//190114

/포워딩은 서버측에 저장. 세션은 껏다 키면 사라짐. 몇 시간 후 자동 삭제. 사용자가 입력한

정보를 받아 와서, 서버 컴퓨터에 저장해 놓고서는, 클라이언트가 페이지 요청할 때마다,

그 저장해 놓은 정보를 활용해서 ( ${infom} 이런 거에 값 채워서 ) 값이 채워진 채로 보내 줌.

/쿠키 - 클라이언트 쪽에 파일로 남겨 둠.

/requestScope , sessionScope 등은, 브라우저 측에서 미리 만들어놓은 객체, 즉, 변수.

그러니까 시작이 소문자로 시작이지.

+)미리 객체를 만들어 놓고, 필요할 때 객체 받아 와서 실행.

/주소를 다르게 해도 상관은 없다만, 똑같은 주소에 대해 메소드만 다르게 해서

오버로딩 할 수 있음. 즉, RequestMapping의 value 값이 같더라도,

method 값이 get으로 요청 들어오는 것과 post로 오는 것을 구별한 후

알아서 나눠서 메소드 실행해 줌.

/script에서 id.focus랑 id.select 있으면, 커서가 가있거나 블록이 씌워져 있음

/처음 온 건지, 입력한 정보에 오류가 있어서 팅겼다가 포워딩 돼서 온 건지, 확인하려면

+) 걍 포워딩 되는 과정에서 컨트롤러에서 flag 심어주고, 로그인 페이지에서는 flag있나 확인

+) 응용 - 쫓겨나는 이유마다 다른 flag 세워주면 더 상세하게 가능

+) script에 메소드 없이 쓰면 시작하자마자 실행 됨. 이 때 flag 상태에 따라 다르게 실행 가능.

/페이지는 하난데, jstl 이용해서, 상태에 따라 다르게 뜸. 로그인 상태에 따라, 뭐에 따라...

/페이지는 안 바뀌면서 부분적으로 서버 다녀오는 건 ajax

//190115

/가독성

+) 패러미터는 한 줄씩, 콤마는 맨 앞

+) 주석: 접근자가 public인 것들. 그 중 class 위에. 뭐하는 클래스인지, 누가 언제 만든 건지

-)이런 것들은 뭔가 논문이랑 똑같네! 공적 영역에서는 다들 이런 게 중요한 요소인 듯

+) 클래스 아래. 멤버 변수, 클래스 변수의 역할

+) 메소드 앞에는 각 패러미터에 대한 설명. 어떤 정보가 담겨서 넘어오는 건지.

return 값은 뭐가 나가는 건지.

/index 문서

+) 이클립스 상단 옵션 project - Generate Javadoc

+) 문서 생성할 프로젝트 선택, 주석 포함할 범위 선택.

+)VM options => -encoding UTF-8

/쿠키

+) 객체 받아오는 거는 세션이랑 똑같아. 이미 생성돼 있어. 따라서 패러미터에 변수명만 정해서 넣어주면 됨. -) HttpServletResponse

+) 쿠키는 hashmap처럼 사용. Cookie 라는 객체 생성해서, 호출자와 데이터값을 짝으로 넣어 줌.

+) 쿠키는 데이터에 제약이 있음. 문자열(String)만/크기는 몇 백 바이트 정도/언어는 영어, 숫자만

+) 이렇게 만들고 나면 혼자 간직만하다가 버림. 보내 줘야 함. 그게 response

+) 없어져도 크게 문제될 것 없는 일시적 정보들, 여기에 저장

+) 생성할 때는 response지만, 읽어올 때는 request (모든 요청은 얘가 다 관리)

+) 클라이언트의 아이피 주소 받아오기 => HttpServletRequest 객체.getRemoteAddr(); 하면

+) 브라우저(크롬)에서 설정 -> "쿠키" 검색 -> 콘텐츠 설정 -> 쿠키 -> 모든 쿠키 및 사이트 데이터 보기 -> 암거나 클릭 -> 쿠키 이름 클릭 -> 콘텐츠: 거기 저장된 정보임.

+) 쿠키 삭제는 없음. 꼼수를 통해서 삭제하는 기능을 수행하게끔은 할 수 있는데,

똑같은 이름으로 덮어쓰면서 수명(지속시간)을 매우 짧게 만들어 놓으면 됨.

//190116

/스프링은 단순 노가다를 줄여주는 여러 기능을 제공. 스프링 프레임웍에서 자체 지원하는 여러 기능 중 하나가 있는데, 쿠키를 일일이 꺼내서 검색할 필요 없이, 알아서 해줌.

+)우리가 넣어놓은 여러 쿠키 중에, "test"라는 녀석을 찾으려면, 있는 쿠키들 다 꺼내와서,

for문 돌리면서, 이름이 "test" 인 게 있는지 다 확인해봐야 함.

+) @CookieValue 어노테이션 쓰면, 이런 작업을 알아서 한번에 수행해 줌.

request 객체 통해서 하나하나 찾아서 읽어 보고, 혹시 없으면 defaultValue를 가져 옴.

+) default가 중요, 없으면 null, 밑에서 nullpointerexception 뜨고 난리 남

+) default에도 어노테이션 가능? 불가능. 그렇게 할 거면, 일단 "test2"는 패러미터에서 @CookieValue로 받아 오고, 밑에서 자바 코딩으로 처리하기. 만약 test2가 defaultValue를 받아 왔다면, test1의 값을 넣어 주는 코딩

/Cookie는 String 밖에 못 넣음. 변환 과정을 Integer 객체와 parseInt, toString 등의 메소드 필요

-) 귀찮으니까, String x = "" + int 하면 저절로 string으로 변환 활용.

/쿠키 읽어 오거나, 갱신하거나, 생성하거나, 다시 심어주는 작업들과,

모델이나 세션을 통해서 브라우저까지 보내주는 작업은, 서로 별개임.

/ <% %> 이 안에서는 자바 코딩 사용 사능. 두 쌍 사이에 html 태그를 넣을 수도 있음. 결국엔 println 처럼 출력값을 넣어주는 것 뿐이니까. / <%= %> 자바의 변수나 수식을 통해서 출력만 담당.

/xml에는 있는 정보가, 실제로 패키지나 클래스로는 없다? 오류 내버림

/밑에서 무슨 jar 받을지 적어 주고, 위에ㅓ 레퍼런스 주소, repository 주소, 즉 어디서 받는지도 알려주고

/bean - 미리 객체 만들어 놓기. 처음에 켤 때 읽어 오기 + 객체 생성 미리 해놓기 => 다른 데서 객체 생성 계속 안 하고 ,하나 만들어 놓은 걸로 다같이 불르면 가져오는 걸로 씀

+) <bean id = "a" class = "B"> 이러면, B a = new B(); 이렇게 말하는 거임.

+) 자바에서 @Autowired B a; 하면

+) 시작하면 @Controller 다 찾아서 그 안에 @Autowired 다 찾아서 대입해 놓고 시작함.

jsp / 폼 / 테이블

//190117

/DB에서 칼럼의 이름들과, 자바에서 VO 클래스의 변수 이름들이, 모두 각각 같으면 좋다.

+) getter setter 이용해서 값을 불러오거나 저장하기 때문에, 같으면 자동으로 되고,

다르면 따로 또 만져줘야 하니까 번거로움.

/기본 생성자 만든 후에 getter setter 작업 들어가니까, 일단 기본 생성자는 무조건 있어야 함.

/다 완성되고 실행해보는 것보다, 한 요소 추가할 떄마다 시범 확인. 계속 중간중간 켜 보기.

/확인 리스트

+) pox.xml 에서 자바와 스프링 버전

+) ojdbc6.jar 이게 있어야 오라클 접속 가능. repositories 는 다울 받을 위치. 주소만 알려주는 거.

+) 스프링에 관한 건 자동으로 들어 있을 거고

+) Servlet이랑 JSP 버전을 최신 버전으로 맞춰 주기. 굳이 필요는 없지만 혹시나

+) mybatis / mybatis-spring / ojdbc6 / spring-jdbc / +a 파일 업로드, 다운로드

+) web.xml 은 모든 웹 프로젝트라면 다 필요한 거임. 스프링이든 뭐든.

+) servlet-context.xml 이거는 익스플로러에서 보는 주소랑, 실제 서버의 주소랑, 단축 경로

-)그렇다해도 이거는 보안이랑은 상관은 없는 게, 어차피 공개되는 자료라

+) basepackage 중요. 왜 자꾸 다 없대 -> 이 베이스 패키지 밑에 있어야 읽어 옴.

어노테이션 붙은 거라든가, 미리 해오는 작업들을 다 이 아래에 있는 것만 가져옴. 중요.

+) root-context.xml 여기서 <bean> 태그는 객체 미리 만들어 놓는 거. id는 객체 변수 이름

-)new 클래스이름 해서 인스턴스 생성해서 우리가 적어 놓은 변수 이름에 넣어 줌

-) 나중에 쓸 때는 @Autowired 어노테이션으로.

(단, @Controller 어노테이션 있는 곳에서만 사용 가능.

+) db.properties 에서는 공백허용 X 특히 맨 끝에 스페이스바 누르고 엔터치면 안 됨

+) mybatis-config.xml 에서 <typeAliases> 는 단축어. <mappers>는 맵퍼 파일 어딨는지 알려줌.

+) Mapper.xml 에서는 위에 <mapper namespace = "~~클래스 이름~~"> 명확히 써 줘야,

그 인터페이스에다가 우리가 미리 짜 놓은 sql 쿼리를 자바 언어로 번역해서 채워놓아 줌.

-) parameterType 등은 Aliases 로 미리 등록해 놓은 대로.

-) sql 쿼리 문의 id 는 메소드 이름과 같아야

-) insert / update / delete 등은 returnType 이 기본으로 int. 몇 개 수정 됐는지

//190118

/ <c:if test = "${ str1.equals('str2') }"> jsp에서는 문자열 검색할 때 이렇게.

/el (expression language) 에서는 중활고 안에 자바의 수식이 들어가도 됨.

-) { str1 == null } / {str2 != null }

/ <c:if test = "~~"> 이 때, test 안에는 "true" 아니면 "false" 이 둘만 들어갈 수 있어. 그런데,

우리가 그 안에 EL, 즉 ${ ~~ } 을 쓸 수 있는 이유는, EL 이 자동으로 true or false로 변환해주는

기능이 있기 때문이야. 다시 말하면, ${ ~~ } 안에 수식이 들어 있을 경우, 그 결과를 true or false로 자동 변환해서 출력해주는 기능을 수행해주기 때문에. test = "~~~" 안이 저절로 true or false로 채워진다는 거지.