당연히 @Component가 붙어야 함)

1. Mybatis 연습

- Mybatis를 사용하여 SqlSessionFactory로부터 SqlSession을 가져와서 매핑된
Sql문을 호출하여 회원정보 CRUD를 수행한다.

필요한 세팅 - mariadb-java-client / mybatis

build.gradle의 dependencies에 mariadb-java-client와 mybatis를 추가하고 빌드
(mvnrepository.com에서 MariaDB Java Client 와 Mybatis를 검색)

프로세스

- **필요한 클래스 및 설정 파일들**

- Practice01.java
- Member.java
- MemberMapper.xml
- mybatis-config.xml
- jdbc.properties

1. Member.java 생성

- 새롭게 practice2 패키지를 만들고, 편의를 위해 eomcs.lms의 Member.java를
복사하여 practice2.domain에 붙여넣기 한다.

2. Mapper 설정 전 테스트

- Practice01.java를 만들어서 Member의 등록, 조회, 수정, 삭제 시에 호출 할 가상의
메소드의 메소드명 / 파라미터 / 리턴타입을 고민해 본다.

3. mybatis-config 설정

- src/main/resources에 com.eomcs.lms와 같은 패키지를 만든다.
- conf라는 하위 패키지를 만들고 그 안에 mybatis-config.xml 파일을 생성한다.

- <https://mybatis.org/mybatis-3/> 에 접속하여 한국어 페이지 선택 후 [시작하기]에 들어간다.
- 아래의 TEST용 코드와 설정용 코드를 각각 Practice01.java와 mybatis-config.xml에 복사한다.

The screenshot shows the MyBatis 3.0 Korean documentation page. The main heading is "XML에서 SqlSessionFactory 빌드하기". Below the heading, there is a paragraph explaining that all MyBatis applications use SqlSessionFactory instances, which are created using SqlSessionFactoryBuilder. The text states that the builder can be used to create SqlSessionFactory instances from XML configuration files.

Below the text, there is a code block labeled "TEST용 코드" (Test Code) which shows the Java code for creating a SqlSessionFactory:

```
String resource = "org/mybatis/example/mybatis-config.xml";
InputStream inputStream = Resources.getResourceAsStream(resource);
SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
```

Below the code block, there is a paragraph explaining that the XML configuration file specifies the MyBatis settings, including the transaction manager and the data source. The text states that the configuration file is used to create the SqlSessionFactory.

Below the paragraph, there is a code block labeled "mybatis-config.xml 코드" (mybatis-config.xml Code) which shows the XML configuration file:

```
<?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>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC">
        <property name="driver" value="${driver}" />
        <property name="url" value="${url}" />
        <property name="username" value="${username}" />
        <property name="password" value="${password}" />
      </transactionManager>
      <dataSource type="POOLED">
        <property name="driver" value="${driver}" />
        <property name="url" value="${url}" />
        <property name="username" value="${username}" />
        <property name="password" value="${password}" />
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="org/mybatis/example/BlogMapper.xml" />
  </mappers>
</configuration>
```

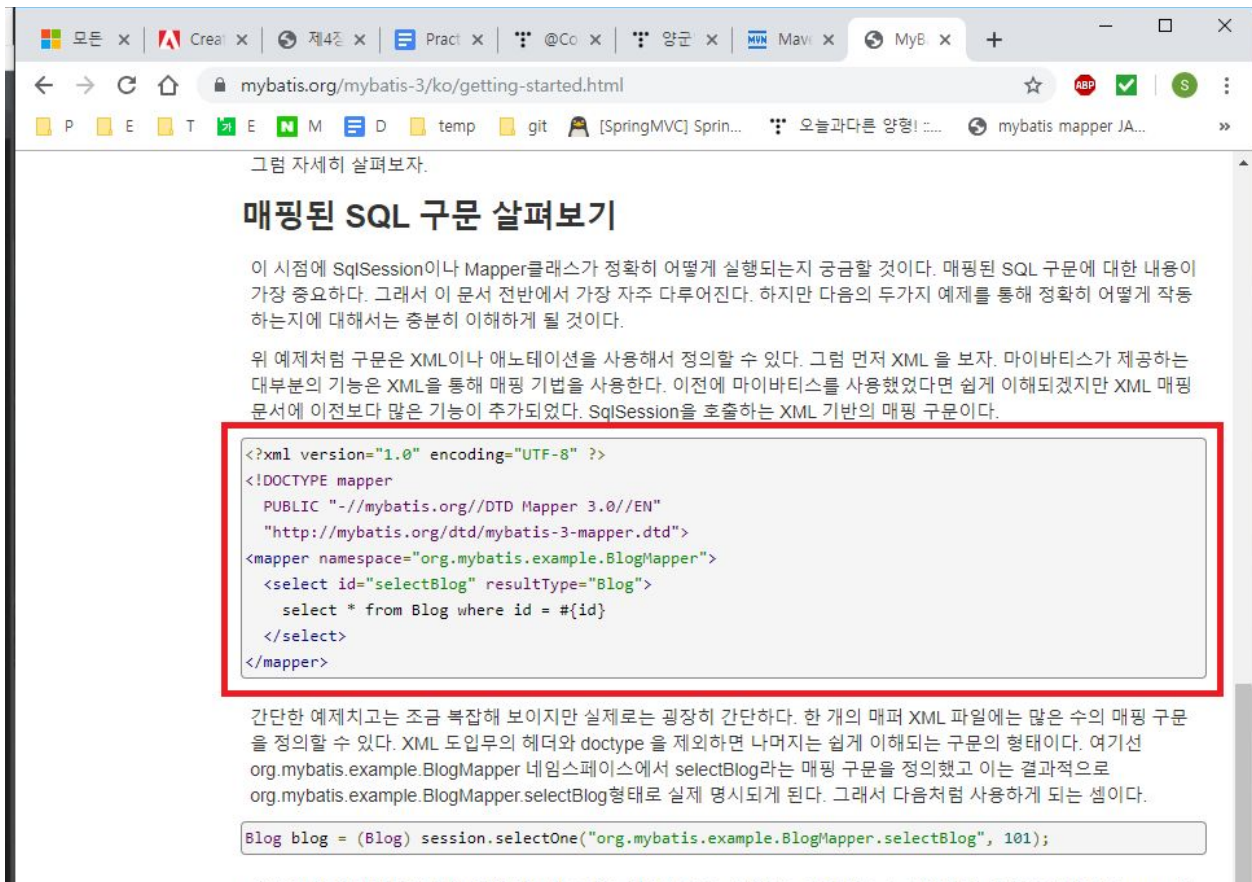
Below the code block, there is a paragraph explaining that the XML configuration file specifies the MyBatis settings, including the transaction manager and the data source. The text states that the configuration file is used to create the SqlSessionFactory.

At the bottom of the page, there is a heading "XML 을 사용하지 않고 SqlSessionFactory 빌드하기".

- datasource 접속 정보를 유지보수를 위해 따로 분리한다; jdbc.properties를 mybatis-config.xml과 같은 장소에 생성하여 배치하고 name=value의 순서대로 jdbc.driver / jdbc.url / jdbc.username / jdbc.password로 이름을 정해주고 값을 준다.
- mybatis-config.xml로 돌아와서 위의 4개의 항목에 해당하는 부분의 value 이름을 jdbc.properties에서 설정한 것과 똑같이 바꿔준다.

4. Mapper 설정

- mapper라는 하위 패키지를 추가로 만들고 그 안에 MemberMapper.xml 파일을 생성하고 [시작하기]에서 아래의 템플릿 코드를 복사해온다.



그럼 자세히 살펴보자.

매핑된 SQL 구문 살펴보기

이 시점에 SqlSession이나 Mapper클래스가 정확히 어떻게 실행되는지 궁금할 것이다. 매핑된 SQL 구문에 대한 내용이 가장 중요하다. 그래서 이 문서 전반에서 가장 자주 다루어진다. 하지만 다음의 두가지 예제를 통해 정확히 어떻게 작동하는지에 대해서는 충분히 이해하게 될 것이다.

위 예제처럼 구문은 XML이나 애노테이션을 사용해서 정의할 수 있다. 그럼 먼저 XML을 보자. 마이바티스가 제공하는 대부분의 기능은 XML을 통해 매핑 기법을 사용한다. 이전에 마이바티스를 사용했었다면 쉽게 이해되었지만 XML 매핑 문서에 이전보다 많은 기능이 추가되었다. SqlSession을 호출하는 XML 기반의 매핑 구문이다.

```
<?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="org.mybatis.example.BlogMapper">
  <select id="selectBlog" resultType="Blog">
    select * from Blog where id = #{id}
  </select>
</mapper>
```

간단한 예제치고는 조금 복잡해 보이지만 실제로는 굉장히 간단하다. 한 개의 매핑 XML 파일에는 많은 수의 매핑 구문을 정의할 수 있다. XML 도입부의 헤더와 doctype을 제외하면 나머지는 쉽게 이해되는 구문의 형태이다. 여기서 org.mybatis.example.BlogMapper 네임스페이스에서 selectBlog라는 매핑 구문을 정의했고 이는 결과적으로 org.mybatis.example.BlogMapper.selectBlog 형태로 실제 명시되게 된다. 그래서 다음처럼 사용하게 되는 셈이다.

```
Blog blog = (Blog) session.selectOne("org.mybatis.example.BlogMapper.selectBlog", 101);
```

- <mapper namespace>는 임의로 아무거나 정해도 상관없다. 기본적으로 mapping된 SQL 문들은 sqlSession의 메소드에 [namespace.id] 형식의 파라미터에 주어 호출한다. 예) sqlSession.selectList(ohora.findAll)
- [Mapper XML 파일] 페이지를 참고하여 insert / select / update / delete 엘리먼트에 SQL문을 매핑하고 id를 지정한다. (Note #1 참고)
- Mapper 파일의 수정이 끝났으면 mybatis-config.xml에서 <mapper resource>에 매퍼의 경로를 포함한 풀네임을 써준다.
- 편의상 mybatis-config.xml에 typeAliases 엘리먼트도 추가하여 package name으로 domain 패키지의 경로를 지정하여 사용한다.

Keyword

SqlSession, properties resource, typeAliases, namespace, resultMap

Tip

- 경로와 대소문자 및 namespace와 mapper의 id가 틀려서 오류가 나는 경우에 주의. 특히 패키지 경로를 입력할 때 구분자가 "/"인지 "."인지 꼼꼼하게 체크한다.

2. Mybatis-Spring 연동 및 DAO를 통한 매핑

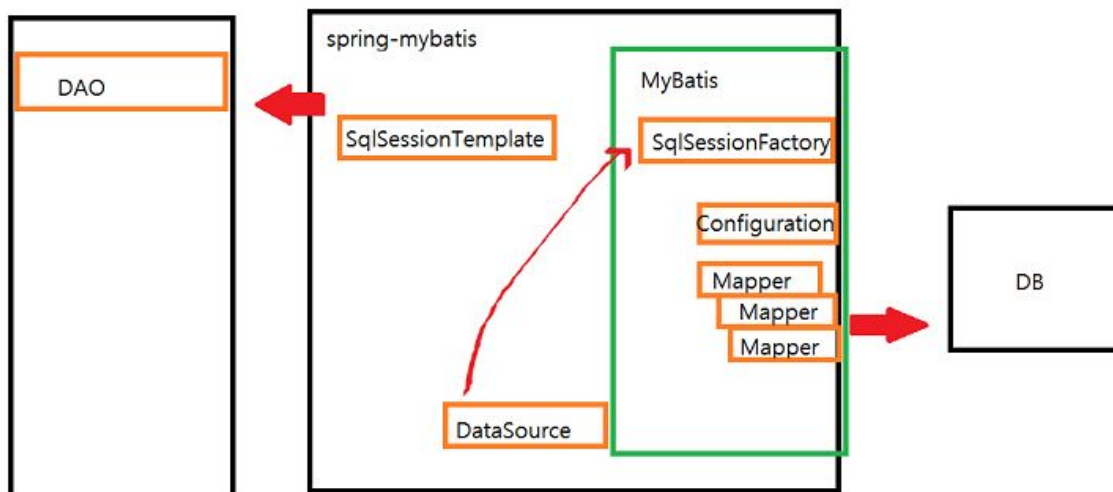
- Spring의 애노테이션을 사용한 설정방법 준비 및 객체 만드는 추가 연습
- Mybatis와 Spring의 연동으로 SqlSession을 직접 다루지 않고 애노테이션을 붙여 Spring이 자동으로 객체를 만들게 하고, Mybatis가 DB와 DAO를 매핑시켜 간단하게 회원정보 CRUD 기능구현

필요한 세팅 - *spring-jdbc / mybatis-spring*

build.gradle의 dependencies에 spring-jdbc와 mybatis-spring을 추가하고 빌드
(mvnrepository.com에서 Spring JDBC 와 Mybatis Spring를 검색)

Spring-Mybatis의 DB의 데이터에 접근하는 구조

- 개발자는 DB의 데이터를 다루는 규칙을 **DAO**를 통해 정의한다.
- Spring-Mybatis는 **Datasource**를 담당하는데, **DataSource**는 DB 서버와의 기본적인 연결 / DB 커넥션 풀 / 트랜잭션처리 등의 기능이 들어있는 객체이다.
- **SqlSessionFactory**는 SqlSession을 만드는 객체이며 SqlSession은 sql문을 호출하는 객체이다.
- Spring을 사용하여 이러한 객체들을 자동으로 준비하여 IoC 컨테이너에 저장하는 것으로 일일이 객체를 만드는 수고를 덜어준다.



프로세스

- 앞서 나온 그림에서 설명한 대로 핵심이 되는 DataSource 객체, SqlSessionFactory 객체, DAO 객체, Mapper, TransactionManager, 설정 파일들을 준비하고 Spring의 애노테이션 기법을 사용하여 객체를 자동으로 생성하여 Mybatis가 사용하게 한다.

- **필요한 클래스 및 설정 파일들**

- Practice02.java
- Member.java (기존 파일에 @Component 추가)
- MemberDao.java
- MemberMapper2.xml
- DatabaseConfig.java
- MybatisConfig.java
- jdbc.properties

1. practice2.dao 패키지에 **MemberDao** 인터페이스를 만들고, 간단한 CRUD 메소드를 정의한다.
2. MemberMapper.xml을 복사해 **MemberMapper2.xml**을 만들어준다.
3. Mybatis를 사용하여 직접 SqlSession으로 sql 문을 실행 할 때는 [namespace.id]의 형태로 자신이 직접 namespace를 지정해서 매핑된 SQL문을 실행할 수 있었지만 spring이 자동으로 만들어주는 객체들을 사용할 때는 <mapper namespace>를 경로까지 들어간 DAO 풀네임으로 지정해준다.
4. config 패키지를 만들고 DB 설정용 파일인 **DatabaseConfig.java**를 생성한다.

이 config 클래스에는 DataSource와 TransactionManager를 생성하여 리턴하는 메소드를 두개 만들고 @Bean을 붙여 IoC 컨테이너가 객체를 만들게 한다.

@Configuration : 이 클래스에 붙여서 이 파일이 설정파일임을 명시하고 IoC 컨테이너가 이 클래스를 참조하여 객체를 만들게 한다.

@EnableTransactionManagement : 이 클래스에 붙여서 이 클래스가 @Transactional 애노테이션이 붙은 메소드를 처리할 객체로 설정한다.

@PropertySource : 이 클래스에 붙여서 properties 파일을 참조하게 한다.

@PropertySource를 사용한다면 **@Value**도 사용하여 String 레퍼런스에 값을 넘겨준다.

@Bean : IoC 컨테이너가 찾은 **@Configuration**가 붙은 클래스 안에 무언가 객체를 만들어 리턴하는 메소드가 있고 그 메소드에 **@Bean**이 붙어 있다면 IoC 컨테이너는 이것도 객체로 만들어서 저장한다.

-
5. config 패키지에 Mapper 정보가 담긴 SqlSessionFactory 객체를 설정하는 **MybatisConfig.java**를 생성한다.

@Configuration : 위와 동일 (이 애노테이션이 없으면 **@Bean**이 붙어도 객체가 만들어지지 않는다.)

@MapperScan : 지정해준 경로에 있는 DAO를 스캔하여 Mapper와 매칭시킨다.

@ComponentScan : 지정해준 경로에서 **@Component**가 붙은 클래스를 모두 찾아 객체로 만든다.

6. domain 패키지에 있는 Member 클래스에도 **@Component**를 붙여준다.
7. Practice02에서 IoC 컨테이너를 만들고, (이 때 config 패키지의 경로를 지정하여 그곳에 있는 모든 Config 파일들을 참조하게 한다.) IoC 컨테이너에서 MemberDao 빈을 꺼내와서 CRUD 기능을 구현한다.

Keyword

MemberDao, MemberMapper, SqlSessionFactory, DataSource, PlatformTransactionManager

Tip

- <http://mybatis.org/spring/index.html> 를 최대한 참고한다. (한국어 번역페이지도 내부에 있으므로 이해하는데 도움이 된다) 하지만 모든 내용이 다 나와있는 것은 아니므로 인터넷 검색도 해보고 충분히 생각해보고 난 뒤에 답안을 볼 것.