public class HomeController

“redirect:/board/boardlist”를 반환한다. 즉 root가 /board/boardlist이다.

public class SerachVO

private String fd; // 검색 조건

private String kw; // 검색어

private String page; // 페이지

private int ipage; // 형변환 페이지

private int startpage; // 시작페이지

private int lastpage; // 마지막페이지

public class BoardDao

public List<HashMap<String, Object>> getBoardList(SearchVO param)

board.xml 에서 boardList id를 찾아 SQL 구문을 실행한 후 HashMap 타입으로 반환한다.

public int getBoardCount(SearchVO param)

board.xml 에서 boardCount id를 찾아 SQL 구문을 실행한 후 Integer 타입으로 반환한다.

public HashMap<String, Object> getBoardDetail(int param)

board.xml 에서 boardDetail id를 찾아 SQL 구문을 실행한 후 HashMap 타입으로 반환한다.

public void regeditContent(HashMap<String, Object> param)

board.xml 에서 regeditContent id를 찾아 SQL 구문을 실행한 후 HashMap 타입으로 반환한다.

public int getMyContentNo(int param)

board.xml 에서 getMyContentNo id를 찾아 SQL 구문을 실행한 후 Integer 타입으로 반환한다.

public void updateBoard(HashMap<String, Object> param)

board.xml 에서 updateBoard id를 찾아 SQL 구문을 실행한 후 HashMap 타입으로 반환한다.

public class BoardServiceImpl implements BoardService

public List<HashMap<String, Object>> getBoardList(SearchVO param)

보드 목록을 조회하는 메서드이다.

SearchVO 에서 fd, kw가 비어있으면 각각 “title”, “”로 초기화 한다.

SearchVO 에서 page가 비어있으면 ipage를 1로 초기화한다.

page가 비어있지 않다면 ipage에 page값을 대입한다.

startpage = (ipage - 1) \* 10 연산을 수행한다.

이후 boardDao.getBoardList(param)을 호출 후 반환한다.

public int getBoardCount(SearchVO sv);

보드 건수를 조회하는 메서드이다.

SearchVO 에서 fd, kw가 비어있으면 각각 “title”, “”로 초기화 한다.

이후 boardDao.getBoardCount(param)을 호출 후 반환한다.

public HashMap<String, Object> getBoardDetail(String brdno);

글의 상세 페이지를 불러오는 메서드이다.

brdno를 매개변수로 입력받아 param에 대입한다.

이후 brddao.getBoardDetail(param)을 호출 후 반환한다.

public int regeditContent(String brdtitle, String brdmemo, HttpSession session);

글을 등록하는 메서드이다.

세션에서 “userInfo”를 HashMap<String, Object>형으로 형변환하여 map 변수에 대입한다.

map에서 “userno”키를 찾아, value를 userno에 대입한다.

HashMap<String, Object> param을 선언하고, userno, brdtitle, brdmemo를 put한다.

brddao.regeditContent(param) 를 호출한다.

brddao.getMyContentNo(userno) 를 호출하고, 반환값을 brdno에 대입한다.

brdno를 반환한다.

public void updateBoard(String brdtitle, String brdmemo, HttpSession session, String brdno);

글을 수정하는 메서드이다.

세션에서 “userInfo”를 HashMap<String, Object>형으로 형변환하여 map 변수에 대입한다.

map에서 “userno”키를 찾아, value를 userno에 대입한다.

HashMap<String, Object> param을 선언하고, userno, brdtitle, brdmemo, brdno를 put한다.

brddao.updateBoard(param) 를 호출한다.

public class BoardController

@GetMapping(“/boardlist”)

public String getBoardList(HttpServletRequest req, SearchVO sv, ModelMap modelmap, HttpSession session)

보드 목록을 조회하는 메서드이다.

brdsvc.getBoardCount(sv) 를 호출하여 반환값을 count에 대입한다.

brdsvc.getBoardList(sv) 를 호출하여 반환값을 blist에 대입한다.

modelmap에 blist, count, session을 put한다.

“board/boardlist”를 반환한다.

@GetMapping(“/boarddetail”)

public String getBoardDetail(HttpServletRequest req, ModelMap modelmap, @RequestParam("brdno")String brdno, HttpSession session)

brdsvc.getBoardDetail(brdno) 를 호출하여 반환값을 map에 대입한다.

map에 brdno를 put 한다.

modelmap에 session(key: “userInfo”), map을 addAttribute한다.

“board/boarddetail”을 반환한다.

@GetMapping(“/boardregedit”)

public String getBoardRegeditView(HttpServletRequest req)

“board/boardregedit”을 반환한다.

@PostMapping(“/boardregedit”)

public String getBoardRegedit(HttpServletRequest req, HttpSession session, @RequestParam("brdtitle") String brdtitle, @RequestParam("brdmemo") String brdmemo)

brdsvc.regeditContent(brdtitle, brdmemo, session)를 호출하여 brdno에 대입한다.

“redirect:/board/boarddetail?brdno=” + brdno를 반환한다.

@GetMapping(“/boardupdate”)

public String updateBoard(HttpServletRequest req, ModelMap modelmap, @RequestParam("brdno")String brdno)

brdsvc.getBoardDetail(brdno)를 호출하여 map에 대입한다.

modelmap에 map을 addAttribute한다.

“/board/boardupdate”를 반환한다.

@PostMapping(“/boardupdate”)

public String updateBoard(HttpServletRequest req, HttpSession session , @RequestParam("brdtitle") String brdtitle, @RequestParam("brdmemo") String brdmemo, @RequestParam("brdno") String brdno)

brdsvc.updateBoard(brdtitle, brdmemo, session, brdno)를 호출한다.

“redirect:/board/boarddetail?brdno=” + brdno를 반환한다.

boardlist.js

let signup = document.getElementById('btn-signup');

id가 “btn-signup”인 버튼

let login = document.getElementById('btn-login');

id가 “btn- login'”인 버튼

let logout = document.getElementById('btn-logout');

id가 “btn- logout'”인 버튼

let regedit = document.getElementById('btn-regedit');

id가 “btn- regedit'”인 버튼

signup.addEventListener('click', function() {

window.location.href = '/user/signup';

});

signup 버튼이 클릭되면 ‘user/signup 주소로 이동

login.addEventListener('click', function() {

window.location.href = '/user/login';

});

login 버튼이 클릭되면 ‘user/ login’ 주소로 이동

logout.addEventListener('click', function() {

window.location.href = '/user/logout';

});

logout 버튼이 클릭되면 ‘user/ logout' 주소로 이동

regedit.addEventListener('click', function() {

window.location.href = '/board/boardregedit';

});

regedit 버튼이 클릭되면 ‘user/ regedit’ 주소로 이동

boardregedit.js

function updateContent(brdno)

호출되면 '/board/boardupdate?brdno=' + brdno 주소로 이동

function backBoardList(page, field, keyword)

호출되면 ‘/board/boardlist?page=’ + page + ‘&fd=’ + field + ‘&kw=’ + keyword 주소로 이동

function paramCheck(brdTitle, brdMemo)

호출되면 brdTitle의 길이가 50초과이면 false, brdMemo의 길이가 500초과이면 false, 둘 다 아니면 true를 반환한다.

function formSubmit()

id가 ‘regForm’, ‘brdtitle’, brdmemo’인 요소를 각 변수에 할당한다.

paramCheck(brdTitle, brdMemo)를 호출하여 반환값이 true이면 regFrom을 submit한다.

public class UserLoginVO

private String id;

private String pass;

public class UserSignupVO

private String id;

private String pass;

private String name;

public class UserDao

public int LoginCheck(UserLoginVO param)

user.xml 에서 LoginCheck id를 찾아 SQL 구문을 실행한 후 Integer 타입으로 반환한다.

public HashMap<String, Object> getUserInfo(UserLoginVO param)

user.xml 에서 userInfo id를 찾아 SQL 구문을 실행한 후 HashMap 타입으로 반환한다.

public void signUp(UserSingupVO param)

user.xml 에서 signup id를 찾아 SQL 구문을 실행한다.

public class UserServiceImpl implements UserService

public int LoginCheck(UserLoginVO param)

userdao.LoginCheck(param) 를 호출하여 반환한다.

public HashMap<String, Object> getUserInfo(UserLoginVO param)

userdao.getUserInfo(param) 를 호출하여 반환한다.

public void signUp(UserSingupVO param)

userdao.signUp(param) 를 호출한다.

public class UserController

@GetMapping("login")

public String login(HttpServletRequest req, ModelMap modelMap)

“user/UserLogin”을 반환한다.

@PostMapping("login")

public String loginCheck(HttpServletRequest req, ModelMap modelMap, UserLoginVO param)

usrSvc.LoginCheck(param)를 호출하여 반환값을 result에 대입한다.

result가 0이 아니면 // 로그인 성공

usrSvc.getUserInfo(param)를 호출하여 userInfo에 대입한다.

userInfo를 세션에 setAttribute한다.

“redirect:/board/boardlist”를 반환한다.

result가 0이면 // 로그인 실패

“redirect:/user/login”을 반환한다.

@GetMapping("logout")

public String logout(HttpServletRequest req)

세션을 종료한다.

“redirect:/user/login”을 반환한다.

@GetMapping("signup")

public String signup(HttpServletRequest req, ModelMap mp)

"user/UserSignUp"을 반환한다.

@PostMapping("signup")

public String signup1(@RequestParam("sid") String sid,

@RequestParam("sname") String sname,

@RequestParam("spass") String spass,

@RequestParam("srepass") String srepass,

ModelMap mp,

UserSingupVO param)

sid, spass, sname을 param에 담아 signUp(param)을 호출 후 “user/UserLogin”을 반환한다.

signup.js

function nameCheck()

UserSignUp.jsp에서 id가 sname인 객체를 찾아 값을 sname에 대입한다.

sname === “” or 길이가 0이하이면 “이름을 입력해주세요.”를 출력하고 false를 반환한다.

길이가 20초과이면 “이름을 20글자 이내로 입력해주세요.”를 출력하고 false를 반환한다.

위 두 조건이 맞지 않으면 true를 반환한다.

function passCheck()

UserSignUp.jsp에서 id가 spass인 객체를 찾아 값을 spass에 대입한다.

정규식을 사용하여 spass가 8자리 이상, 숫자와 특수문자가 포함되어 있지 않으면 "8자리 이상, 숫자와 특수문자를 사용한 비밀번호만 사용 가능합니다."를 출력하고 false를 반환한다.

포함되어 있으면 message를 “”로 초기화 하고 true를 반환한다.

function repassCheck()

UserSignUp.jsp에서 id가 spass, srepass인 객체를 찾아 값을 각각 spass, srepass에 대입한다.

spass === srepass이면 message를 “”로 초기화 하고 true를 반환한다.

조건이 틀리면 "비밀번호를 다시 확인해주세요."를 출력하고 false를 반환한다.

function valCheck()

UserSignUp.jsp에서 id가 signForm, sid, sname, spass인 객체를 찾아 값을 각각 signForm, sid, sname, spass에 대입한다.

nameCheck(), passCheck(), repassCheck()를 호출하여 모두 true이면 signForm을 submit 한다.