DB 연동하기

1) 설정 파일에서 임시 코드 제거

> DB 설정을 제외하는 문법을 제거 한다.

```
@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})
```

2) Mapper 설정

설정 클래스 추가

> DB용 설정 클래스를 따로 만든다.(설정용 패키지를 하나 만듦)

```
package com.marobiana.ex.config;

import javax.sql.DataSource;

import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.mybatis.spring.annotation.MapperScan;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.Resource;
import org.springframework.core.io.support.PathMatchingResourcePatternResolver;

@Configuration
@MapperScan(basePackages="com.marobiana.*")  // interface(mapper)가 있는 패키지 경로
public class DatabaseConfig {

    @Bean
    public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {
        SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean();
        sessionFactory.setDataSource(dataSource);

        Resource[] res = new PathMatchingResourcePatternResolver().getResources("classpath:mappers/*Mapper.xml");
        sessionFactory.setMapperLocations(res);

        return sessionFactory.getObject();
    }
}
```

DB 정보 프로퍼티 추가

SpringBoot에서 `.properties` 파일 또는 `.yml` 파일을 읽어서 설정 정보들을 읽는다.
yml로 변화되는 추세이므로 `application.properties` 를 **삭제**하고 `application.yml` 로 추가한다.

application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/test?serverTimezone=UTC&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=비밀번호
```

application.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/test?serverTimezone=UTC&characterEncoding=UTF-8
    username: root
    password: 비밀번호
```

쿼리 수행 로그 세팅

수행된 쿼리문을 Console에서 볼 수 있도록 로그를 찍는 세팅을 한다.
SpringBoot web starter에 기본으로 `logback` 이라는 로깅 라이브러리가 세팅되어 있으며 설정을 추가하면 된다.

`src/main/resources` 에 설정 파일명을 `logback-spring.xml` 로 설정하면 logback 설정 파일을 읽는다.

`logback-spring.xml`

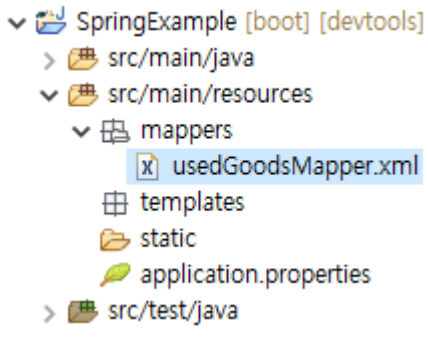
```
<?xml version="1.0" encoding="UTF-8"?>
<configuration debug="true">

    <!-- Appenders -->
    <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <Pattern>%d %5p [%c] %m%n</Pattern>
        </encoder>
    </appender>

    <!-- Logger -->
    <logger name="com.marobiana" level="DEBUG" appender-ref="console" />

    <!-- Root Logger -->
    <root level="info">
        <appender-ref ref="console" />
    </root>
</configuration>
```

위의 내용의 코드를 추가하면 수행된 쿼리문이 로그에 찍힌다.

쿼리 Mapper

쿼리가 들어갈 mapper 위치

- `src/main/resources` 밑에 `mappers` package를 만들고 `*Mapper.xml` 로 지어 사용한다.



쿼리 예제

```
<?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.marondal.lecture.dao.ExampleListDAO">
    <select id="selectExampleList" parameterType="int" resultType="com.marondal.lecture.model.ExampleList">
        SELECT
            `id`
            , `lectureId`
            , `lectureListId`
            , `order`
            , `subject`
            , `userId`
            , `author`
            , `filePath`
            , `createdAt`
            , `updatedAt`
        FROM
            `exampleList`
        WHERE
            `id` = #{id}
    </select>

 </mapper>
```

2. JPA 설정

Lombok 라이브러리 추가하기

build.gradle에 lombok 의존성 라이브러리 추가

```
compileOnly 'org.projectlombok:lombok'
annotationProcessor 'org.projectlombok:lombok'
```

반드시 `build gradle Refresh` 한다.

Lombok 라이브러리 STS와 연결

- lombok jar 다운로드

Windows

※ `lombok.jar` 의 위치가 한글 폴더로 되어있으면 STS가 안 열리므로 주의!!!

- lombok.jar를 수행시킨 후 `Specify location` 버튼으로 `sts.exe` 가 있는 위치를 로드 후 install/update

MAC

- lombok.jar를 수행시킨 후 창이 뜨면 `Specify location` 버튼 클릭
- SpringToolSuite4 > Contents > Ecplise > SpringToolSuite4.ini 선택 후 install/update

ini 파일에 추가 되어 있어야 함

```
-javaagent:D:\shinboram\5_spring\ex\ex_sts\lombok.jar
```

STS에서 Project > Clean 으로 빌드를 다시 한다.

동작 테스트

DTO 객체에서 `@Data` 어노테이션을 붙였을 때 에러가 나지 않고 getter/setter를 사용할 수 있으면 된다.

JPA 설정

build.gradle에 JPA 의존성 라이브러리 추가

```
implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
```

반드시 `build gradle Refresh` 한다.

쿼리 결과 보기(바인딩 된 파라미터) - p6spy 라이브러리 의존성 추가

```
implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.5.7'
```

반드시 `build gradle Refresh` 한다.

프로퍼티 설정

`application.properties` 에 설정을 추가한다.

key	설명
`spring.jpa.show-sql`	`true` : JPA 쿼리문 확인 가능
`spring.jpa.hibernate.ddl-auto`	`create` :기존 테이블을 삭제 후 새로 생성 `create-drop` :기존 테이블을 삭제 후 새로 생성 + 종료 후 다시 drop (drop + create + drop) `update` :테이블과 매핑 정보 비교 후 변경사항만 수정(만약 기존 테이블 없으면 create) `validate` :차이가 있으면 경고를 날린 후 수행하지 않음 `none` :테이블 자동 생성기능 사용 안함(있는 테이블로 사용)
`spring.jpa.properties.hibernate.format_sql`	`true` : jpa 구현체인 하이버네이트 동적쿼리 로그를 예쁘게 출력

application.properties

```
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.format_sql=true
```

application.yml

```
spring:
  jpa:
    show-sql: true
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: true
```

테이블 컬럼명을 CamelCase로 지정하는 설정

application.properties

```
spring.jpa.hibernate.naming.implicit-strategy=org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyJpaImpl
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

application.yml

```
spring:
  jpa:
    hibernate:
      naming:
        implicit-strategy: org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyJpaImpl
        physical-strategy: org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

Copyright © marondal 2021