**템플릿**

**프로젝트 설정**   
Generator : Jakarta EE  
Template : web application  
application server : tomcat  
language : java  
build system : gradle

Lombok  
file>setting>build, execution, deployment > compiler > annotation processors > enable annotation processing = true

Gradle  
file>setting>build, execution, deployment > build tool > gradle > 다음을 사용하여 태스크 실행 > intellij로 지정

**플러그인**

Lombok

mybatisx

**톰캣 설정**   
구성 편집 >   
deployment에서 아티팩트 추가.  
deployment > application context : /  
톰캣 자체 없으면 좌측 탭에서 + 눌러 로컬 톰캣서버 추가

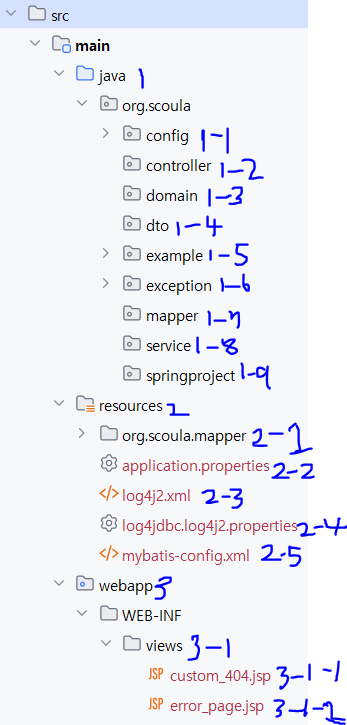
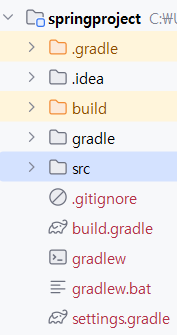
**settings.gradle**

rootProject.name=”아티팩트이름”으로 지정

**build.gradle**

plugins **{** id 'java'  
 id 'war'  
**}**group 'org.scoula'  
version '1.0-SNAPSHOT'  
  
repositories **{** mavenCentral()  
**}**ext **{** junitVersion = '5.11.0'  
  
 springVersion='5.3.37'  
 lombokVersion='1.18.30'  
**}**sourceCompatibility = '1.17'  
targetCompatibility = '1.17'  
  
tasks.withType(JavaCompile) **{** options.encoding = 'UTF-8'  
**}**dependencies **{** *//servlet* compileOnly('jakarta.servlet:jakarta.servlet-api:6.1.0')  
  
 *//spring* implementation("org.springframework:spring-context:$**{**springVersion**}**")  
 **{**exclude group: 'commons-logging', module: 'commons-logging'**}** implementation "org.springframework:spring-webmvc:$**{**springVersion**}**"  
 implementation 'javax.inject:javax.inject:1'  
  
 *//aop* implementation 'org.aspectj:aspectjrt:1.9.20'  
 implementation 'org.aspectj:aspectjweaver:1.9.20'  
  
 *//jsp,servlet,jstl* implementation('javax.servlet:javax.servlet-api:4.0.1')  
 compileOnly 'javax.servlet.jsp:javax.servlet.jsp-api:2.3.3'  
 implementation 'javax.servlet:jstl:1.2'  
  
 *//log4j2 logging system* implementation 'org.apache.logging.log4j:log4j-api:2.0.1'  
 implementation 'org.apache.logging.log4j:log4j-core:2.0.1'  
 implementation 'org.apache.logging.log4j:log4j-slf4j-impl:2.18.0'  
  
 *//jdbc 관련 로그 처리를 위한 로그시스템 라이브러리* implementation 'org.slf4j:slf4j-api:2.0.9'  
 runtimeOnly 'org.slf4j:jcl-over-slf4j:2.0.9'  
 runtimeOnly 'org.slf4j:slf4j-log4j12:2.0.9'  
 implementation 'log4j:log4j:1.2.17'  
 implementation 'org.bgee.log4jdbc-log4j2:log4jdbc-log4j2-jdbc4:1.16'  
  
  
  
 *//xml korean* implementation 'xerces:xercesImpl:2.12.2'  
  
 *//lombok* compileOnly "org.projectlombok:lombok:$**{**lombokVersion**}**"  
 annotationProcessor "org.projectlombok:lombok:$**{**lombokVersion**}**"  
  
 *//jackson json. java object<=>json parsing.* implementation 'com.fasterxml.jackson.core:jackson-databind:2.9.4'  
  
 *//test for spring & lombok* testImplementation "org.springframework:spring-test:$**{**springVersion**}**"  
 testCompileOnly "org.projectlombok:lombok:$**{**lombokVersion**}**"  
 testAnnotationProcessor "org.projectlombok:lombok:$**{**lombokVersion**}**"  
  
 *//database jdbc & DBCP* implementation 'com.mysql:mysql-connector-j:8.1.0'  
 implementation 'com.zaxxer:HikariCP:2.7.4'  
  
 *//스프링에서 jdbc와 transaction처리하게 하는 라이브러리* implementation "org.springframework:spring-tx:$**{**springVersion**}**"  
 implementation "org.springframework:spring-jdbc:$**{**springVersion**}**"  
  
 *//mybatis와 스프링 연동용 라이브러리* implementation "org.mybatis:mybatis:3.4.6"  
 implementation "org.mybatis:mybatis-spring:1.3.2"  
  
 *//junit test* testImplementation("org.junit.jupiter:junit-jupiter-api:$**{**junitVersion**}**")  
 testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:$**{**junitVersion**}**")  
**}**test **{** useJUnitPlatform()  
**}**

**디렉토리 구조 및 필수 파일들.**

**1**

**1자바코드만 모아놓은 패키지 2자바코드를 제외한 모든 것을 넣는 패키지 3내부로직 제외를 모은패키지**

* 1. 각 설정들(컨텍스트들)이 모인 패키지. RootConfig, ServletConfig, WebConfig 파일이 필수로 필요함.
  2. 각 설정들에 있는 설정과 등록된 객체, 매퍼 등을 이용하여 비즈니스로직처리(서비스)를 명하는 컨트롤러들이 모여있는 패키지
  3. Sql mapper를 통해서 sql의 결과를 객체 형대로 매핑되는 타입(VO)(DAO)들이 모인 패키지
  4. 일반 BL에서 쓰일 데이터 바구니 객체들을 모아놓은 패키지
  5. 1-2~1-4&1-6~1-8 형태가 들어있다. 관련된 것들만 모아놓은 패키지.
  6. 모든 컨트롤러들의 예외 발생시 처리하는 컨트롤러 어드바이스들이 들어있는 패키지
  7. BL처리하는 서비스 객체가 활용할 매퍼 인터페이스(쉬운 디비 연산을 위한)들이 모여있는 패키지.
  8. 실제 BL코드가 있는 서비스 객체들이 있는 패키지. 각 객체 안에선 매퍼를 이용한 디비 연산이 있을 수 있음.
  9. 아티팩트 폴더. 프로젝트 생성시 하나 만들어져 있음

2-1 연동할 1-7매퍼 인터페이스와 경로가 같아야하는 (sql지정 설정 관리)매퍼xml들이 모여있는 패키지

2-2 어플리케이션에서 사용할 설정정보들이 키-값 쌍으로 저장되어있는 파일. 꺼내 쓸 수 있음

2-3 로깅 시스템을 위한 설정파일, 2-4 jdbc 관련처리를 더 디테일하게 볼 수 있는 로깅시스템 설정파일

2-5 마이바티스의 대한 설정, 규칙 등을 지정하는 파일.

3-1 사용자가 보게되는 페이지들이 모여있는 패키지. 에러페이지는 공통으로 필요해서 템플릿화 함.

사용할 dependecy와 버전등 다양한 설정을 하는 build.gradle과 settings.gradle. 동기화 잊지 말기

**Resources 밑에 필수 파일들**

**애플리케이션 설정 src/main/resources/application.properties**

//application에서 사용할 설정들  
  
//db연결을 위한 jdbc 관련 설정.  
*#jdbc.driver=com.mysql.cj.jdbc.Driver  
#->*jdbc.driver=net.sf.log4jdbc.sql.jdbcapi.DriverSpy  
*#jdbc.url=jdbc:mysql://127.0.0.1:3306/scoula\_db  
#->*jdbc.url=jdbc:log4jdbc:mysql://localhost:3306/scoula\_db  
jdbc.username=scoula  
jdbc.password=1234

**로그 시스템 설정 src/main/resources/log4j2.xml**

*<?*xml version="1.0" encoding="UTF-8"*?>  
<!--log를 설정할 때 level 이 쓰여있는 것이 보이는데 이 역시 단계가 존재한다.  
높은 등급에서 낮은 등급으로의 6개의 로그 레벨을 가지며 지정한 레벨 등급 이상의 로그만 표출하는 방식이다.  
일반적으로 log level은  
ALL > 다 출력  
TRACE > 경로 추적  
DEBUG > 일반정보를 상세히 나타낼 때  
INFO > 일반정보를 나타낼때  
WARN > 에러는 아니지만 주의가 필요  
ERROR > 일반 에러  
FATAL > 가장 크리티컬한 에러  
OFF 출력하지 말아라 그냥.  
순으로 볼 수 있다. ( 낮은거에서 높은순으로 ALL → OFF)  
결론적으로 DEBUG 를 level로 선언했다면 DEBUG 이상의 것들의 log를 표출하는것이다.  
-->*<Configuration>  
 *<!-- Appender , Layout 설정 -->* <Appenders>  
 *<!--- Appenders는 로그가 출력되는 위치를 나타냄  
 - 정의한 name은 AppenderRef 태그에서 참조를 할 때 사용-->* <Console name="console" target="SYSTEM\_OUT">  
 <PatternLayout charset="UTF-8" pattern="%-5level %c(%M:%L) -%m%n"/>  
 </Console>  
 </Appenders>  
  
  
 *<!-- Logger 설정 -->* <Loggers> *<!--로깅 작업의 주체, 각 패키지 별, 로그 정책을 설정 할 수 있다.-->* <Root level="INFO"> *<!--일반적인 로그 정책에 대해 정의를 할 수 있으며, 반드시 한개를 정의해야한다.-->* <AppenderRef ref="console"/>  
 </Root>  
  
 *<!--해당 범위에서는 레벨 몇부터만 출력하라와 중복 출력 여부를 지정-->* <Logger name="\_org.springframework.web.servlet.HandlerMapping.Mappings" level="DEBUG" additivity="false">  
 *<!--여러개 정의를 할수 있으며 name에 정의하는 패키지 범위를 설정 할 수 있다.-->  
 <!--- name : 로그의 name, 로그를 정의할 패키지의 범위  
 - level : 로그의 Level(위에 log level 참고)  
 - additivity : 중복로깅여부-->* <AppenderRef ref="console"/>  
 </Logger>  
  
 <Logger name="org.scoula" level="INFO" additivity="false">  
 <AppenderRef ref="console"/>  
 </Logger>  
  
 <Logger name="org.springframework" level="INFO" additivity="false">  
 <AppenderRef ref="console"/>  
 </Logger>  
  
 <Logger name="jdbc" level="WARN" additivity="false">  
 <AppenderRef ref="console"/>  
 </Logger>  
  
 <Logger name="jdbc.sqlonly" level="INFO" additivity="false">  
 <AppenderRef ref="console"/>  
 </Logger>  
 </Loggers>  
</Configuration>

**JDBC 처리 관련 log 세부사항 출력 설정 src/main/resources/log4jdbc.log4j2.properties**

log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator  
log4jdbc.auto.load.popular.drivers=false  
log4jdbc.drivers=com.mysql.cj.jdbc.Driver

**mybatis 설정 src/main/resources/mybatis-config.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>*<!--root 태그-->* <settings>  
 *<!--db에서 칼럼명 snake표기법을  
 java vo객체에서는 필드명을 camel case로-->* <setting name="mapUnderscoreToCamelCase" value="true"/>  
  
 </settings>  
  
 <typeAliases>  
 *<!--긴 패기지 이름을 생략할 수 있게 하는 설정-->* </typeAliases>  
  
</configuration>

**src>main>java>group id>config 밑에 있는 필수 파일들**

**범용 설정 src/main/java/org/scoula/config/RootConfig.java**

package org.scoula.config;  
  
import com.zaxxer.hikari.HikariConfig;  
import com.zaxxer.hikari.HikariDataSource;  
import org.apache.ibatis.session.SqlSession;  
import org.apache.ibatis.session.SqlSessionFactory;  
import org.mybatis.spring.SqlSessionFactoryBean;  
import org.mybatis.spring.annotation.MapperScan;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.context.annotation.PropertySource;  
import org.springframework.jdbc.datasource.DataSourceTransactionManager;  
  
import javax.sql.DataSource;  
  
@Configuration  
@ComponentScan(basePackages = {  
 *//controller가 아닌 범용 클래스들 스캔* "org.scoula",  
})  
@PropertySource({  
*//rootconfig에서 사용할 .properties파일들 경로 지정.* "classpath:/application.properties"  
})  
@MapperScan(basePackages = {  
 *//rootconfig 컨텍스트에 등록시킬  
 //동적으로 구현할 mapper 인터페이스 위치 지정* "org.scoula.mapper"  
})  
public class RootConfig {  
 *//범용, 디비 관련  
 //해당 클래스의 컨텍스트에 등록된 빈은 다른데에서  
 //활용할 수 있다.  
  
 //.properties 내용에 있는 키에 대한 밸류* @Value("${jdbc.driver}") String driver;  
 @Value("${jdbc.url}") String url;  
 @Value("${jdbc.username}") String username;  
 @Value("${jdbc.password}") String password;  
  
 @Bean*//hikari datasource객체 컨텍스트에 빈등록* public DataSource dataSource(){  
 HikariConfig config = new HikariConfig();  
  
 config.setDriverClassName(driver);  
 config.setJdbcUrl(url);  
 config.setUsername(username);  
 config.setPassword(password);  
  
 HikariDataSource dataSource = new HikariDataSource(config);  
 return dataSource;  
 }  
   
 @Autowired*//application context는 스프링이 자동으로 컨텍스트에 등록해놓음* ApplicationContext applicationContext;  
   
 @Bean*//sqlsessionfactory를 컨텍스트에 빈 등록* public SqlSessionFactory sqlSessionFactory() throws Exception {  
 SqlSessionFactoryBean sqlSessionFactory = new SqlSessionFactoryBean();  
 sqlSessionFactory.setConfigLocation(  
 applicationContext.getResource("classpath:/mybatis-config.xml")  
 );  
 sqlSessionFactory.setDataSource(dataSource());  
  
 return (SqlSessionFactory) sqlSessionFactory.getObject();  
 }  
  
 @Bean*//datasource tx manager도 등록* public DataSourceTransactionManager transactionManager(){  
 return new DataSourceTransactionManager(dataSource());  
 }  
}

**frontcontroller(dispatchservlet) 설정 src/main/java/org/scoula/config/ServletConfig.java**

package org.scoula.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.web.multipart.MultipartResolver;  
import org.springframework.web.multipart.support.StandardServletMultipartResolver;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  
import org.springframework.web.servlet.config.annotation.ViewResolverRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
import org.springframework.web.servlet.view.InternalResourceViewResolver;  
import org.springframework.web.servlet.view.JstlView;  
  
import java.awt.\*;  
  
@EnableWebMvc  
@ComponentScan(basePackages = {  
 *//controller들이 있는 패키지 경로 나열* "org.scoula.controller",*//컨트롤러 모여있음* "org.scoula.exception"*//exception 대응 컨트롤러*})  
public class ServletConfig implements WebMvcConfigurer {  
 *//dispatcher servlet == front controller* @Override*//resources 요청 시 처리* public void addResourceHandlers(ResourceHandlerRegistry registry) {  
 registry  
 .addResourceHandler("/resources/\*\*")  
 .addResourceLocations("/resources/");  
 }  
  
 @Override*//jsp view resolver 설정.* public void configureViewResolvers(ViewResolverRegistry registry) {  
 *//jsp 엔진 사용* InternalResourceViewResolver bean=new InternalResourceViewResolver();  
   
 *//jsp 엔진 결정* bean.setViewClass(JstlView.class);  
 bean.setPrefix("/WEB-INF/views/");  
 bean.setSuffix(".jsp");  
   
 *//jsp로 가게끔* registry.viewResolver(bean);  
 }

@Bean*//http body에다가 binary데이터를 넣을때 필요함* public MultipartResolver multipartResolver() {  
 StandardServletMultipartResolver multipartResolver=new StandardServletMultipartResolver();  
 return multipartResolver;  
 }  
}

**앱 관련 설정(web.xml대체) src/main/java/org/scoula/config/WebConfig.java**

package org.scoula.config;  
  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.filter.CharacterEncodingFilter;  
import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;  
  
import javax.servlet.Filter;  
import javax.servlet.MultipartConfigElement;  
import javax.servlet.ServletRegistration;  
  
@Slf4j  
@Configuration  
public class WebConfig extends AbstractAnnotationConfigDispatcherServletInitializer {  
 *//web.xml 대체. 톰캣에게 앱에 대해서 알리는 역할.  
 //앱 관련 설정. 필터링. 캐릭터셋 적용.* final String LOCATION="c:/upload/";*//파일 업로드시 서버의 저장위치* final long MAX\_FILE\_SIZE = 10L \* 1024 \* 1024;*//한 파일 최대 크기* final long MAX\_REQUEST\_SIZE = 20L \* 1024 \* 1024;*//한 요청 최대 크기* final int FINAL\_SIZE\_THRESHOLD = 1024 \* 1024 \* 5;*//메모리를 사용할지, 임시파일을 사용할지 기준.* @Override*//rootconfig class 정보 갖고오기* protected Class<?>[] getRootConfigClasses() {  
 return new Class[]{RootConfig.class};  
 }  
  
 @Override*//servletconfig class 정보 갖고오기* protected Class<?>[] getServletConfigClasses() {  
 return new Class[]{ServletConfig.class};  
 }  
  
 @Override*//dispatch servlet이 담당할 url 매핑 패턴 지정* protected String[] getServletMappings() {  
 return new String[]{"/"};  
 }  
   
 *//post body 문자 인코딩 필더 설정* protected Filter[] getServletFilters(){  
 CharacterEncodingFilter characterEncodingFilter = new CharacterEncodingFilter();  
  
 *//utf-8로 설정* characterEncodingFilter.setEncoding("UTF-8");  
 characterEncodingFilter.setForceEncoding(true);  
  
 return new Filter[]{characterEncodingFilter};  
 }  
  
 @Override*//서블릿 설정* protected void customizeRegistration(ServletRegistration.Dynamic registration){  
 *//nohandlerfound 404 예외 발생시 예외 던지게끔 설정.* registration.setInitParameter("throwExceptionIfNoHandlerFound", "true");  
  
 *//바이너리 데이터를 맡을 multipart설정* MultipartConfigElement multipartConfig=new MultipartConfigElement(  
 LOCATION,  
 MAX\_FILE\_SIZE,  
 MAX\_REQUEST\_SIZE,  
 FINAL\_SIZE\_THRESHOLD  
 );  
  
 registration.setMultipartConfig(multipartConfig);  
 }  
}

**기타 필수 //예외, 매퍼xml템플릿**

**컨트롤러의 예외 처리   
src/main/java/org/scoula/exception/CommonExceptionAdvice.java**

package org.scoula.exception;  
  
import lombok.extern.log4j.Log4j2;  
import org.springframework.http.HttpStatus;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.ControllerAdvice;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.bind.annotation.ResponseStatus;  
import org.springframework.web.servlet.NoHandlerFoundException;  
  
import javax.servlet.http.HttpServletRequest;  
  
@ControllerAdvice*//@Controller의 자식. 역할 분별. 모든 컨트롤러에 적용됨*@Log4j2*//로깅를 위한*public class CommonExceptionAdvice {  
 *//컨트롤러들에서 에러가 발생했을 때 작동하는 컨트롤러.  
  
 //범용적인 예외 발생시 해당 메서드가 맡음* @ExceptionHandler(Exception.class)  
 public String except(Exception e, Model model) {  
 *log*.error("Exception,,,,,"+e.getMessage());  
  
 model.addAttribute("exception", e);  
  
 *log*.error(model);  
  
 *// 경로/error\_page.jsp로 포워딩* return "error\_page";*//다른 컨트롤러의 반환과 의미가 같음* }  
  
 *//특정(지금은 nohandlerfound 404) 예외 발생시 해당 메서드가 맡음* @ExceptionHandler(NoHandlerFoundException.class)  
 @ResponseStatus(HttpStatus.*NOT\_FOUND*)*//404상대로 응답 놰라* public String handle404(NoHandlerFoundException ex, Model model, HttpServletRequest request) {  
 *log*.error(ex);  
 *//저장* model.addAttribute("uri",request.getRequestURI());  
  
 return "custom\_404";*// 경로/custom\_404.jsp로 포워딩* }  
}

**예외처리 뷰(페이지)  
src/main/webapp/web-inf/view/custom404.jsp**

<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
 <h1>해당 URL(${uri})은 존재하지 않음</h1>  
</body>  
</html>

**src/main/webapp/web-inf/view/error\_page.jsp**  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<!DOCTYPE html>  
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  
<%@ page session="false" import="java.util.\*" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>*<%--에러 메세지 출력 jstl--%>*<h4><c:out value="${exception.getMessage()}"></c:out></h4>  
  
<ul>*<%--에러 메세지 출력 jstl--%>* <c:forEach items="${exception.getStackTrace()}" var="stack">  
 <li><c:out value="${stack}"></c:out></li>  
 </c:forEach>  
</ul>  
</body>  
</html>

**Mybatis mapper 템플릿 src/resources/org/scoula/mapper/MapperTemplate.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="org.scoula.mapper.TestMapper" >

</mapper>