1. pom.xml에 의존추가 부분

```
<!-- JDBC -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-jdbc</artifactId>
			<version>5.2.8.RELEASE</version>
		</dependency>
		<!-- mybatis -->
		<dependency>
			<groupId>org.mybatis</groupId>
			<artifactId>mybatis</artifactId>
			<version>3.5.5</version>
		</dependency>
		<!--  mybatis-spring -->
		<dependency>
			<groupId>org.mybatis</groupId>
			<artifactId>mybatis-spring</artifactId>
			<version>2.0.5</version>
		</dependency>
		<!-- lombok -->
		<dependency>
			<groupId>org.projectlombok</groupId>
			<artifactId>lombok</artifactId>
			<version>1.18.8</version>
			<scope>provided</scope>
		</dependency>

		<!-- Mail sender -->
		<dependency>
			<groupId>javax.mail</groupId>
			<artifactId>mail</artifactId>
			<version>1.4.7</version>
		</dependency>
		<!--  file upload  -->
		<dependency>
			<groupId>commons-fileupload</groupId>
			<artifactId>commons-fileupload</artifactId>
			<version>1.3.1</version>
		</dependency>
```

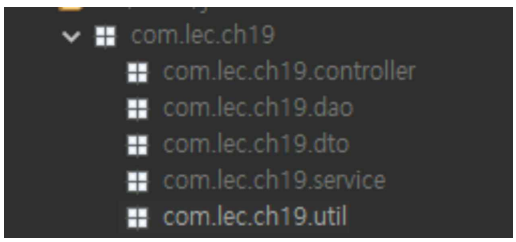
2. web.xml에 한글처리부분 및 *.do처리 부분

```
	<servlet-mapping>
		<servlet-name>appServlet</servlet-name>
		<url-pattern>*.do</url-pattern>
	</servlet-mapping>
	<!-- 한글처리 -->
	<filter>
		<filter-name>encodingFilter</filter-name>
		<filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
		<init-param>
			<param-name>encoding</param-name>
			<param-value>UTF-8</param-value>
		</init-param>
		<init-param>
			<param-name>forceEncoding</param-name>
			<param-value>true</param-value>
		</init-param>
	</filter>
	<filter-mapping>
		<filter-name>encodingFilter</filter-name>
		<url-pattern>/*</url-pattern>
	</filter-mapping>
```

3. index.jsp 생성

main.do로 가라

4. 필요한 페키지 생성

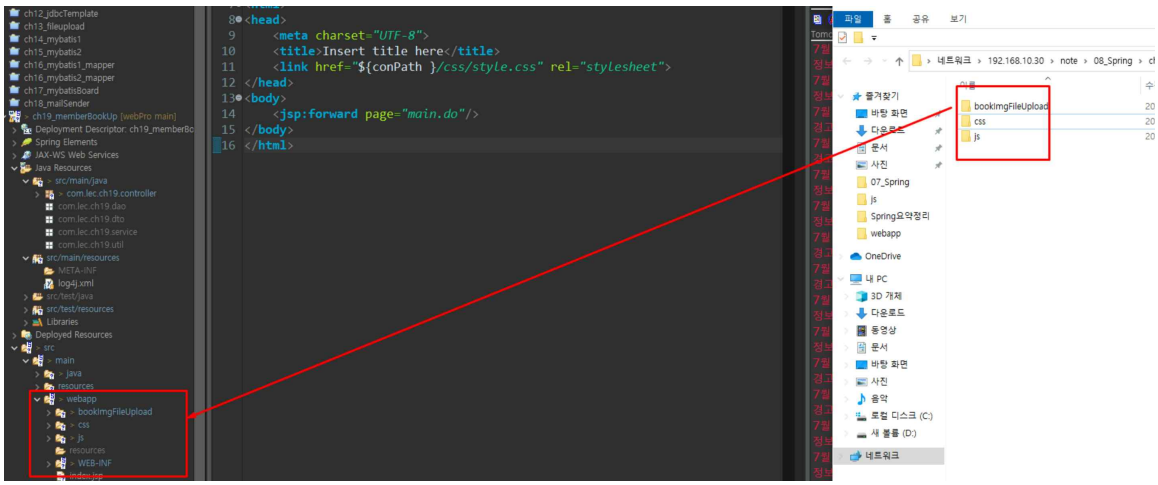


5. 컨트롤러 생성 (min.do만 처리할 아이)

6. 지금까지 오타 없는 확인

```java
@Controller
public class MainController {
    @RequestMapping(value="main" , method = {RequestMethod.GET,RequestMethod.POST})
        public String main() {
         return "main";
    }
}
```

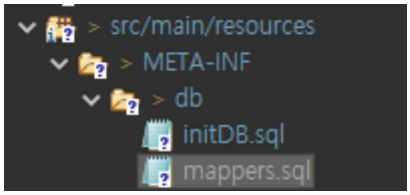
-필요한아이 webapp 에 가져오기



-servlet에 필요한 아이

```
<!-- Handles HTTP GET requests for /resources/** by efficiently serving up
<resources mapping="/resources/**" location="/resources/" />
<resources mapping="/bookImgFileUpload/**" location="/bookImgFileUpload/"
<resources mapping="/css/**" location="/css/" />
<resources mapping="/js/**" location="/js/" />
<resources mapping="/img/**" location="/img/" />
```

-db생성



-initDB

```
-- SEQUENCE & TABLE DROP/CREATE



-- DUMMY DATA INSERT
```

DROP TABLE BOOK CASCADE CONSTRAINTS;

DROP TABLE MEMBER CASCADE CONSTRAINTS;

DROP SEQUENCE BOOK_SQ;

-MEMBER 테이블

```
CREATE TABLE MEMBER(
        mID VARCHAR2(100) PRIMARY KEY,
        mPW VARCHAR2(100) NOT NULL,
        mNAME VARCHAR2(100) NOT NULL,
        mMAIL VARCHAR2(100) NOT NULL,
        mPOST VARCHAR2(100),        -- 우편번호 (API사용)
        mADDR VARCHAR2(100)         -- 주소
);
```

- BOOK테이블

```
CREATE SEQUENCE BOOK_SQ MAXVALUE 99999999 NOCYCLE NOCACHE;

CREATE TABLE BOOK(
    bNUM NUMBER(8) PRIMARY KEY,
    bTITLE VARCHAR2(100) NOT NULL,
    bWRITER VARCHAR2(100) NOT NULL,
    bRDATE DATE DEFAULT SYSDATE NOT NULL,
    bIMG1 VARCHAR2(100) DEFAULT 'noImg.png' NOT NULL,
    bIMG2 VARCHAR2(100) DEFAULT 'noImg.png' NOT NULL,
    bINFO VARCHAR2(1000)
    );
```

-- DUMMY DATA INSERT

```
INSERT INTO MEMBER VALUES ('aaa','1','손석구','son@naver.com','12345','서울');

INSERT INTO BOOK VALUES (BOOK_SQ.NEXTVAL, 'SPRING','김이나',SYSDATE, 'noImg.png','noImg.png','스프링개념서');

INSERT INTO BOOK (bNUM, bTITLE, bWRITER, bRDATE, bIMG1, bINFO)
    VALUES (BOOK_SQ.NEXTVAL, 'ORACLE','고작가',SYSDATE, 'noImg.png','스프링개념서');

INSERT INTO BOOK (bNUM, bTITLE, bWRITER, bRDATE,  bINFO)
    VALUES (BOOK_SQ.NEXTVAL, 'ORACLE','고작가',SYSDATE,'스프링개념서');

SELECT * FROM BOOK;
```

-mappers

```
-- Member.xml(회원가입,  id로 memberDto로 가져오기,  로그인, 정보수정)
--Book.xml (페이징없이 신규순list, 페이지징! 포함도서list(책이름순), 책갯수 , 상세보기, 도서등록 , 도서수정
```

-- Member.xml(회원가입, id로 memberDto로 가져오기, 로그인, 정보수정)

-- idConfirm

SELECT * FROM MEMBER WHERE mID='aaa';

-- joinMember

INSERT INTO MEMBER VALUES ('bbb','111','유재석','yu@naver.com','67890','서울');--getDetailMember

-- modifyMember

UPDATE MEMBER SET mNAME='오석구',

mPW = '123',

mMAIL = 'kok3443@naver.com',

mPOST = '12345',

mADDR ='제주'

where mID='aaa';

SELECT * FROM MEMBER WHERE mID='aaa';

--Book.xml (페이징없이 신규순list, 페이지징! 포함도서list(책이름순), 책갯수 , 상세보기, 도서등록 , 도서수정)

```
-- mainList

SELECT * FROM BOOK;

SELECT * FROM (SELECT ROWNUM RN, A.*

                    FROM (SELECT * FROM BOOK ORDER BY bRDATE DESC)A)

                    WHERE RN BETWEEN 1 AND 3;


-- bookList

SELECT * FROM (SELECT ROWNUM RN, A.*

                    FROM (SELECT * FROM BOOK ORDER BY bTITLE )A)

                    WHERE RN BETWEEN 1 AND 3;


-- totCntBook

SELECT COUNT(*) FROM BOOK;


-- getDetailBook

SELECT * FROM BOOK WHERE BNUM =1;


-- registerBook

INSERT INTO BOOK VALUES (BOOK_SQ.NEXTVAL, 'SPRING','김이나',SYSDATE,
'noImg.png','noImg.png','마이바티스어려움책');

SELECT * FROM BOOK;
```

```
--modifyBook

UPDATE BOOK SET BTITLE='MYBA',
                                bWRITER = '이소영',
                                bIMG1 = 'noImg.png',
                                bIMG2 = 'noImg.png',
                                bINFO ='마이바티스'
                where bNUM='4';

SELECT * FROM BOOK WHERE BNUM = 4;
```

- DTO 생성

```
println( 백업파일 :    + backupPath + bimg[idx]);// 백업 : 있다가
= filecopy(uploadPath + bimg[idx], backupPath + bimg[idx]);
```

```
private boolean filecopy(String serverFile, String
backupFile) {
		boolean isCopy = false;
		InputStream is = null;
		OutputStream os = null;
		try {
			File file = new File(serverFile);
			is = new FileInputStream(file);
			os = new FileOutputStream(backupFile);
			byte[] buff = new byte[(int) file.length()]
			while(true) {
				int nReadByte = is.read(buff);
				if(nReadByte == -1) break;
				os.write(buff, 0, nReadByte);
			}
			isCopy = true;
		} catch (Exception e) {
			System.out.println(e.getMessage());
		} finally {
			try {
				if(os!=null) os.close();
				if(is!=null) is.close();
			} catch (IOException e) {
				System.out.println(e.getMessage());
			}
		}
		return isCopy;
	}
```