**기본이 DBconnectin이후 작업.**

**한글 인코딩. (web.xml)**

<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>

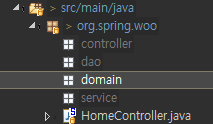
</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

domin,dao,controller,service로 패키지를 나눈다.

테이블 만들기.

create table tbl\_board(

bno int PRIMARY key,

title VARCHAR(200),

content varchar(200),

insertDate varchar(100)

)

board VO 객체 만들기(domain 객체.)getter,setter,toString (alt+shift+s)

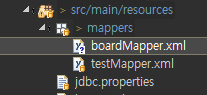
private int bno;

private String title;

private String content;

private String insertDate;

mapper 생성 후. 작업.(mybatis 기본 틀.)



<?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="board">

</mapper>

CRUD mapper작업.(postgresSQL을 사용하기 때문에 구문이 다를 수 있음.)항상 DB에 쿼리를 날린후 Mybatis에 넣어준다. ;제외시키고

<insert id="insert" parameterType="BoardVO">

insert into

tbl\_board(bno,title,content,insertDate)

values(nextval('seq\_board'),#{title},#{content},now())

</insert>

<select id="read" resultType="BoardVO">

select \* from tbl\_board WHERE BNO = #{bno}

</select>

<update id="update" parameterType="BoardVO">

UPDATE tbl\_board

SET TITLE = #{title},

CONTENT = #{content}

WHERE BNO = #{bno}

</update>

<delete id="delete">

DELETE FROM tbl\_board WHERE BNO = #{bno}

</delete>

parameterType은 VO의 객체를 가져다 쓴다.(이름 똑같이 해줘야 한다.)

resultType은 결과를 VO에 넣어준다.

mybatis-config.xml에 설정을 해줘서 BoardVO만 써주지만 그렇지 않을경우 org.spring.woo.domin.BoardVO를 직접 써줘야 한다.

dao설정(dao란 DB를 직접적으로 연결하는 부분인다.)

-interface

public interface BoardDAO {

//BoardVO를 파라미터로 넘긴다.

public void insert(BoardVO vo) throws Exception;

//int bno를 파라미터로 받고 VO로 저장한다.

public BoardVO read(int bno) throws Exception;

public void update(BoardVO vo) throws Exception;

public void delete(int bno) throws Exception;

}

-daoImpl(interface 상속)

Repository를 걸어주지 않으면 DAO랑 연결이 되지 않는다.

(servlet-context.xml에 밑처럼 Repository Service두개를 설정해주는거다.)

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

@Repository

public class BoardDAOImpl implements BoardDAO{

//sql factory 가져오기 inject awtowired전부 가능

@Inject

private SqlSession session;

String name = "board.";

@Override

public void insert(BoardVO vo) throws Exception {

//등록기능 mapper의 경로를 써준다.

session.insert(name+"insert",vo);

}

@Override

public BoardVO read(int bno) throws Exception {

//select 구문중 1개의 값만 있을때 사용.

return session.selectOne(name+"read",bno);

}

@Override

public void update(BoardVO vo) throws Exception {

session.update(name+"update",vo);

}

@Override

public void delete(int bno) throws Exception {

session.delete(name+"delete",bno);

}

}

-service만들기 (dao랑 내용은 같다.)

dao의 역할(DB 연결) service의 역할 (DB와 controller의 중간 데이터 가공.)지금은 가공할 일이 없다.

public interface BoardService {

public void insert(BoardVO vo) throws Exception;

public BoardVO read(int bno) throws Exception;

public void update(BoardVO vo) throws Exception;

public void delete(int bno) throws Exception;

}

daoImpl (dao 상속)

@Service를 써줘야 연결이 된다. DAO에 연결 시켜준다.

@Service

public class BoardServiceImpl implements BoardService{

@Inject

private BoardDAO dao;

@Override

public void insert(BoardVO vo) throws Exception {

dao.insert(vo);

}

@Override

public BoardVO read(int bno) throws Exception {

return dao.read(bno);

}

@Override

public void update(BoardVO vo) throws Exception {

dao.update(vo);

}

@Override

public void delete(int bno) throws Exception {

dao.delete(bno);

}

}

-controller

jsp와 java의 연동.

get 방식은 단순 페이지. Post방식은 service(기능)가 들어가는 방식.

@Controller //컨트롤러 어노테이션

@RequestMapping("/board/\*") //경로 설정 메소드 (board밑으로 전부 경로 설정된다.)

public class BoardController {

//logger.info사용 (sysout같은거다.{기본 컨트롤러에 이미 되어 있다.})

private static final Logger logger = LoggerFactory.getLogger(BoardController.class);

//service 가져오기

@Inject

private BoardService service;

//insertPage설정.(view.board.insertPage로 경로가 설정 되어 있다.)

//web.xmp에서 확인 가능.

@RequestMapping(value = "/insertPage", method = RequestMethod.GET)

public void insertPage() throws Exception {

}

//insert 기능 redirect는 이 경로로 이동하게 된다.

@RequestMapping(value = "/insert", method = RequestMethod.POST)

public String insert(Model model, BoardVO vo) throws Exception {

service.insert(vo);

return "redirect:/board/insertPage";

}

//read기능 bno를 파라미터로 가져와야 한다.

@RequestMapping(value = "/read", method = RequestMethod.GET)

public void readPage(Model model, @RequestParam("bno") int bno) throws Exception {

model.addAttribute("vo", service.read(bno));

}

@RequestMapping(value = "/updatePage", method = RequestMethod.GET)

public void updatePage(Model model, Integer bno) throws Exception {

model.addAttribute("vo", service.read(bno));

}

@RequestMapping(value = "/delete", method = RequestMethod.POST)

public String delete(Model model, int bno) throws Exception {

service.delete(bno);

return "redirect:/board/insertPage";

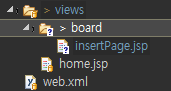
}

}

jsp

아래와 같이 경로와 이름을 맞춰주면 해당 jsp로 연결이 된다.

url = localhost/board/inserPage





jsp는 항상 UTF-8로 바꿔준다.

jsp에서 controller로 url날리는 다양한 방법.

1. form태그의 action을 이용한다.

(action을 이용해서 url을 넣어주면 name의 값들이 해당 url로 이동하게 된다.)

1. a태그를 이용한 경로 이동. url: localhost/board/insertPage에서

localhost/board/insertPage?title = “a” &content = “b”

라는 식으로 파라미터를 날려 줄수 있다 대신 해당 자료를 날리게 되면 @RequestParam으로 상대 파라미터를 받아 줘야 한다.

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>등록</title>

</head>

<body>

<form action="/board/insert" id="form" method="post">

<table border="1">

<tr>

<th>제목 </th>

<th>내용</th>

</tr>

<tr>

<td><input type="text" name="title" id="title"></td>

<td><input type="text" name="content" id="content"></td>

</tr>

</table>

<button type="submit">등록</button>

</form>

</body>

</html>