JDBC와 MyBatis를 이용한 DB연결

개요

- JDBC란?
  : http://inlab.cu.ac.kr/~wlgns12www/java/JDBC.pdf

- MyBatis란?
  : http://inlab.cu.ac.kr/~wlgns12www/java/MyBatis.pdf

※ 전 4챕터(Simple Server 2 and Test)에서 진행하였던 프로젝트를 기반으로 본 챕터를 진행하도록 한다. 본 실습에서는 MySQL관련 명령어는 제공하지 않는다.

※ 실습을 진행하기 전 MySQL의 Table은 아래와 같이 구성해 두었다.

테이블 이름	**Test**	
필드 이름	**name**	**age**
자료형	**varchar(15)**	**int**
저장된 데이터	"park"	25
	"lee"	26
	"jang"	23

실습1) install JDBC & MyBatis & Lombok

1. build.gradle의 dependencies에 아래의 의존성 코드 추가 후 build Refresh 수행

```
dependencies {
    ...(중략)

    // mybatis
    compile 'org.mybatis.spring.boot:mybatis-spring-boot-starter:1.3.2'
    compile 'org.springframework.boot:spring-boot-starter-jdbc'
    compile 'mysql:mysql-connector-java'

    // lombok
    compileOnly 'org.projectlombok:lombok:1.18.12'
    annotationProcessor 'org.projectlombok:lombok:1.18.12'

    ...(중략)
}
```

2. build가 정상적으로 수행되었다면 실습2)로 넘어가도록 한다.

실습2) 스프링 부트와 JDBC & MyBatis 연동을 위한 외부 설정 파일 셋팅

1. 스프링 부트 외부설정 파일 생성
   : src/main/resources 오른쪽클릭 -> New -> File 왼쪽클릭

2. 스프링 부트의 외부설정 파일 이름 설정
   : 반드시 **"application.properties"**로 작성한다. (해당 이름으로 하여야 스프링에서 참조한다.)

3. application.properties에 아래 박스와 같이 입력

```
server.port=8080

# jdbc init
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://[host_ip]:[mysql_port]/[DB_name]?serverTimezone=UTC
spring.datasource.username=[db user id]
spring.datasource.password=[user password]

# mybatis
mybatis.mapper-locations=mybatis/mapper/**/*.xml
```

4. main/java/test_application/Application 클래스의 main을 실행시켜서 정상적으로 실행되는지 확인
   : jdbc와 spring boot가 정상적으로 연동되었는지 확인하기 위함

5. [4.]까지 진행하였다면 스프링부트와 jdbc, mybatis의 연동이 완료되었다.
   [4.]까지 완료하였다면 실습3)으로 넘어가도록 한다.

실습3) mybatis 설정 파일 셋팅

1. mybatis의 설정파일을 생성하기 위한 디렉터리(폴더) 생성
   : src/main/resources 오른쪽 클릭 -> New -> Directory 왼쪽클릭

2. Directory 이름은 **"mybatis"**로 설정

3. 생성된 디렉터리에 새로운 설정파일 생성
   : src/main/resources/mybatis 오른쪽 클릭 -> New -> File 왼쪽클릭

4. File 이름은 **"mybatis-config.xml"**로 설정

5. mybatis-config.xml에 아래의 박스와 같이 입력

```
<?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>
    <mappers>
        <mapper resource="myMapper.xml"/>
    </mappers>
</configuration>
```

6. [5.]까지 진행하였다면 실습4)로 넘어가도록 한다.

실습4) Person 클래스 재작성

※ 기존 챕터4(Simple Server 2 and Test)에서는 Person 클래스에 lombok을 적용하지 않았으며, MyBatis랑 연동하기 위한 Getter와 Setter가 부족하였다.
본 실습4)에서는 MyBatis와 Person 클래스를 연동하기 위해 Person 클래스에 lombok 어노테이션을 적용한다.

1. src/main/java/test_application/test_model/Person 클래스의 내용을 아래와 같이 재작성

```
// 기존내용
package test_application.test_model;

public class Person {
    private final String name;
    private final int age;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public String getName(){
        return this.name;
    }

    public int getAge(){
        return this.age;
    }
}
```

```
// 변경된 내용
package test_application.test_model;

import lombok.*;

@Getter // getName(), getAge()
@Setter // setName(), setAge()
@AllArgsConstructor // Person(name, age)
public class Person {
    private String name;
    private int age;
}
```

2. [1.]까지 진행하였다면 실습5)로 넘어가도록 한다.

실습5) MyBatis 인터페이스 작성

1. 인터페이스를 작성하기 위한 새로운 패키지 생성
   : src/main/java/test_application 오른쪽클릭 -> New -> package 왼쪽클릭

2. 패키지 이름은 **"test_db.test_mapper"**로 작성

3. 생성된 패키지에 새로운 클래스 생성
   : src/main/java/test_application/test_db/test_mapper 오른쪽클릭 -> New -> Java Class 왼쪽클릭

4. 클래스 이름은 **"myMapper"**로 작성

5. myMapper 클래스의 내용은 아래 박스와 같이 입력

```
package test_application.test_db.test_mapper;

import org.apache.ibatis.annotations.Mapper;
import test_application.test_model.Person;

import java.util.List;

@Mapper
public interface myMapper {
    List<Person> findAllPerson();
    Person findOne(String name);
}
```

6. 이제 인터페이스인 myMapper의 모든 메소드(findAllPerson(), findOne())를 구현하여야 한다.
   하지만, 직접 Java Class로 구현하는 것이 아니라 MyBatis *.xml파일로 구현할 것이다.

7. [5.]까지 정상적으로 진행하였다면 실습6)로 넘어가도록 한다.

실습6) MyBatis 인터페이스 구현

1. 인터페이스 구현을 위한 xml 파일을 저장시킬 디렉터리 생성
   : src/main/resources/mybatis 오른쪽클릭 -> New -> Directory 왼쪽클릭

2. Directory 이름은 **"mapper"**로 작성
   : application.properties파일에 mybatis의 경로를 "mybatis/mapper/**/*.xml"로 설정하였다.
   : 그러므로 실습3)에서 생성하였던 mybatis 디렉터리 위에 mapper를 생성한 것이다.

3. 인터페이스 구현을 위한 xml 파일생성
   : src/main/resources/mybaits/mapper 오른쪽클릭 -> New -> File 왼쪽클릭

4. File 이름은 **"myMapper.xml**"로 작성

: 실습5)에서 생성하였던 myMapper 인터페이스를 구현하기 위하여 같은 이름으로 xml파일을 생성
: 이제 myMapper 인터페이스의 두 메소드(findAllPerson(), findOne())를 xml파일로 구현한다.

5. myMapper.xml의 내용을 아래 박스와 같이 입력

```
<?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">

<!-- namespace : Mapper 인터페이스 경로 -->
<mapper namespace="test_application.test_db.test_mapper.myMapper">
<!-- type : 클래스 파일 경로 -->
    <resultMap id="person" type="test_application.test_model.Person">
        <!-- column : 디비속성이름 / property : 클래스속성이름 -->
        <result column="name" property="name" />
        <result column="age" property="age" />
    </resultMap>

    <select id="findAllPerson" resultMap="person">
        SELECT * FROM test;
    </select>

    <select id="findOne" resultMap="person">
        SELECT * FROM test
        <where>
            <if test="value!=null and value!=''">
                AND name=#{name}
            </if>
        </where>
    </select>
</mapper>
```

: select태그의 id속성을 통해 myMapper 인터페이스의 구현할 메소드를 설정하는 모습을 볼 수 있다.
: findAllPerson() 쿼리내용 => test 테이블에 존재하는 모든 튜플을 끌어온다.
: findOne() 쿼리내용 => test 테이블에서 특정 name인 튜플 하나만 끌어온다.

6. [5.]까지 완료하였으면 이제 실제로 테스트를 진행해 볼 것이다.
이번에는 JUnit을 사용하지 않고 직접 Server를 올려서 웹 브라우저에서 테스트할 것이다.

7. 테스트를 진행하기전 src/main/java/test_application/test_web/Controller에 API를 추가해야 함으로 다음 실습7)로 넘어가도록 하자.

실습7) Controller에 API 추가

1. 기존 Controller에 다음 박스의 내용을 추가

```
package test_application.test_web;
...(중략)
```

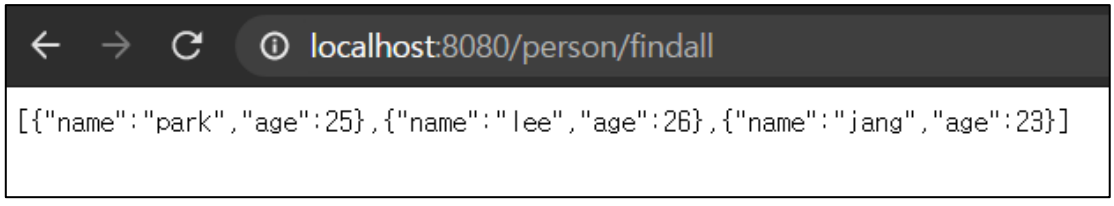
```
@RestController
public class Controller {

    @GetMapping("/")
   ...(중략)

    @GetMapping("/person")
    ...(중략)

    @Autowired // Bean을 자동 주입
    myMapper myMapper;

    @GetMapping("/person/findall")
    public List<Person> responsePersonAll(){
        List<Person> person = myMapper.findAllPerson();

        return person;
    }

    @GetMapping("/person/findone/{name}")
    public Person responsePersonOne
            (@PathVariable("name") String name){
        Person person = myMapper.findOne(name);

        return person;
    }
}
```

: RESTful하지 못한 URI이지만 지금은 실습임으로 잠깐 참아주도록 한다...ㅎ

2. Controller에 [1.]과 같이 추가하였다면 Application에서 서버를 실행시킨 뒤 테스트를 진행하도록 하자. 실습7)에서는 테스트 결과를 보여준다.

실습7) 테스트 결과 확인

1. HTTP GET /person/findall



2. HTTP GET /person/findone/{name}

참고문헌

- 이현아 (2018). https://medium.com/cashwalk/springboot-mybatis-gradle-mysql-7090359d5427