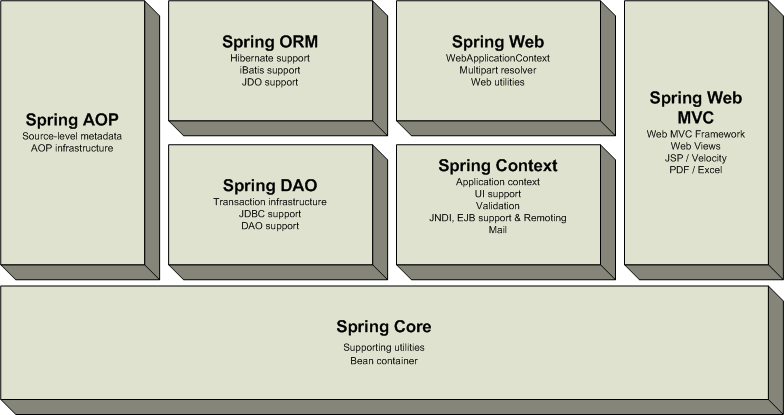
ㅇ



**- spring framework**

**1. ioc**

**2. di**

**3. aop**

**4. mvc**

**5. spring dao== jdbc ==> mybatis**

**6. 다른 db 연결 기술 지원**

**spring(main) + mybatis**

**mybatis**

**DB 이용**

**SPRING FRMEWORK**

**1>자바 객체 생성-관리=bean의 라이프 사이클 관리한다**

**2> bean의 의존성을 관리한다.(ioc , di)**

**3> AOP 기능 제공한다.**

**4> MVC 기능 제공한다.**

**5> dao, vo 등 기존 non-spring 환경에서도 사용하던 객체 그대로 사용 가능하다(스프링 라이브러리를 상속받거나 하지 않아도)**

**= pojo bean 사용 가능하다**

**(plain old java object)**

**6> spring dao 기능으로 db 연동 기능 제공한다.**

**7> 각종 다른 프레임워크 연동 기능 제공한다.**

**8> mybatis 프레임워크 연동 가능하다**

**= spring dao 기능 대신 사용하는 형태로 현재 개발 추세**

**9> spring 은 db 연결 기능으로**

**spring + jdbc**

**spring + spring dao**

**spring + mybatis**

**환경이 가능하다.**

**MYBATIS FRMEWORK**

**- 객체지향 언어인 자바의 관계형 데이터 베이스 프로그래밍을 좀더 쉽게 할수 있게 도와주는 개발 프레임워크.**

**- 자바에선 데이터베이스 프로그래밍을 하기 위해 JDBC(자바에서 제공하는 데이터베이스 프로그래밍 API)를 제공**

**- JDBC는 관계형 데이터 베이스를 사용하기 위해 다양한 API를 제공**

**- JDBC는 코드 반복이 너무 많다는 단점이 있음**

**- 처음에는 ibatis 라는 이름으로 시작 ->MYBATIS로 변경**

**jdbc driver , jdbc url, account, password**

**"jdbc:oracle:thin:@ip:port:xe"**

**==> 자바 소스 제외 분리 xml**

**xml**

**<select id="a">**

**select \* from board**

**</select>**

**자바소스**

**마이바티스... "a" sql 실행 문장**

**try{**

**1. Class.forNam(jdbc driver")**

**2. Connection c = DriverManager.getConnection**

**(jdbc url, account, password)**

**3. sql – 결과**

**4. c.close()**

**}catch(Exception E){**

**}**

**1. 코드 반복 많다 : db-config.xml**

**2. 자바 언어 내부에 sql 언어 포함 : sql-mapping.xml**

**3. db 연결 복잡한 정보 매번 작성 : db-config.xml**

**db-config.xml-> 1개**

**board-mapping.xml**

**member-mapping.xml**

**1. xml 설정 읽어라**

**2. db 연결 객체 가져와라**

**3. sql 정보 가져와서 실행 – 결과(BoardVO, String, AarrayList) 설정**

**4. 출력**

**0. 라이브러리 준비**

**- mybatis.jar : pom.xml <dependency > .....**

**- ojdbc6.jar : 오라클 jdbc driver 파일 압축파일**

**1. xml**

**mybatis-config.xml**

**emp-mapping.xml**

**spring mvc**

**form action="/mvc/test" input name="a"-->요청 파라미터 변수**

**@RequestMapping("/test")**

**public String test(EmpVO vo){**

**EmpVO : 멤버변수 a : setA()**

**mybatis**

**<select id="" resultType="test.A"**

**select \* from employees**

**employees 테이블 컬럼명 = A 클래스 멤변수명**

**EmpDAO**

**EmpVO**

**EmpMain**

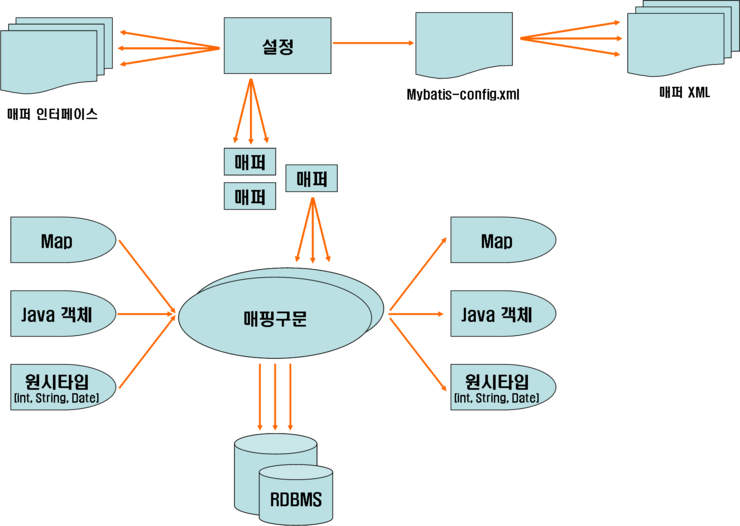
**SPRING MVC**

**emp-mapping.xml**

**mybatis-config.xml**

**mybatis.jar**

**odjcb.jar**



Mybatis가 JDBC보다 좋은점은 뭘까??

1.간결한 코드처리가 가능하다.

(JDBC는 try-catch-finally처럼 데이터를 처리하기까지 많은 코드를 직접 작성해야 한다.

하지만 mybatis는 이러한 코드를 상당히 줄일 수 있어 개발속도가 향상,

Spring과 mybatis를 같이 사용하면 코드 제작 없이 JDBC의 처리가 가능.)

2.SQL문을 따로쓴다.(어떻게보면 코드를 재사용한다고 생각할 수도 있다.)

 (JDBC는 SQL문을 처리하기 위해 별도의 파일을 작성하는등 번거로운 작업이많다.

Mybatis는 XML이나 애노테이션방식으로 SQL문을 별도로 처리가능하다.

==>별도로 처리도 가능하고 / JDBC처럼 사용해도 되고

따라서 많은 양의 SQL에 대해서는 별도의 XML파일에서 작업하여 간결함을 유지)

3.Mybatis-Srping이 있어 코드의 양이 줄여진다.

(스프링 프레임워크에서 mybatis-Spring라이브러리를 이용하면

SQL 문의 호출없이도 데이터를 얻을 수 있다.

따라서 스프링에서는 mybatis와 스프링을 연계하면 코드의 양을 줄일 수 있다)

MyBatis는 스프링에서 어떻게 사용될까??

스프링에서는 크게

1.UI를 담당하는 부분(view)

2.사용자의 요구사항을 처리하는 부분(service)

3.데이터를 처리하는 부분(DAO)

이 있다.

**DAO(Data Access Object)는 Mybatis를 호출하여 사용하는 구조입니다.**

아이바티스(ibaits)는 아파치 재단에서 진행되던 오픈소스 프로젝트로

프로젝트 팀원들이 구글 코드로 이전하면서  ibatis 는 개발이 중단 되었다.

(사실상 개발 중단이 아닌 프로젝트 소속기관을 이전하면서 이름만 변경된 것이다 )

구글 코드로 이전하면서 ibatis는 새롭게 이름을 다시 지었는데  이것이 Mybatis 다.

**- JDBC 의 단점**

**DB 연결**

**SQL 전송**

**SQL 결과 리턴**

**DB 연결 해제**

**예외처리**

**코드 반복과 사용 객체 해제 빈번**

**- Mybatis 의 대표적 특징은 SQL문을 분리하는 SQL Maper 프레임 워크라는 점이다.**

**- SQL을 분리한다는 것은 무엇인가?**

JDBC 로 데이터베이스로 접속하는 프로그래밍을 한다면

pstmt = conn.prepareStatement("select \* from board");

.............

쿼리문이 자바 소스코드 안에 문자열로 취급되어 들어간다

**1. SQL과 자바는 서로 다른 언어이다.**

이 두가지가 한 파일(또는 클래스) 내에 섞여 있다는 것은 굉장히 껄끄러운 문제점이다.

객체지향 프로그래밍은 각 메서드 클래스별 하나의 책임(SRP)을 맡길 것을 권장하는데,

1개의 메서드에 1개의 역할이 아닌 2개의 언어가 혼재

**2. 객체지향적인 SQL 작성의 어려움**

XML

<select id=”boardlist” resultMap=”BoardVO”>

select \* from board

</select>

**쿼리문을 xml 로 분리한 mybatis 프로그래밍 중 일부이다.**

**mybatis는 xml 로 쿼리문을 따로 관리하고  자바의 DAO 클래스(Persistence계층)들은 필요한 쿼리문을 이곳에서 불러서 사용한다.**

**- JDBC 코드의 대안 마이바티스**

**- 마이바티스 코드는 매번 데이터베이스 자원을 생성하고 해제하기 위한 코드가 필요하지 않다.**

**- 파라미터를 설정하기 위한 코드가 거의 없다.**

**- 결과셋을 가져와서 값을 설정하는 코드가 거의 없다.**

**- JDBC는 SQL이 자바 코드 내의 문자열로 처리되는 반면, 마이바티스는 SQL 을 XML 이나 인터페이스 내의 annotation으로 별도 관리한다. 나중에 SQL 만 별도로 보고자 할 때 편리하다.**

**- 마이바티스를 구성하는 구성 요소는 5가지로, 설정 파일, 매퍼, 결과 매핑과 매핑 구문, 파라미터 타입,   결과 타입이다.**

**1> 마이바티스 파라미터 표기법**

마이바티스에서는 파라미터를 설정하기 위한 다음과 같은 독특한 표기법을 사용한다.

**#{commentNo}** 이것이 마이바티스에서 사용하는 파라미터의 표기법이다.

추출한 SQL을 XML에 명시하는 경우 값을 설정할 때 어떻게 해야 할까? SQL을 XML로 분리했기 때문에 JDBC코드처럼 값을 설정할 방법이 필요하다. 따라서 마이바티스는 #{}형태의 별도 문법을 제공한다. 방법은 파라미터 이름을 그대로 적어주면 된다. 자바빈일 경우 변수명 또는 getter/setter 메소드에서 get/set을 뺀 나머지 문자를 소문자로 시작해서 적어 주면 된다. 마이바티스는 자바빈에 getter 메소드가 있으면 getter 메소드를 사용해 값을 가져오지만, getter 메소드가 없을 경우에는 자바의 리플렉션을 사용해 값을 가져온다.**Map 객체일 때는 key 값을 적어 주면 설정해준다.** 파라미터의 값이 한 개인 원시 타입의 경우 #{} 안에 아무 값이나 적어줘도 알아서 설정해준다.

**2> 데이터를 조회하는 매핑 구문**

**데이터를 조회하는 매핑 구문**

<?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">

**XML과 DOCTYPE 선언**

첫 번째 줄은 이 문서가 XML문서라는 것을 나타내며, 두 번째 줄 부터는 이 XML문서가사용하는 엘리먼트와 엘리먼트 간의 구조를 정의하는 DTD 선언이다.

<mapper namespace="myMapper">

**메퍼 네임스페이스**

**매퍼의 네임스페이스는 매핑 구문들의 그룹 정도로 이해하면 된다. 네임 스페이스의 명명에 있어 중복이 발생하면 에러가 발생하므로 네임스페이스와 매핑 구문의 아이디를 정할 때 일정한 규칙을 정한 후 처리하는 것이 좋다. 대게는 테이블명으로 네임스페이스를 정하고 매핑 구문 아이디는 SQL의 맥락에 따라 명명한다.**

    <select id="selectCommentByPrimaryKey" parameterType="long"

         resultType="ldg.mybatis.model.Comment">

         SELECT

            comment\_no AS commentNo,

            user\_id AS userId,

            comment\_content AS commentContent,

            reg\_date AS regDate

        FROM COMMENT

        WHERE comment\_no = #{commentNo}

    </select>

</mapper>

**매핑 구문**

매핑 구문은 JDBC 코드에서 가져온 SQL과 몇 가지 XML 속성으로 구성한다. **parameterType 속성에는 조회 조건에서 사용할 값을 가진 파라미터의 타입을 지정해 주면 된다. 파라미터는 대개 자바빈을 사용하거나 Map을 많이 사용한다. 자바빈을 사용하면 해당 클래스를 지정해주면 되고 Map일 때는 map을 적어주면 된다.**

**- mybatis 실습**

**1>pom.xml 파일**

**1-1. 오라클 library 추가 확인!!!**

**1-2. mybatis 라이브러리 추가 선언**

<!-- mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.6</version>

</dependency>

**2> db-config.xml**

**2-1.data 연결 설정**

<environments default=*" "*>

<environment id=*" "*>

</environment>

</environments>

**2-2.매퍼 파일 설정**

<mappers>

<mapper resource=*" "* />

</mappers>

**2-3.타입 alias 설정**

<typeAliases>

<typeAlias type=*"mybatis.EmpVO"* alias=*"emp"* />

</typeAliases>

**3> emp-mapper.xml**

**필요한 sql 설정**

**4> EmpVO 작성**

**5> EmpDAO 작성(SqlSession 객체 선언)**

**public** **class** EmpDAO {

SqlSession session;

**public** **void** setSession(SqlSession session) {

**this**.session = session;

}

**6>EmpService, EmpServiceImpl 작성**

**7> EmpMain 작성**

**7-1. SqlSessionFactoryBuilder 생성**

7-2. SqlSessionFactory 생성(각 config 파일마다 생성하므로 복수개 될 수도 있음)

SqlSessionFactory factory = builder.build

(

Resources.*getResourceAsReader*

("edu/kitri/mybatis/mybatis-config.xml")

);

**7-3.** SqlSession 생성(true :auto commit , 나머지 : 수동)

SqlSession session = factory.openSession(true);//auto

SqlSession session = factory.openSession(false);//x

SqlSession session = factory.openSession();//x

**7-4.**

**parameterType="int|String|EmpVO|int[]|java.util.List|java.util.Map"**

**<xxx id="" parameterType="int|String" >**

**#{a}==> 이름 아무거나**

**<xxx id="" parameterType="EmpVO"**

**#{EmpVO의 변수명:getter메소드로 호출 가능해야 함}**

**<xxx id="" parameterType="int[]"**

<foreach collection=*"array"* item=*"a"*

open=*"("* close=*")"* separator=*","*>

#{a}

</foreach>

**<xxx id="" parameterType="int[]"**

#{array[0]} #{array[1]} 의 형식도 가능

**<xxx id="" parameterType="java.util.List"**

<foreach collection=*"list"* item=*"a"*

open=*"("* close=*")"* separator=*","*>

#{a}

</foreach>

**<xxx id="" parameterType="java.util.Map"**

<foreach collection=*"list"* item=*"a"*

open=*"("* close=*")"* separator=*","*>

#{a}

</foreach>

#Map의 키값

**session.xxxx("id값", 100);**

**session.xxxx("id값", "java");**

**session.xxxx("id값", empvo);**

**-스프링 + mybatis 연동**

**1>pom.xml: 2개 추가**

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.3.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>4.3.18.RELEASE</version>

</dependency>

**2>mybatis/mybatis-config.xml 복사하여**

**mybatis-spring-config.xml 로 이름 변경**

**3> 아래 태그만 남기고 나머지 주석처리**

<typeAliases>

<typeAlias

type=*"mybatis.EmpVO"* alias=*"emp"* />

<typeAlias

type=*"mybatis.EmpVO2"* alias=*"emp2"* />

</typeAliases>

**4> web-inf/spring 폴더에 mybatis-spring.xml 파일 생성**

<!-- 1.데이터소스(데이터연결con)설정 -->

<bean id=*"dataSource"*

class=

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

<property name=*"driverClassName"*

value=*"oracle.jdbc.driver.OracleDriver"* />

<property name=*"url"*

value=*"jdbc:oracle:thin:@localhost:1521:xe"* />

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

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

</bean>

<!-- 2.SqlSessionFactoryBean : db-config.xml,

sql-mapping.xml

board-mapping.xml

member-mapping.xml

xxx-mapping.xml

-->

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

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

<property name=*"configLocation"*

value=*"classpath:/mybatis/mybatis-spring-config.xml"* />

<property name=*"mapperLocations"*

value=*"classpath:/mybatis/\*-mapping.xml"* />

</bean>

<!-- 3.SqlSessionTemplate api :

sqlSessionFactory 객체 생성자에 전달-->

<bean id=*"sqlSession"*

class=*"org.mybatis.spring.SqlSessionTemplate"*>

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

</bean>

**5>servlet-context.xml 파일에 아래 부분 추가**

<!-- 4.@annotation -->

<context:component-scan base-package=*"mybatis"* />

**6> web.xml 파일에 아래 부분 수정 추가**

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

classpath:spring/tvspring.xml

/WEB-INF/spring/mybatis-spring.xml

</param-value>

<!-- <param-value>/WEB-INF/spring/root-context.xml</param-value> -->

</context-param>

**7> mybatis.MybatisController.java**

@Controller

**public** **class** MybatisController {

@Autowired

EmpService service ;

@RequestMapping("/getall\_emp2")

**public** ModelAndView getAllEmp2() {

**8> EmpDAO 수정**

@Component

**public** **class** EmpDAO {

@Autowired

SqlSession session;

// public void setSession(SqlSession session) {

// this.session = session;

// }

**9> EmpServiceImpl 수정**

@Service

**public** **class** EmpServiceImpl **implements** EmpService {

@Autowired

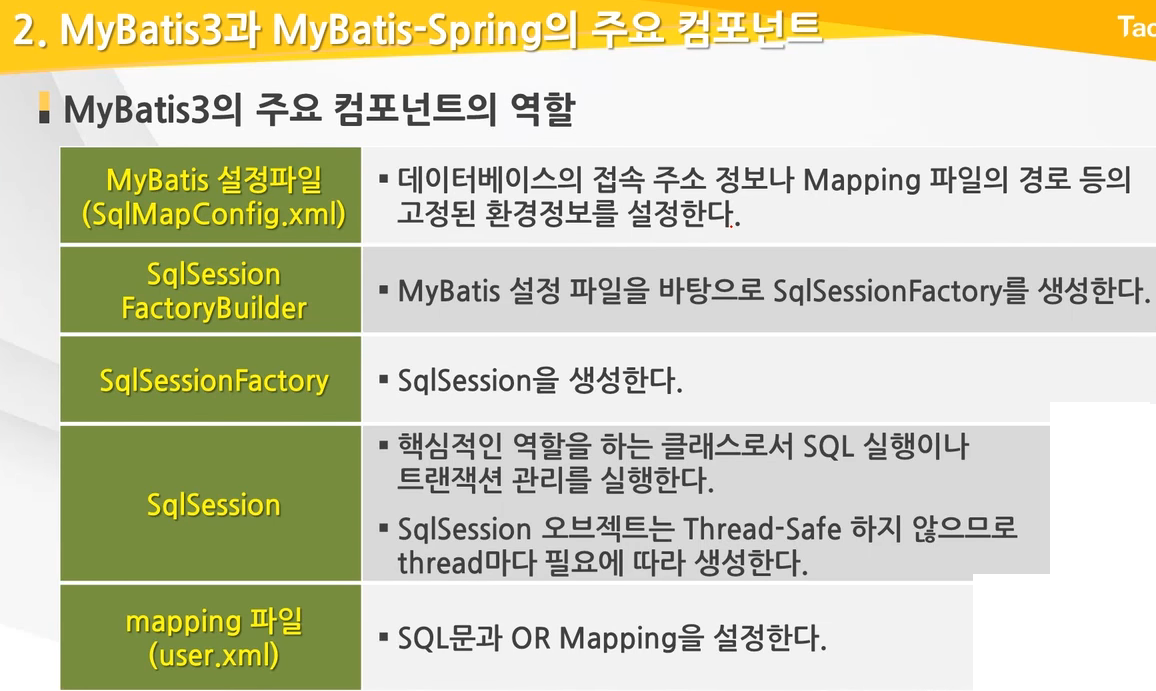
EmpDAO dao ;

// public void setDao(EmpDAO dao) {

// this.dao = dao;

// }

**10>/mybatis/getall\_emp2.jsp**



**조회 select**

**DDL: DATA DEFINITION LANG.(데이터정의언어)**

**CREATE TABLE / ALTER / DROP**

**(자동 반영)**

**DML : DATA MANIPULATAION LANG(데이터조작언어)**

**테이블 데이터 저장 / 수정 / 삭제**

**INSERT UPDATE DELETE**

**(자동 반영X)**

**COMMIT / ROLLBACK**

**DCL**

**TCL : COMMIT / ROLLBACK**

**TRANSACTION CONTROL LANG.**

**sql**

**- employees 테이블 입사일(hire\_date) 늦은 사원부터 순서대로 조회**

**select first\_name, hire\_date**

**from employees**

**order by hire\_date desc**

**(내림차순); 큰(2020..)-->작(01...)**

**order by hire\_date desc(내림차순); 큰(2020..)-->작(01...)**

**order by hire\_date [asc] : 작 ->큰**

**- employees 테이블에서 11-20번째 데이터 조회**

**select rownum, first\_name, hire\_date**

**from employess**

**where rownum >= 11 and rownum <= 20;**

**- rownum 함수 : 조회레코드번호**

**>, >= 연산 못한다 (단 1 제외)**

**1 길동 20/02/07**

**2 .....**

???????

**- employees 테이블에서 입사일 최근순서부터 11-20번째 데이터 조회**

**select r, first\_name, hire\_date**

**from (select rownum r, first\_name, hire\_date**

**from**

**(select \* from employees order by hire\_date desc)**

**)**

**where r >= *? and r <= ?***

**1페이지 입력 : 1-10 10명 사원**

**int page = 2;**

**int start = (page-1) \* 10 +1; --> 11**

**int end = page \* 10; -> 20**

**2페이지 입력 : 11-20 10명 사원**

**- mybatis.EmpVO—employees crud-main**

**- 웹서버, 웹클라이언트 x**

**- spring mvc(cmv) +spring orm (다른 db 프레임워크연동)+mybaTIS 연동**

**obejct – relatinal db – mapping**

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

**- 1. 라이브러리 다운로드**

**pom.xml : <dependency > ... : mybatis.jar**

**oracle jdbc 드라이버 : ojdbc6.jar**

**- 2. db 연결정보설정.xml**

**jdbc driver, url, id, pw===> SqlSession api**

**3. sql mapping .xml**

**<select id="" resultType="int"**

**select count(\*) from employees**

**108**

**1 2 3 4 5 ... 11**

**<select id="" resultType="String"**

**select upper('java') from dual**

**<select id="" resultType="xxxVO" parameterType="int[]"**

**<insert id=""**

**<delete id=""**

**<update id=""**

**3. XXXVO – (MYBATIS : sql mapper 프레임워크)**

**sql 결과(레코드)--- EmpVO 매핑**

**4. XXXDAO : mybatis 코드 작성**

**5. main**

**===> spring mvc 연동**

**- 1. 라이브러리 다운로드**

**pom.xml : <dependency > ... : mybatis.jar**

**spring-jdbc.jar(orm기능)**

**mybatis-spring.jar(orm기능)**

**oracle jdbc 드라이버 : ojdbc6.jar**

**1. java main- java application**

**프로젝트-마우스 우클릭-properties – java build path-Libraries-add External jars**

**- 경로 선택 (odbc6.jar)**

**2. servlet, jsp, spring mvc – java web server**

**( 탐색기 - tomcat경로\lib\ojdbc6.jar붙여넣기)**

**프로젝트\mian\webapp\web-inf\lib(생성)\ojdbc6.jar붙여넣기**

**- 3. db 연결정보설정.xml**

**typeAlias 제외하고 삭제**

**4. sql mapping .xml(변동X)**

**3. XXXVO – (MYBATIS : sql mapper 프레임워크)**

**sql 결과(레코드)--- EmpVO 매핑**

**4. XXXDAO : mybatis 코드 작성**

**5. main 삭제**

**6. mybatis-spring연동설정파일(spring bean 설정파일)**

**-새로 정의**

**(SRC/main/webapp/web-inf/spring/mybatis-spring.xml)**

**6-1.MYBATIS – jdbc driver, url, username, pw – DataSource 생성( connection pool - ...)**

**6-2.mapper 파일 이름**

**6-3. typealias 이름**

**6-4. SqlSessionTemplate bean 생성**

**JDBC : ConnectionPool = spring DataSource**

**jdbc : Connection = spring SqlSessionTemplate = mybatis SqlSession**

**6. CONTROLLER – MODEL –**

**VIEW**

**/mybatis/emplist**

**@Contoller**

**Controller**

**@Autowired**

**EmpDAO dao;**

**@RequestMapping**

**("/empllist")**

**a()**

**web.xml**

**mybatis-spring.xml**

**servlet-context.xml**

**스프링 라이브러리**

**DispatcherServlet**

**EmpVO**

employees

**@Repository**

**EmpDAO**

**mybatis-config.xml**

**emp-mapping.xml**

**마이바티스 라이브러리**

**mybatis/emplist.jsp**

**edu.multi.mvc: /emplist : EmpDAO**

**edu.multi.mybatis: /emplist : EmpDAO**

*/mybatis/detailemp* **요청**

**요청한 사번의 사원 모든 정보 리턴받아 응답**

**1. 컨트롤러 메소드(리턴타입, 매개변수)**

**2. 모델**

**3. 뷰 : WEB-INF/views/mybatis/detailemp.jsp**

**사번 이름과 성 입사일 부서코드 이메일 전화번호**

**직종코드 급여**

[**http://70.12.60.90:9090/spring/**](http://70.12.60.90:9090/spring/)

**==>mybatis\_spring연동.zip 공유**

**게시판관리프로그램 crud**

**회원관리프로그램 crud(회원가입-로그인/로그아웃)**

**-로그인/로그아웃**

**MEMBER 테이블 회원가입 가정**

**mybatis/login.jsp-로그인폼 뷰**

**mybatis/loginresult.jsp-id, pw – db 조회 결과**

**홈페이지 최초 1번 로그인 – 홈페이지 다른 화면 x**

**ModelAndView mv..**

**mv.addObject("", 객체)===> 1개 view**

**===> 1개 요청 처리 동안 (c m v) 공유**

**여러개 view 1번 로그인 정보 지속 공유**

**(브라우저 종료 이전 / 브라우저 실행 30분동안 무액션 )**

**javax.servlet.http.HttpSession**

**1. HttpSession 생성**

**HttpSession session = request.getSession();**

**2. HttpSession 공유 정보 저장**

**session.setAttribute("세션값이름", 객체)**

**session.setAttribute("세션값이름2", 객체2)**

**3. HttpSession 저장 정보 추출**

**???<-(형변환)session.getAttribute("세션값이름")**

**4. HttpSession 저장 정보 삭제**

**session.removeAttribute("세션값이름")**

**5. HttpSession 소멸**

**session.invalidate();**

**LoginSessionController**

**MemberVO**

**MemberDAO**

**-프로젝트 / 스프링시험 / 1,2,3 정리**

**4시**

**60점 이하**

**20문제**

**10문제 화면**

**- 프로젝트**

**2월 17일까지 종일 구현**