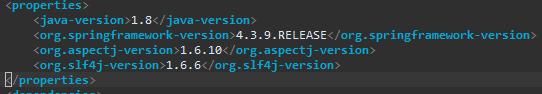
**2. 스프링 PostgreSQL 데이터 베이스 연동.**

**스프링 기본설정**

**처음에는 pom.xml을 설정 해준다. 자바와 스프링 버전 변경.**

****

**project Facets 자바 버전 변경.**

**2. pom.xml 추가 항목.(webmvc 복 붙후 artifactId 변경.)**

**<!-- test jdbc 설정-->**

**<dependency>**

**<groupId>org.springframework</groupId>**

**<artifactId>spring-test</artifactId>**

**<version>${org.springframework-version}</version>**

**</dependency>**

**<dependency>**

**<groupId>org.springframework</groupId>**

**<artifactId>spring-jdbc</artifactId>**

**<version>${org.springframework-version}</version>**

**</dependency>**

**<!-- mybatis 설정 -->**

**<dependency>**

**<groupId>org.mybatis</groupId>**

**<artifactId>mybatis</artifactId>**

**<version>3.4.1</version>**

**</dependency>**

**<dependency>**

**<groupId>org.mybatis</groupId>**

**<artifactId>mybatis-spring</artifactId>**

**<version>1.3.0</version>**

**</dependency>**

**<!-- PostgreSQL 9.4 -->**

**<dependency>**

**<groupId>org.postgresql</groupId>**

**<artifactId>postgresql</artifactId>**

**<version>9.4.1209.jre6</version>**

**</dependency>**

**log4JDBC JUnit을 이용한 DB 테스트**

**<!-- log4jdbc 설정 -->**

**<dependency>**

**<groupId>org.bgee.log4jdbc-log4j2</groupId>**

**<artifactId>log4jdbc-log4j2-jdbc4</artifactId>**

**<version>1.16</version>**

**</dependency>**

**properties -> java Bulid Path -> Add Libray -> JUnit 4추가**

**java test에서 junit 사용.**

**@RunWith,@ContextConfiguration설정 후 Run As에 junitTest를 하면 된다.**

**@RunWith(SpringJUnit4ClassRunner.class)**

**@ContextConfiguration(locations = {"file:src/main/webapp/WEB-INF/spring/root-context.xml" })**

**public class DBTest {**

**@Test**

**public void DBConnectionTest() throws Exception {**

**Class.forName("org.postgresql.Driver");**

**Connection con = DriverManager.getConnection(**

**"jdbc.url=jdbc:postgresql://localhost:5432/postgres",**

**"postgres",**

**"1234");**

**System.out.println(con);**

**con.close();**

**}**

**}**

**root-context.xml에 설정!!!!**

**(Namespaces 체크 mybatis -spring, jdbc, context )**

**source 추가**

**<!-- JDBC-PostgreSQL -->**

**<bean id="dataSource"**

**class="org.springframework.jdbc.datasource.DriverManagerDataSource">**

**<property name="driverClassName" value="org.postgresql.Driver" />**

**<property name="url" value="jdbc.url=jdbc:postgresql://localhost:5432/postgres" />**

**<property name="username" value="postgres" />**

**<property name="password" value="1234" />**

**</bean>**

**dataSource Test(위의 DBTest와 같다.)**

**@RunWith(SpringJUnit4ClassRunner.class)**

**@ContextConfiguration(locations = { "file:src/main/webapp/WEB-INF/spring/root-context.xml" })**

**public class DSTest {**

**@Inject**

**private DataSource ds;**

**@Test**

**public void dsTest()throws Exception{**

**Connection con = ds.getConnection();**

**System.out.println(con);**

**con.close();**

**}**

**}**

**dataSource jdbc.properties 설정.**

**src/main/resources/ -> jdbc.properties 생성**

**jdbc.properties안에 해당 내용을 적어 준다.**

**jdbc.driverClassName=org.postgresql.Driver**

**jdbc.url=jdbc:postgresql://localhost:5432/postgres**

**jdbc.username=postgres**

**jdbc.password=1234**

**root-context.xml 변경**

**properties의 경로를 설정 and dataSource의 url을 변경.**

**해당 내용은 변경을 안해도 아무런 상관이 없지만 나중에 실 DB의 URL id password를 변경하기 쉽게 하기 위해서 이다.**

**<!-- properties -->**

**<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">**

**<property name="locations" value="classpath:/jdbc.properties" />**

**<property name="fileEncoding" value="UTF-8" />**

**</bean>**

**<!-- JDBC-PostgreSQL -->**

**<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">**

**<property name="driverClassName" value="${jdbc.driverClassName}" />**

**<property name="url" value="${jdbc.url}" />**

**<property name="username" value="${jdbc.username}" />**

**<property name="password" value="${jdbc.password}" />**

**</bean>**

**sqlSession설정 (mybatis) classpath의 경로는 src/main/resources이다.**

**classpath의 경로는 .classpath에 들어 가있다 (건들면 멘붕을 경험할 수 있다.)**

**※주의 사항 : sqlSession을 설정 하게 되면 DB DS Test의 내용에 에러가 뜨기 시작한다.**

**value="classpath\*:sql/\*\*/\*.xml은 .xml 확장자와 연결 된다.(규칙준수.)**

**mapping을 실수 하게 되면 되지 아니한다.**

**sqlSession은 해당 inject autoward 로 sqlSeeoin을 가져올때 사용한다.**

**classpath:/mybatis-config.xml해당 경로로 mybatis-config.xml을 사용해서 vo객체를 관리 할 수 있다.**

**ex) 나는 개인 적으로 sql을 mapper로 구분 짓는다.**

**<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">**

**<property name="dataSource" ref="dataSource" />**

**<property name="configLocation" value="classpath:/mybatis-config.xml">**

**</property>**

**<property name="mapperLocations"**

**value="classpath:mappers/\*\*/\*Mapper.xml"></property>**

**</bean>**

**<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate"**

**destroy-method="clearCache">**

**<constructor-arg name="sqlSessionFactory"**

**ref="sqlSessionFactory"></constructor-arg>**

**</bean>**

**ex) 밑에 꺼랑 내용이 다르니 연결이 안될 수도 있다.**

**그러니 저기 부분은 알고 쓰는것이 좋다.**

**<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">**

**<property name="mapperLocations" value="classpath\*:sql/\*\*/\*.xml"/>**

**<property name="dataSource" ref="dataSource" />**

**</bean>**

**<bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">**

**<constructor-arg index="0" ref="sqlSessionFactory" />**

**</bean>**

**mybatise 설정.**

**위의 value에 있던 경로에 xml 파일을 만든다.(나는 testMapper.xml로 만들었다.)**

**현재 시간을 구한다.**

**<?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="test">**

**<select id="time" resultType="java.lang.String">**

**SELECT now()**

**</select>**

**</mapper>**

**※MVC패턴상 대표 적인 네이밍.**

**VO : 객체 저장소**

**DAO(persistense ) : DB와 연결 하는 부분**

**Service : controller 받은 데이터를 DAO와 연결해서 DB로 저장 해 주는 부분.**

**(주로 데이터 가공. 그리고 트랜잭션을 이용해서 2DAO를 연결해 줄수도 있다.)**

**나중에 이미지 처리랑 댓글 처리 할때 사용. 설명은 그쪽으로.**

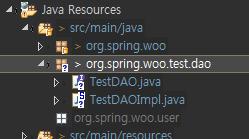
**controller : JSP와 연결**

**intersepter : 로그인 할때**

**자세한 내용은 CRUD에서!!!!!!!!!**

**TestSQL만들기**

**DAO만들기**

****

**TestDAO : 인터페이스 생성 (인터페이스는 굳이 안만들어도 문제는 없지만 나중에 이런것을 추상메소드로 상속하고 있구나를 쉽게 확인 할수 있다 (기능을))**

**TestDAOIml : DAO를 상속 받는다 그래서 DAOImpl로 지었다.**

**public interface TestDAO {**

**public String time()throws Exception;**

**}**

**---------------------------------------------------------------------------**

**@Repository**

**public class TestDAOImpl implements TestDAO {**

**@Inject**

**private SqlSession session;**

**@Override**

**public String time() throws Exception {**

**return session.selectOne("test.time");**

**}**

**}**

**@DAO 설정.(스프링 걸어주기)**

**root-context.xml 에 설정을 해줘야지 DAO를 읽어 올수 있다.**

**<context:component-scan base-package="org.spring.woo.test.dao" />**