회원인증, fileUpload, MyBatis

1. 요구사항 : 회원가입(감사메일 자동 발송), 로그인, 로그아웃, 회원정보수정 가능

책이미지를 파일첨부하여 책등록, 책상세보기, 책정보수정, 책리스트(페이징처리)

1. pom.xml 추가
2. web.xml 추가
3. 패키지 만들기 ex) controller, service, dto, dao, util
4. 컨트롤러 만들기 (main,member,book)

*@*RequestMapping(value="main",method= {RequestMethod.GET,RequestMethod.POST})

어떤 방식으로 넘어올지에 따라 달라진다

1. properties 설정 하기 (DB와 mail)
2. META-INF/ 폴더 만들기 ex) db, config, mappers

Db 폴더 안에 initDB.sql과 mapper.sql 만들기

InitDb.sql

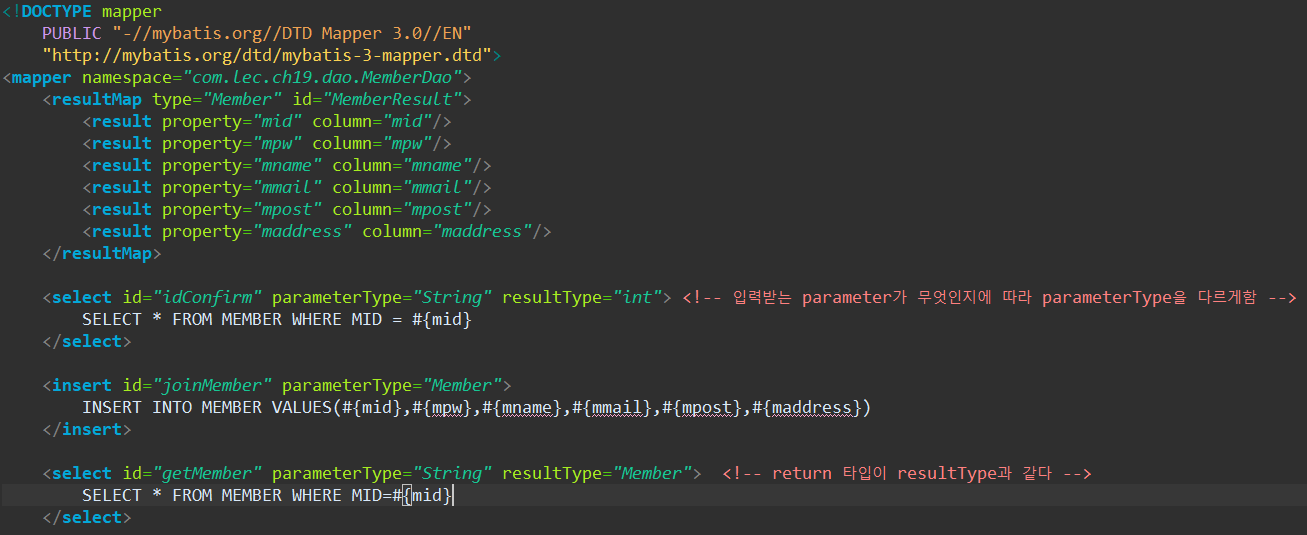
--SEQUENCE & TABLE DROP/CREATE

--DUMMY DATA INSERT

Mapper.sql

--MEMBER.XML (회원가입, id중복 체크,id로 memberDto 가져오기, 로그인, 정보수정 )

1. Sqlmap/mappers/book.xml & member.xml 만들기



입력받는 parameter가 무엇인지에 따라 parameterType을 다르게해줘야 한다.

resultMap은 주로 list형식으로 출력될 때 사용하고 위에서 지정해준 엘리아스(id=”MemberResult”)로 호출한다

resultType = return타입인데 list형식이여도 상관은 없다

전체적인 틀

DTO(페이징이 필요하면 startRow&endRow추가)

-mybatis-config.xml – mappers(쿼리문 xml) – DAO(interface)

-servlet-context.xml에 태그 추가

<mybatis-spring base-package=””>

-root-context.xml에 빈 추가

<**context:property-placeholder** location=*"classpath:META-INF/property/db.properties"*/>

<!-- DataSource 빈 -->

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

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

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

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

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

</**bean**>

<!-- SqlSessionFactoryBean 빈 -->

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

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

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

</**bean**>

<!-- SqlSessionTemplate 빈 필요 -->

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

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

</**bean**>

-servlet-context.xml에 빈 추가

<**context:component-scan** base-package=*"com.lec.ch19"* />

<**mybatis-spring:scan** base-package=*"com.lec.ch19.dao"*/>

<**context:property-placeholder** location=*"classpath:META-INF/property/mail.properties"*/>

<!-- 메일발송 빈 -->

<**beans:bean** id=*"mailSender"*

class=*"org.springframework.mail.javamail.JavaMailSenderImpl"*>

<**beans:property** name=*"host"* value=*"${mail.host}"*/>

<**beans:property** name=*"port"* value=*"${mail.port}"*/>

<**beans:property** name=*"username"* value=*"${mail.username}"*/>

<**beans:property** name=*"password"* value=*"${mail.password}"*/>

<**beans:property** name=*"javaMailProperties"*>

<**beans:props**>

<**beans:prop** key=*"mail.smtp.auth"*>true</**beans:prop**>

<**beans:prop** key=*"mail.smtp.starttls.enable"*>true</**beans:prop**>

<**beans:prop** key=*"mail.debug"*>true</**beans:prop**>

<**beans:prop** key=*"mail.transport.protocol"*>smtp</**beans:prop**>

<**beans:prop** key=*"mail.smtp.ssl.trust"*>smtp.gmail.com</**beans:prop**>

<**beans:prop** key=*"mail.smtp.ssl.protocols"*>TLSv1.2</**beans:prop**>

</**beans:props**>

</**beans:property**>

</**beans:bean**>

<!-- 파일첨부 -->

<**beans:bean** id=*"multipartResolver"* class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>

<**beans:property** name=*"maxUploadSize"* value=*"10485760"*/>

</**beans:bean**>

## [Model 객체](#Model%--%EA%B-%-D%EC%B-%B-)

Model 객체는 Controller 에서 생성된 데이터를 담아 View 로 전달할 때 사용하는 객체이다.

※ Servlet의 request.setAttribute() 와 비슷한 역할을 함

addAttribute("key", "value") 메서드를 이용해 view에 전달할 데이터를 key, value형식으로 전달할 수 있다.

Ex)

model.addAttribute("joinResult", memberService.joinMember(member, httpSession));

member dto와 세션을 만들어서 서비스단의 joinMember dao 로직을 수행한 것을 joinResult에 넣어주고, model객체에 추가한다

[**@ModelAttribute**](#%--ModelAttribute) **Member member, Book book //DTO를 이용할 때**

일반적으로 dto를 통해서 한번에 정보를 받는데, 회원가입만 했는데 member가 model객체에 들어가서 로그인을 안했지만 로그아웃 header가 생긴다

이를 방지하고자 @ModelAttribute 어노테이션을 써서 다른 이름으로 model에 매개변수를 넣어주고, 로그인 로직은 넣어준 매개변수로 호출해서 일반적인 DTO와 구분을 하게 해준다.

@ModelAttribute는 HTTP Body 내용과 HTTP 파라미터의 값들을 Getter, Setter, 생성자를  통해 주입하기 위해 사용한다.

일반 변수의 경우 전달이 불가능하기 때문에 model 객체를 통해서 전달해야 한다.

※@ModelAttribute("파라미터명")