58일차 강의

* requestDispatcher란 ?

현재 request에 담고 있는 정보를 저장하고 있다가 그 다음페이지, 다음페이지에서도

해당 정보를 볼 수 있게 저장하는 기능을한다.

request -> response의 단계에서 forward와 sendRedirect방식이 있는데,

forward방식에서는 A.jsp -> servlet -> B.jsp까지 파라미터정보가 넘어가지만

sendRedirect방식에서는 정보를 제외한 단순히 페이지호출만 한다는 차이가 있다.

forward만 하게되면 따로 처리를 해주지 않을 시 B다음의 단계에서는 A의 파라미터가 소실된다.

이떄 requestDispatcher는 정보유지를 위해 사용된다.

즉, request에 n이라는 이름의 attribute를 저장 후 forward 시킨다 라는 말이다

[ 파일 형상 관리 ]

* cvs -> svn -> git

훈련일지 매뉴얼 ver1

훈련일지 매뉴얼 ver2

훈련일지 매뉴얼 ver3

**[ 이클립스를 사용하여 깃에 올려보자!!! ]**

* 내가 작업한 프로젝트를 깃에 올릴려면 javaEE에서 project Expoloer의 내가 작업한 프로젝트에서 시작한다]
* 프로젝트명에서 마우스 우클릭 -> team
* Team -> share 를 클릭
* 선택사항들을 선택한다

( 어느 레파지토리에 올릴것인지)

* Add to index
* Git Staging 창이 뜬다

(Git Staging 창이 안뜨면 team -> commit 클릭)

* 인덱스를 관리한다
* 커밋 메시지를 작성한다
* 커밋은 (오류가 없고 완벽할 때 하는 것이다)
  + - Brench 는 좀더 다른 것이다 (따로 공부할 것)
* 커밋 한다
* 프로젝트명 – team – show in history

: add to index 한것만 역사로 남아있는 것

* Team – pull

: 다른 사람들이랑 협업할 때 pull 하여,

다른사람과 인덱스를 새로 맞춰줘야 한다는 것

* Team -> synchronize workspace

: 충돌이 있는지 없는지 확인한다

* Team -> push branch master -> preview -> push
* 깃허브 사이트에서 내가 push한 프로젝트가 있는지 확인한다
* **[ git으로 하는 협업 ]**
  + branch master

– readme.md 파일을 만들면 자동생성되나 자동생성 되지않으면 별도로 만들어줘야함

* + contributor
    - 레파지토리의 주인이 승인할때마다 contributor가 추가되거나 기여할수 있는 기회를 준다
    - Fork 해서
  + Settings
    - **Collaborators** 
      * 함께 콜라보레이트 하고 싶은 사람의 아이다나 이메일 주소로 인비테이션을 보낸다
      * 초대를 받은 사람은
      * 함께 협력할 사람들을 승인 할수 있다
      * 승인된 사람은 함께 레파지토리를 관리할수 있다
* 이클립스에서 충돌이 생겼을 때 확인해야할 사항들?
  + **<<<<<<< HEAD**

: 헤드 위의 내용은 깃허브내용과 내가 작성한 내용 동일

* + 내가 작성한 영역 표시 헤드 아래는

**<<<<<<< HEAD**

이 사이 공간은 내가 작성한 내용

**=======**

* + 다른 사람이 작성한 영역 표시

**=======**

이 공간은 다른 사람이 작성한 내용

**>>>>>>> branch 'master' of** [**https://github.com/psuu0007/goHomeJspTeam.git**](https://github.com/psuu0007/goHomeJspTeam.git)

**[ web05\_338 ]**

**[ EL 태그 ]**

* **EL 태그**

: jsp에서 적용할수 있는 외부 라이브러리

* Expresiion Language은 콤마(.)와 대괄호([ ])를 사용하여

자바 빈의 프로퍼티(속성-getter,setter)나 맵, 리스트, 배열의 값을 보다 쉽게 꺼내게 해주는 기술이다.

JSP 에서는 주로 보관소에 들어 있는 값을 꺼낼 때 사용한다.

EL을 사용하면 액션 태그를 사용하는 것보다 훨씬 더 간단히 보관소에 들어 있는 객체에 접근하여 값을 꺼내거나 메서드를 호출할수 있다

* **EL 표기법**

**${객체이름.프로퍼티}**

* EL 표기법에서 객체를 찾는 순서

: jspContext -> ServletRequest -> HttpSession -> ServletContext -> null

* + - Request 랑 session에 동일한 이름의 el표기법이 존재한다면 무조건 request먼저 접급근하기 때문에 session에 접근이 안된다 그럼 어떻게? 검색범위를 지정해줘야함
    - 검색 범위 지정 방식

**${검색하고자하는범위.객체이름.프로퍼티}**

예) ${sessionScope.member.name}

${requestScope.member.name}

* ${memberDto.name} 는 아래 내용과 동일하다

<%

MemberDto memberDto = **new** MemberDto;

memberDto.getName();

%>

* 자바 문법이 el표기법 때문에 사라진다
* Getter, setter 패턴이 없다면 사용이 불가능하다
* 더 이상 request.getAttribute(“”) 할 필요없다
* 더 이상 getname 하던것을 하지 않아도 됨

**[ JSTL ] – (JSP Standard Tag Library)**

: JSTL 은 기본 태그가 아니므로 사용하려면 JSTL API 및 이를 구현한 자바 라이브러리를 별도로 내려받아야 한다.

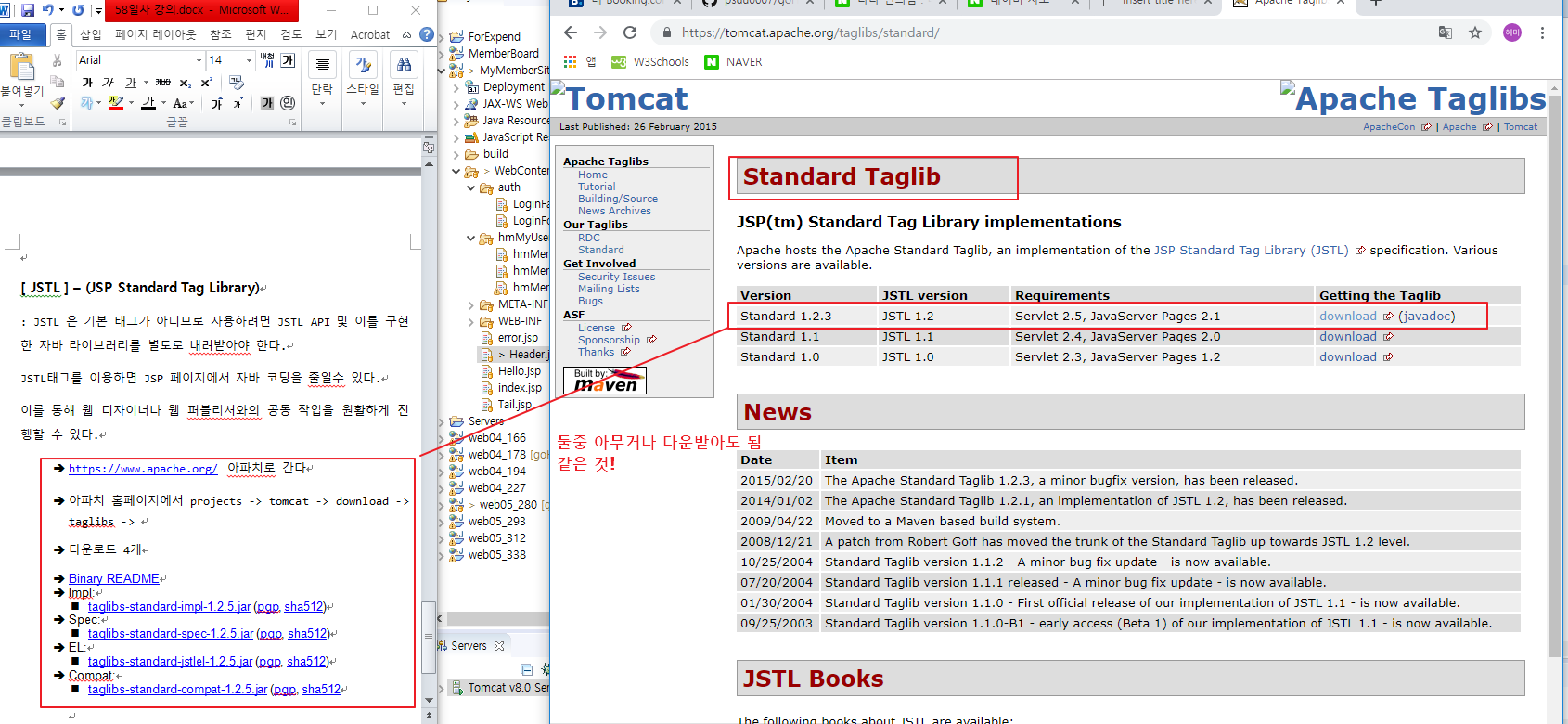
JSTL태그를 이용하면 JSP 페이지에서 자바 코딩을 줄일수 있다.

이를 통해 웹 디자이너나 웹 퍼블리셔와의 공동 작업을 원활하게 진행할 수 있다.

* <https://www.apache.org/> 아파치로 간다
* 아파치 홈페이지에서 projects -> tomcat -> download -> taglibs ->
* 다운로드 4개

[Binary README](https://www.apache.org/dist/tomcat/taglibs/taglibs-standard-1.2.5/README_bin.txt)

1. Impl:
   * [taglibs-standard-impl-1.2.5.jar](http://apache.mirror.cdnetworks.com/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-impl-1.2.5.jar) ([pgp](https://www.apache.org/dist/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-impl-1.2.5.jar.asc), [sha512](https://www.apache.org/dist/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-impl-1.2.5.jar.sha512))
2. Spec:
   * [taglibs-standard-spec-1.2.5.jar](http://apache.mirror.cdnetworks.com/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-spec-1.2.5.jar) ([pgp](https://www.apache.org/dist/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-spec-1.2.5.jar.asc), [sha512](https://www.apache.org/dist/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-spec-1.2.5.jar.sha512))
3. EL:
   * [taglibs-standard-jstlel-1.2.5.jar](http://apache.mirror.cdnetworks.com/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-jstlel-1.2.5.jar) ([pgp](https://www.apache.org/dist/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-jstlel-1.2.5.jar.asc), [sha512](https://www.apache.org/dist/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-jstlel-1.2.5.jar.sha512))
4. Compat:
   * [taglibs-standard-compat-1.2.5.jar](http://apache.mirror.cdnetworks.com/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-compat-1.2.5.jar) ([pgp](https://www.apache.org/dist/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-compat-1.2.5.jar.asc), [sha512](https://www.apache.org/dist/tomcat/taglibs/taglibs-standard-1.2.5/taglibs-standard-compat-1.2.5.jar.sha512)

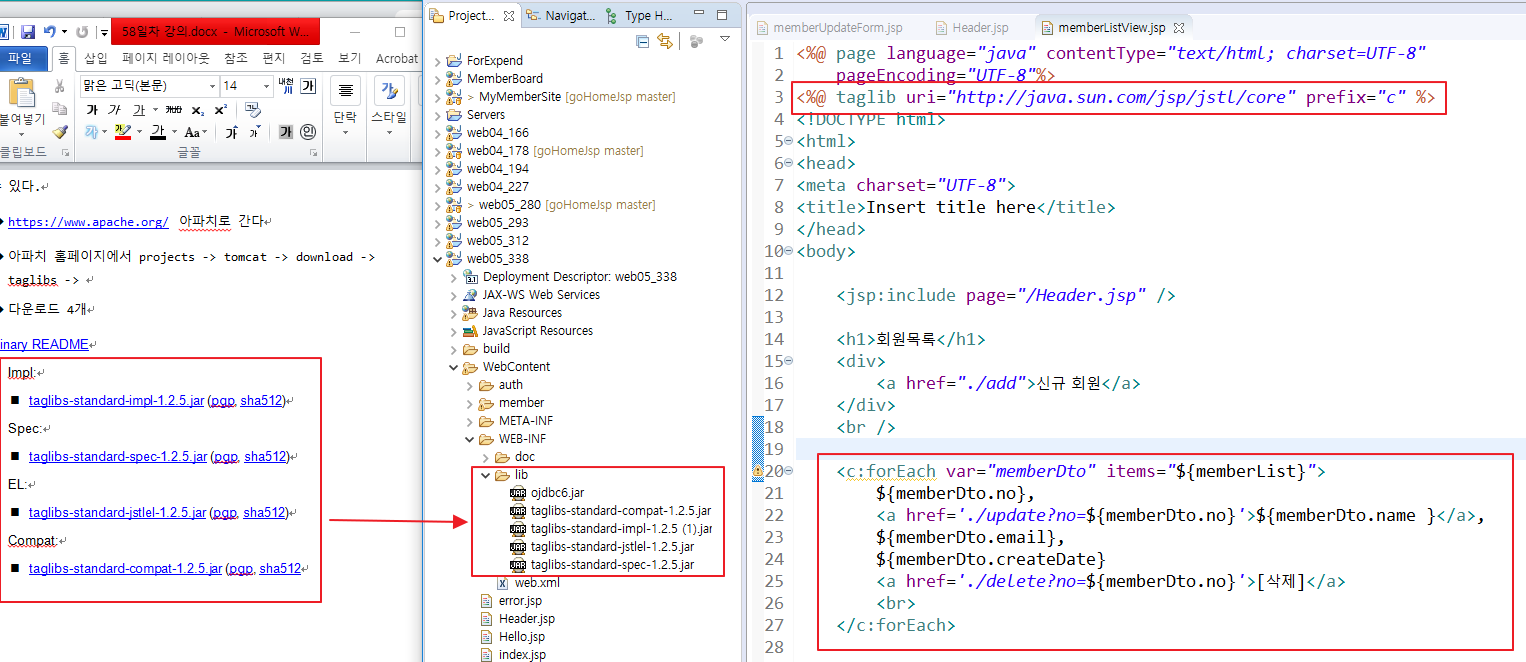


* 위의 네개 파일을 다운받은 후

Web05\_338 프로젝트의 ,

WebContent -> WEB-INF -> lib 에 위의 네가지 파일을 복사 붙여넣기한다

* 외부에서 다운받은 파일을 사용할때는 WebContent -> WEB-INF -> lib 에 넣은후 사용한다



* 외부라이브러리를 사용하기위해서는?
* Taglib 을 사용하기위해 선언한다

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"* %>

* Core태그의 첫글자(prefic)를 따서 C 로 명명하고, 앞으로 태그를 사용하면 된다

<c:forEach var=*"memberDto"* items=*"*${memberList}*"*>

${memberDto.no},

<a href=*'./update?no=*${memberDto.no}*'*>${memberDto.name }</a>,

${memberDto.email},

${memberDto.createDate}

<a href=*'./delete?no=*${memberDto.no}*'*>[삭제]</a>

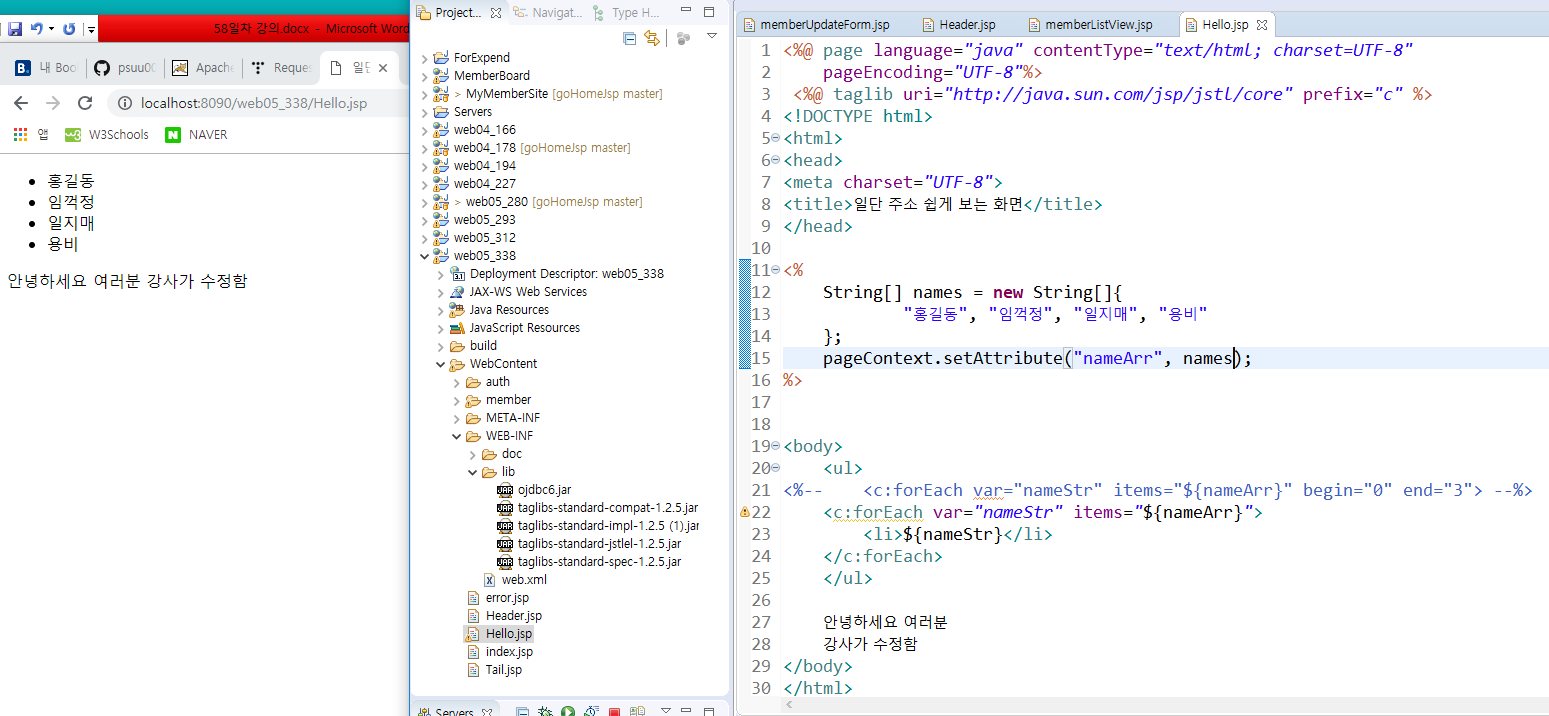
<br>

</c:forEach>

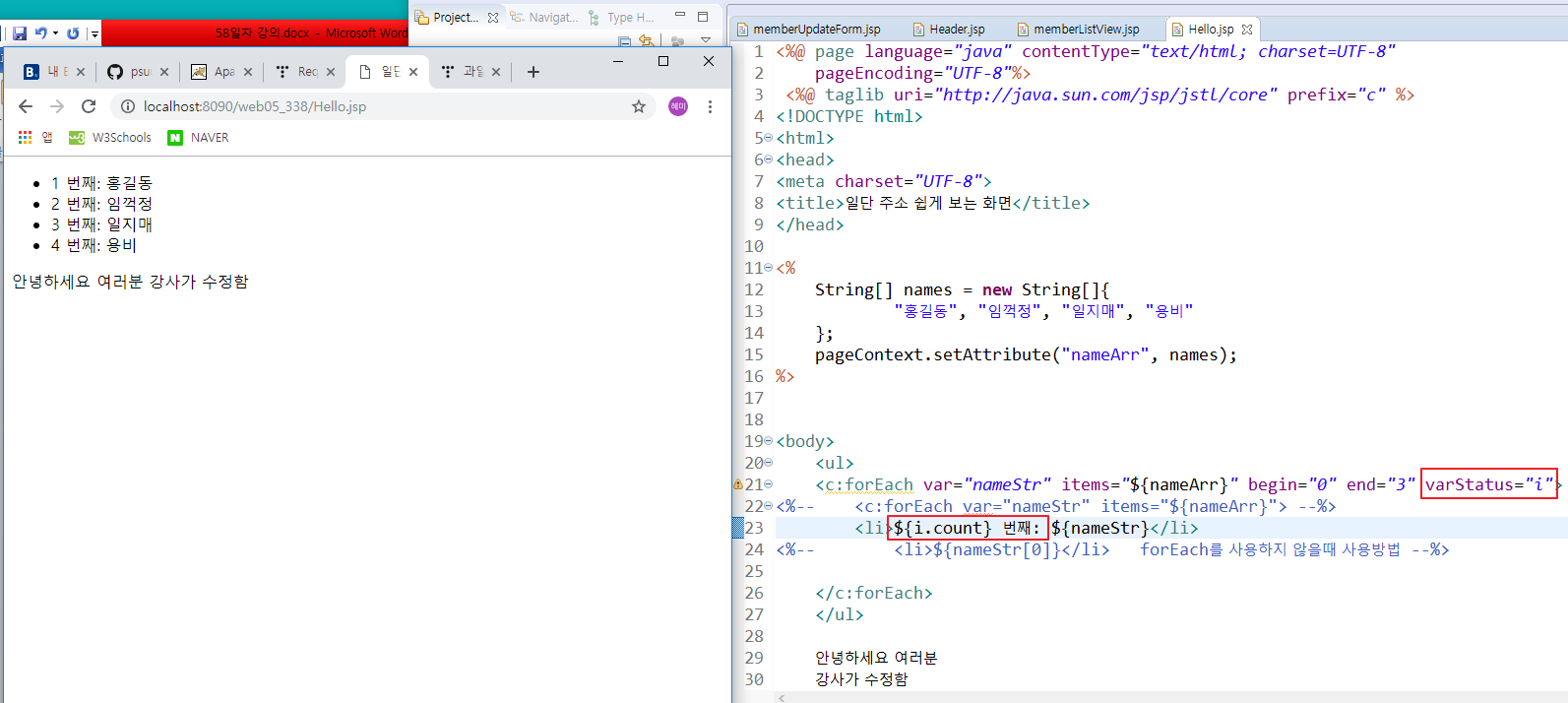
* + - var=변수명, items=목록데이터,

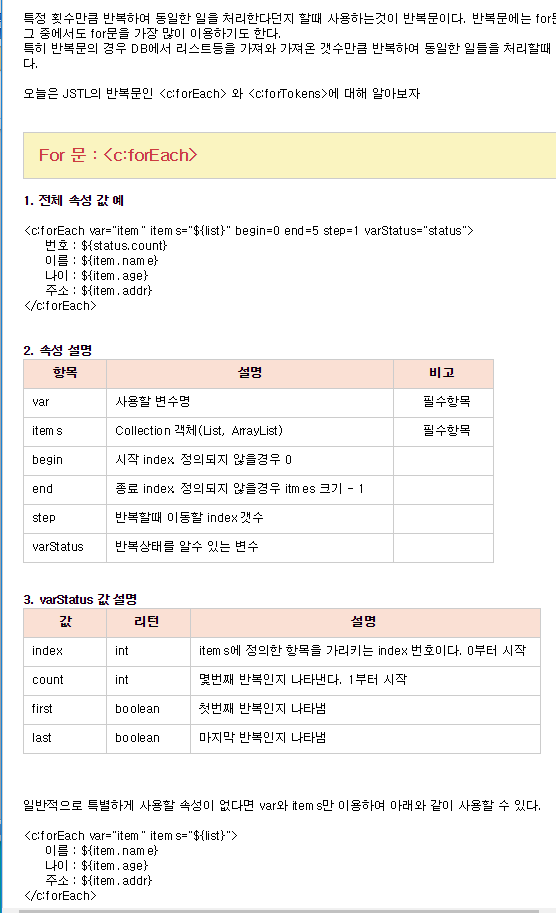
begin=시작인덱스, end=종료인덱스

* + - Var 는 forEach 안에서 사용 가능하다



* VarStatus 는 반복상태를 알수 있는 변수





* (실습) 세가지 정보를
  + - * 번호 1~3
      * 과목 3과목
      * 점수 0~100

를 배열안에 넣고 forEach를 사용해서 출력해라.

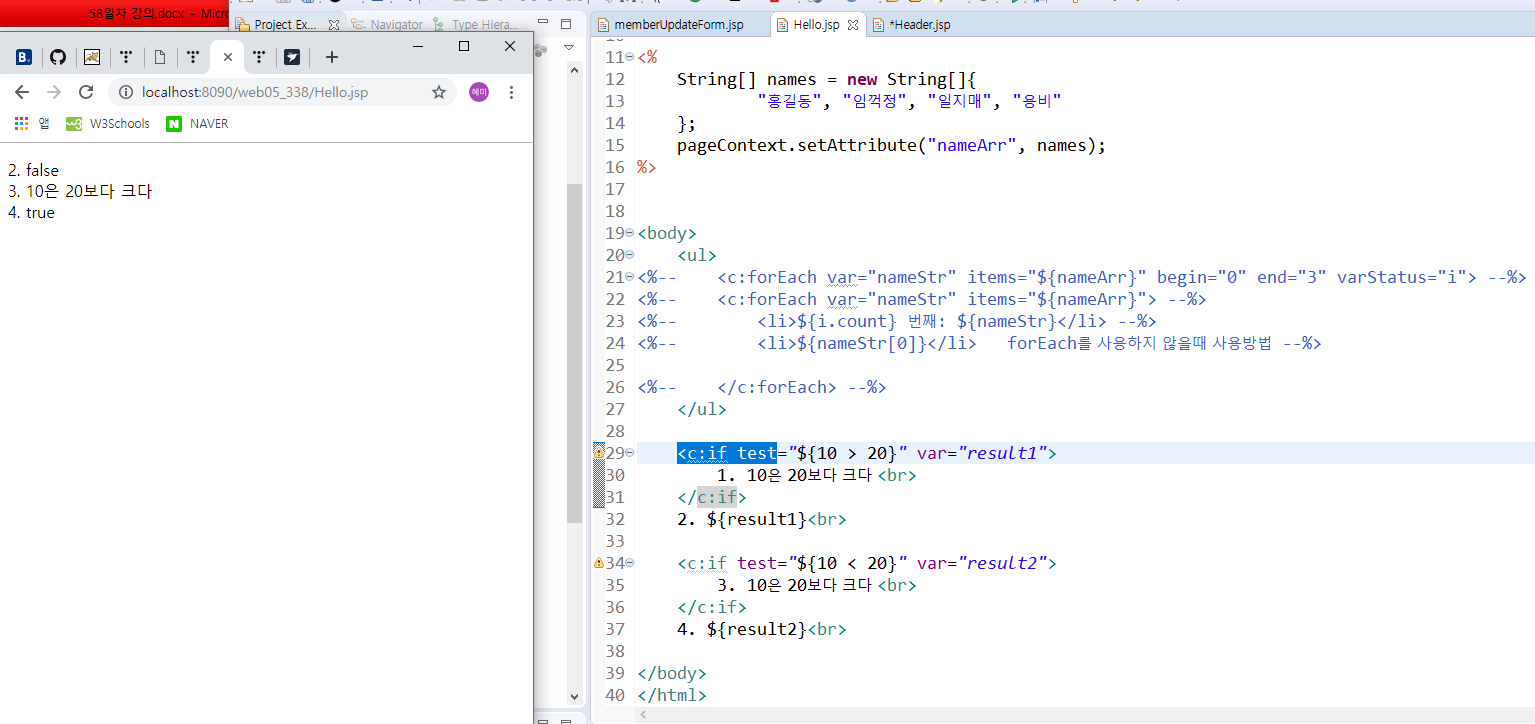
다한 사람은 Dto 객체로 구현해보시오

변호 과목명 점수

1 국어 30

2 영어 40

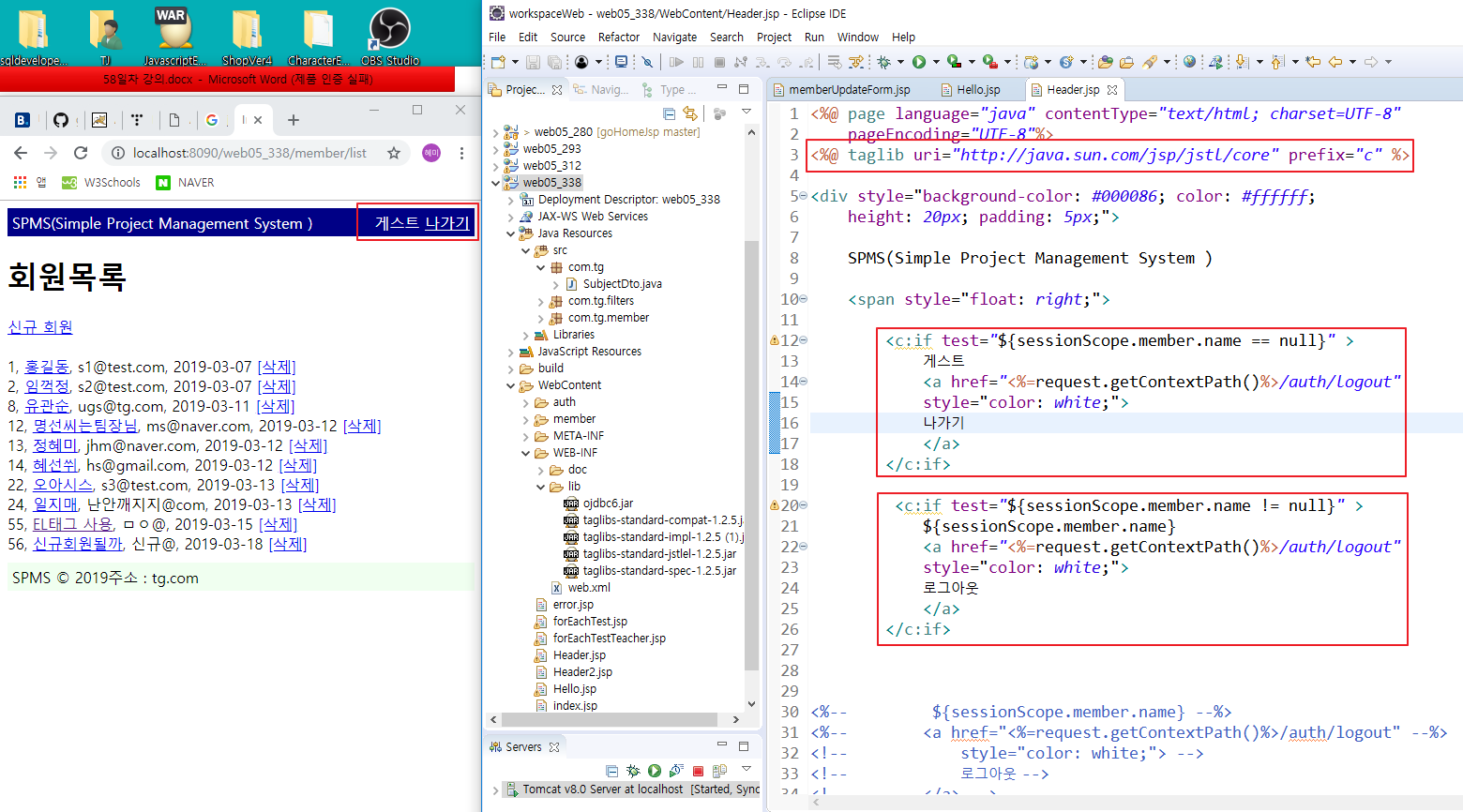
1. 수학 100

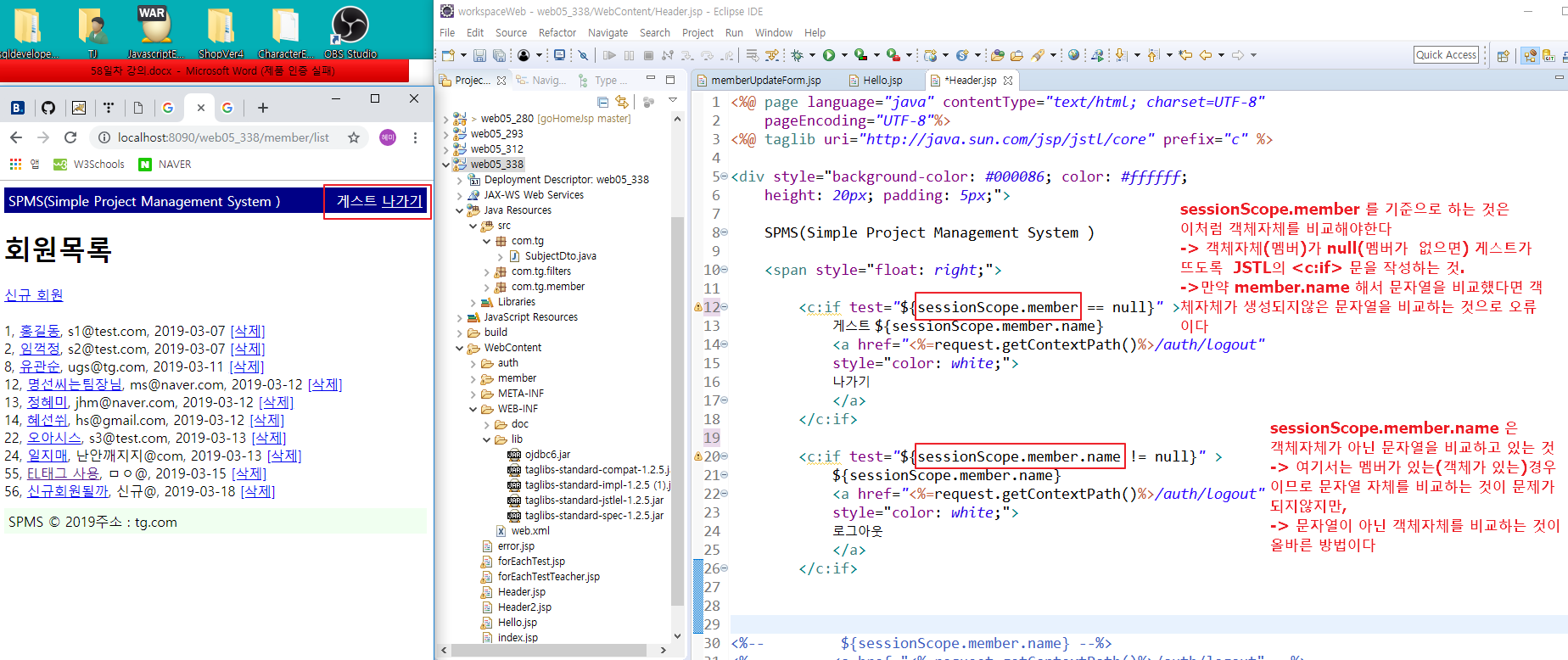


* 로그인하지않은 사용자가 member/list 에 접근했을 때, 게스트 표시가 뜨게하고, 나가기를 누를수있도록

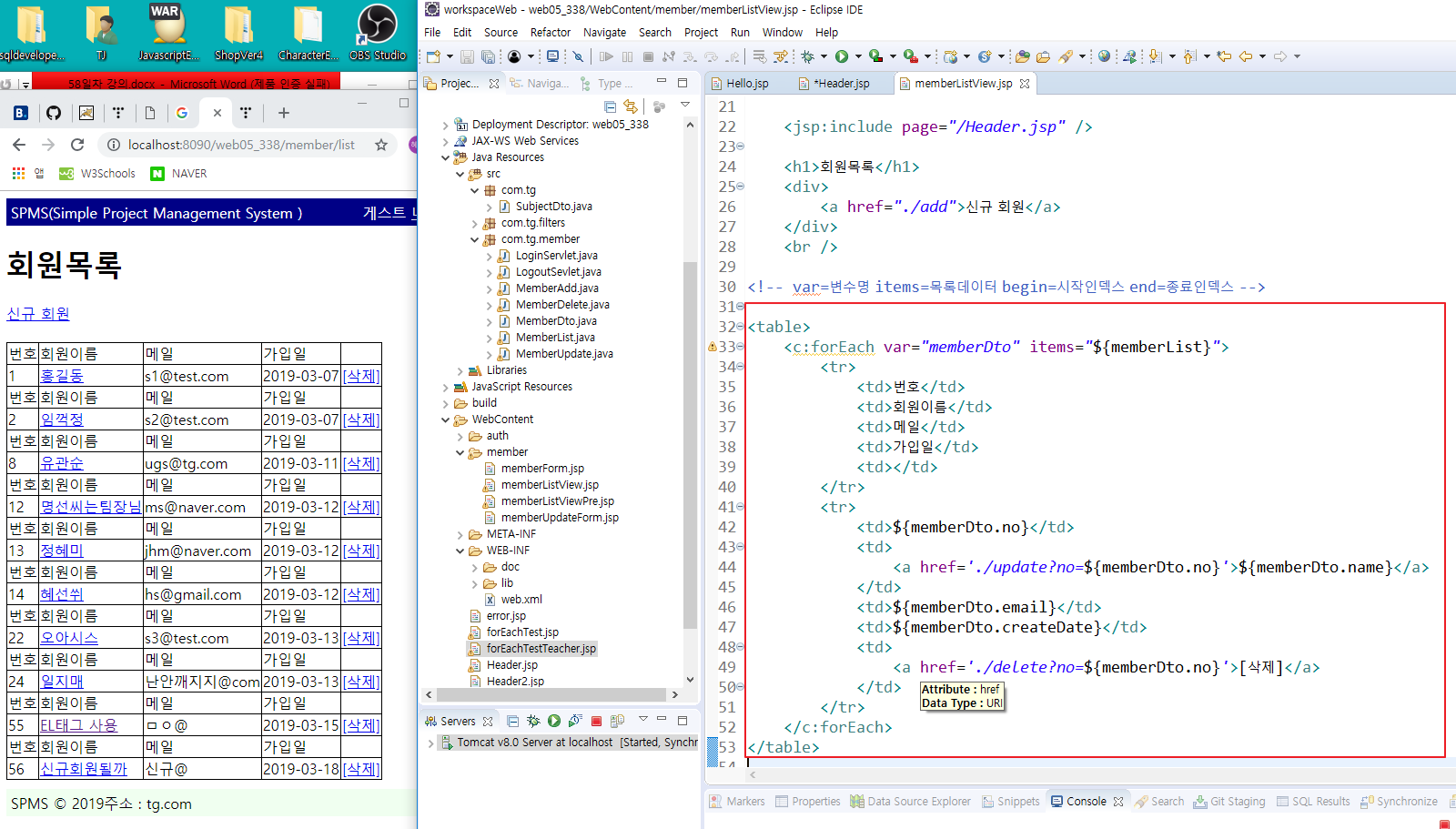
EL태그와 JSTL의 <c:if> 문을 사용하여 만드시오

* JSTL 문을 사용하려면 우선 taglib를 가져오고 작성해야함





* Member/list 를 좀 보기좋게 화면단에서 바꿔보자
* 아래 사진에서 안보이지만 <style> 이 적용되어 있음



* memberListView.jsp 파일에서 ctrl + F11 실행시켜서는 안된다
* 이렇게 직접 실행시키면 멤버가 뜨지 않는다 왜??
* 모든 것은 컨트롤러를 통해서 접근되어져야 데이터베이스의 내용이 들어가는 것
* 그러므로 주소창에 경로를 직접 적어넣어서 접근해야만 한다
* 미니 팀프로젝트 – 데이터베이스에 있는 내용을 통해 게시판을 만들어라!
  + - * 글쓰기가 되는 게시판 페이지를 만들어라
      * 연동은 시켜도 안시켜도 좋다
      * CRUD
      * 페이지 링크를 타면 CRUD가 이렇게 뜬다

또는