

# 비전공자를 위한

## jsp 웹프로그래밍

발행일 : 2023.03.16

변경일 : 2023.03.16

저자 : 박수민

이메일 : [foxman12@hanmail.net](mailto:foxman12@hanmail.net)

발행처 : 휴먼교육

## 목차

---

<b>01. JSP</b>	<b>4</b>
> 01. 프로그램설치	4
> 02. html작업하기	18
> 01. html,jsp,servlet page만들기	21
> 02. html,jsp,servlet 경로 이해하기	30
> 03. jsp 문법 정리	36
> 04. GET,POST를 이용한 사용자 입력처리	42
> 05. Redirect 와 forward	56
> 06. cookie사용방법	68
> 07. 쿠키관리 프로그램 구현	75
> 08. 쿠키를 사용한 로그인	83
> 08. frontcontroller	88
> 09. cookie관리프로그램 to frontcontroller	93
> 10. session	99
> 11. sessionLogin to frontcontroller	103
> 12. scope	107
> 13. el and jstl	111
> 14. jdbc 데이터베이스 연결	131
> 15. jdbc model1	136
> 16. jdbc model2	147
> 17. CustomHobby-Custom만들기	160
> 18. CustomHobby-Hobby만들기	175
> 19. CustomHobby-CustomHobby join만들기	184
> 20. CustomHobby-CustomHobbies join만들기	196



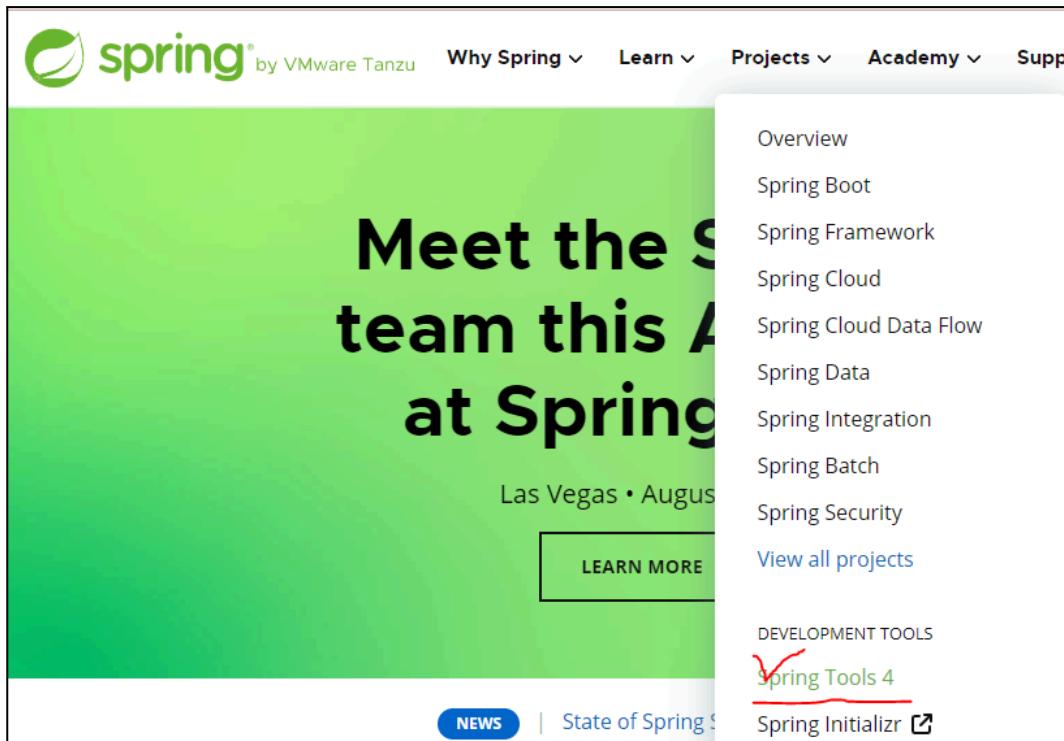
# 01. JSP

## > 01. 프로그램설치

jsp 개발을 위한 프로그램을 설치할 예정이다.

1. spring sts 설치하기

[www.spring.io](http://www.spring.io) 사이트에 들어가 projects >> spring tools4를 선택한다.



jsp는 옛날  
부터 사용하던  
기술이여서  
과거 버전툴을  
사용할  
예정이다.  
아래로  
스크롤해서  
내려가면  
다음과 같은  
부분에  
Spring Tool  
Suite 3  
wiki에 들어  
간다.  
윈도우즈  
버전을  
다운로드 받을  
예정이어서

XXXwin32-x86\_64.zip파일을 클릭해서 다운로드 한다.

full distribution on Eclipse 4.12

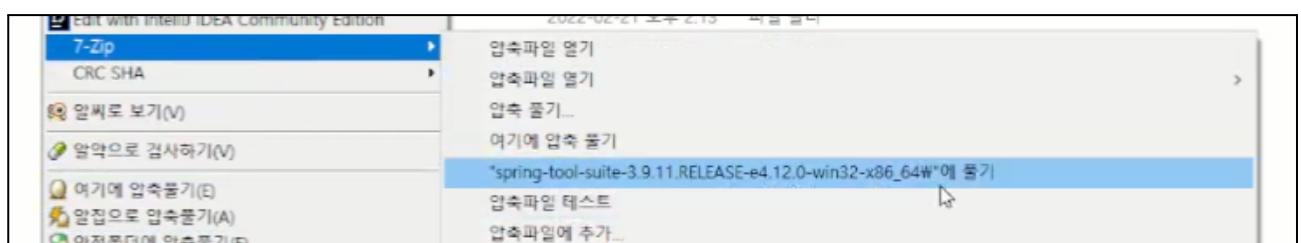
- [https://download.springsource.com/release/STS/3.9.11.RELEASE/dist/e4.12/spring-tool-suite-3.9.11.RELEASE-e4.12.0-win32-x86\\_64.zip](https://download.springsource.com/release/STS/3.9.11.RELEASE/dist/e4.12/spring-tool-suite-3.9.11.RELEASE-e4.12.0-win32-x86_64.zip)
- [https://download.springsource.com/release/STS/3.9.11.RELEASE/dist/e4.12/spring-tool-suite-3.9.11.RELEASE-e4.12.0-macosx-cocoa-x86\\_64.dmg](https://download.springsource.com/release/STS/3.9.11.RELEASE/dist/e4.12/spring-tool-suite-3.9.11.RELEASE-e4.12.0-macosx-cocoa-x86_64.dmg)
- [https://download.springsource.com/release/STS/3.9.11.RELEASE/dist/e4.12/spring-tool-suite-3.9.11.RELEASE-e4.12.0-linux-gtk-x86\\_64.tar.gz](https://download.springsource.com/release/STS/3.9.11.RELEASE/dist/e4.12/spring-tool-suite-3.9.11.RELEASE-e4.12.0-linux-gtk-x86_64.tar.gz)

The screenshot shows the official 7-Zip website at [7-zip.org](https://7-zip.org/). The main content area features the 7-Zip logo and a brief description: "7-Zip is a file archiver with a high compression ratio." Below this is a section titled "Download 7-Zip 22.01 (2022-07-15) for Windows:". A red checkmark points to the first item in a table:

Link	Type	Windows	Size
<a href="#">Download</a>	.exe	64-bit x64	1.5 MB
<a href="#">Download</a>	.exe	32-bit x86	1.2 MB
<a href="#">Download</a>	.exe	64-bit ARM64	1.5 MB

2.7zip 설치하기  
압축 프로그램은  
7zip를 되도록  
사용하자.

<https://7-zip.org/>  
사이트에 가서 체크  
되어 있는  
프로그램을  
다운로드하여  
설치하자.



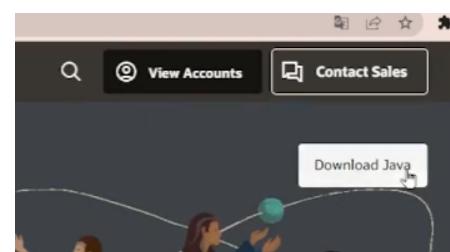
작업하고자 하는 파일에서 마우스 오른쪽을 클릭하면 7-zip 관련 메뉴가 나온다. 압축을  
하거나 풀 때 사용하자. 다운로드된 파일을 관리자 권한으로 c에 복사한 다음 오른쪽

마우스를 클릭하여 압축을 풀어보자.

### 3. jdk 설치하기

[www.oracle.com](http://www.oracle.com) 오라클 사이트에  
들어가서 맨 아래로 스크롤하여 java를  
클릭한다.

이동한 사이트 오른쪽 상단에 다음과  
같은 이미지가 있다.



다운로드 버튼을 클릭해서 사이트를  
이동한다.

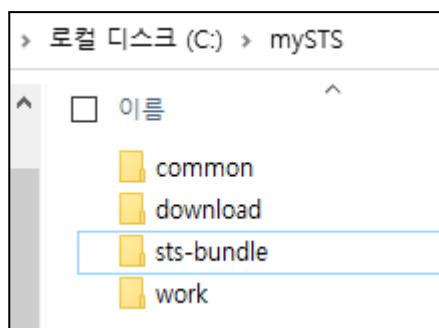
이동한 사이트에서 아래로 스크롤  
하다 보면 다음 페이지 이미지 jre8 탭  
부분이 나타난다. jre8 탭을 클릭한  
다음 windows 탭을 선택한 다음에

[jre-8u371-windows-x64.tar.gz](http://jre-8u371-windows-x64.tar.gz) 파일을 클릭해서 다운로드 받자. 다운로드 할 때 로그인을 해야  
하니 회원가입을 하고 로그인한 다음에 다운로드를 하자.

JRE 8  
Java SE Runtime Environment 8u371

JRE 8 software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE.

Product/file description	File size	Download
x86 Online Installer	2.22 MB	<a href="#">jre-8u371-windows-i586-iftw.exe</a>
x86 Offline Installer	56.89 MB	<a href="#">jre-8u371-windows-i586.exe</a>
x86 Compressed Archive	72.76 MB	<a href="#">jre-8u371-windows-i586.tar.gz</a>
x64 Installer	62.62 MB	<a href="#">jre-8u371-windows-x64.exe</a>
x64 Compressed Archive	79.13 MB	<a href="#">jre-8u371-windows-x64.tar.gz</a>



왼쪽 이미지처럼 c드라이버에 압축파일 sts프로그램 폴더명을 mySTS로 변경하고 안에다가 common, download, work 폴더를 추가로 만들었다. common폴더는 프로그램 실행시 사용할 파일들을 저장할 곳이고, download는 프로그램 설치를 위해서 다운로드한 모든 파일을 저장해놓은 폴더이다. 보통 download폴더에 파일들을 다운로드 받은 다음 압축을 풀어서 common 폴더에 복사해서 사용한다.

상위 이미지는 작업이 이루어진 download폴더와 common폴더이다. download는 프로그램 설치에 필요한 보관용이고 common은 실제 사용할 파일이라고 보면 된다. work폴더는 sts에서 프로그래머가 작업한 코드 파일들을 보관할 폴더이다. 프로그램 실행시 해당 폴더를 work폴더로 설정해야 한다. 나중에 자세히 확인해 볼 예정이다.

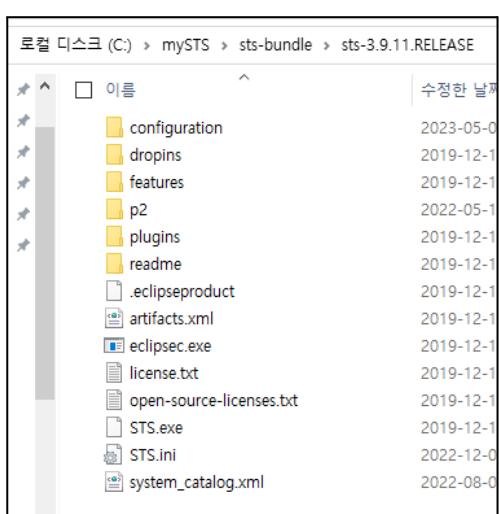
다운로드한 java 실행하는데 필요한 jre파일 jre-8uXXX.tar.gz를 download 폴더에 복사해놓은 다음 7zip으로 압축을 푸 다음 생성된 파일을 한번 더 압축을 풀면 사용 할 수 있는 폴더가 생긴다. 2번 압축을 풀어야 한다는 점을 잊지 말자.

4. Apache tomcat를 다운로드 하여 설치해 보자. Apache Tomcat은 웹 애플리케이션

서버다. 웹서버는 인터넷에서 사용하는 웹사이트를 구동하기 위한 소프트웨어입니다. 사용자가 브라우저를 통해서 웹사이트의 정보를 저장하고 있는 파일들을 요청하면, 웹서버는 이를 확인해서 클라이언트(사용자, 요청자)에게 요청한 주소에 해당하는 데이터를 전달한다.

우리가 사용할 웹서버는 톰캣이다. 브라우저에 tomcat.apache.org 주소를 입력하면 왼쪽과 같은 화면이 나오고 download 메뉴에 tomcat 9 를 클릭하고 이미지 오른쪽 맨 아래 부분 zip 파일을 클릭하면 톰캣을 다운로드 받을 수 있다. 다운로드 한 파일은 download 폴더로 옮겨서 압축을 풀고 압축이 풀린 파일의 폴더는 common 폴더로 옮기자.

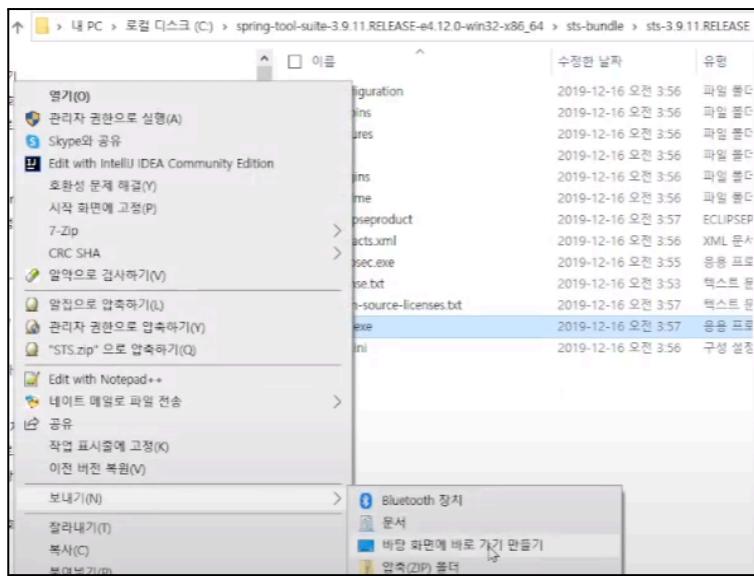
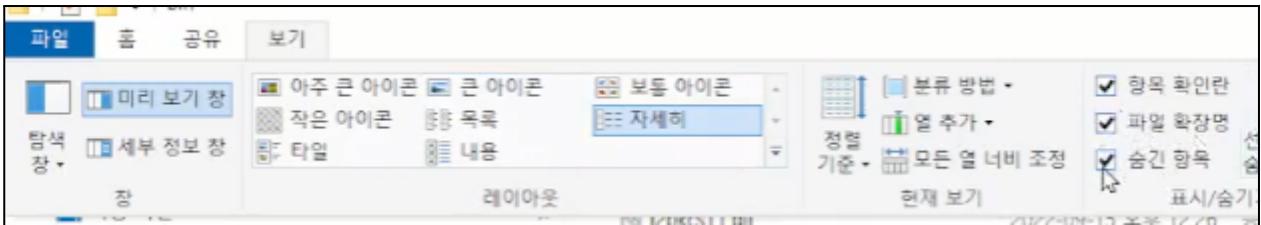
5. ojdbc8\_g.jar 파일은 오라클 jdbc를 사용할 때 필요한 jar 파일이다. 오라클을 설치하였다면 설치된 app 폴더의 본인 계정 밑에 다음과 같은 경로에 있다.  
C:\app\foxman12\product\18.0.0\dbhomeXE\jdbc\lib  
설치하지 않았다면 웹에 검색해서 해당 파일을 찾아 다운로드 받자.



ojdbc8\_g.jar 파일은 download 폴더와 common 폴더에 복사해 놓자. ojdbc8\_g.jar는 압축을 풀지 않아야 한다.

6. 다운로드된 sts를 압축을 푼 폴더 C:\mySTS\sts-bundle\sts-3.9.11.RELEASE에 가면 왼쪽 처럼 sts.exe 파일이 있는데 이 파일이 툴을 실행하고 자 할 때 클릭하는 파일이다. 다음 이미지 처럼 sts.exe 파일에 오른쪽 마우스 클릭 >> 보내기 >> 바탕화면에 바로가기 만들기를 선택하여 바탕화면에 바로가기를 만들어 보자.

파일의 확장자가 보이지 않으면 폴더 상단의 보기 메뉴를 선택해서 아래 이미지 처럼 파일 확장명 숨김 항목들을 체크 한다.



바탕화면에 바로가기가 만들어 지면 왼쪽 이미지처럼 아이콘이 생긴다. sts를 실행하고 싶다면 바탕화면의 해당 아이콘을 클릭하면 된다.

7. sts.exe 파일이 있는 폴더에 sts.ini 파일이 있는데 이 파일은 sts프로그램이 실행될 때 읽어서 사용하는 설정 파일이다. 설정 파일이란 소프트웨어 프로그램의 동작과 설정을 제어하기 위해 사용되는 텍스트 또는 데이터

파일입니다. INI 파일은 섹션(Section)과 키(Key)의 쌍으로 구성된다.

```
STS.ini - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
-vm
C:\mySTS\common\jre1.8.0_333\bin\javaw.exe
-startup
plugins/org.eclipse.equinox.launcher_1.5.400.v201905
-launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86
-product
org.springsource.sts.ide
--launcher.defaultAction
openFile
-vmargs
-Dosgi.requiredJavaVersion=1.8
-Xms256m
-Xmx1024m
-XX:+UseG1GC
-XX:+UseStringDeduplication
--add-modules=ALL-SYSTEM
-Dosgi.module.lock.timeout=10
-Dfile.encoding=UTF-8
```

왼쪽 이미지처럼 파일을 메모장으로 열어서 javaw.exe 파일 위치와 인코딩 설정을 해야 한다.

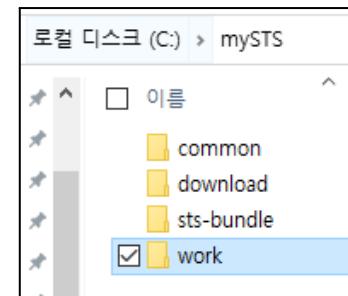
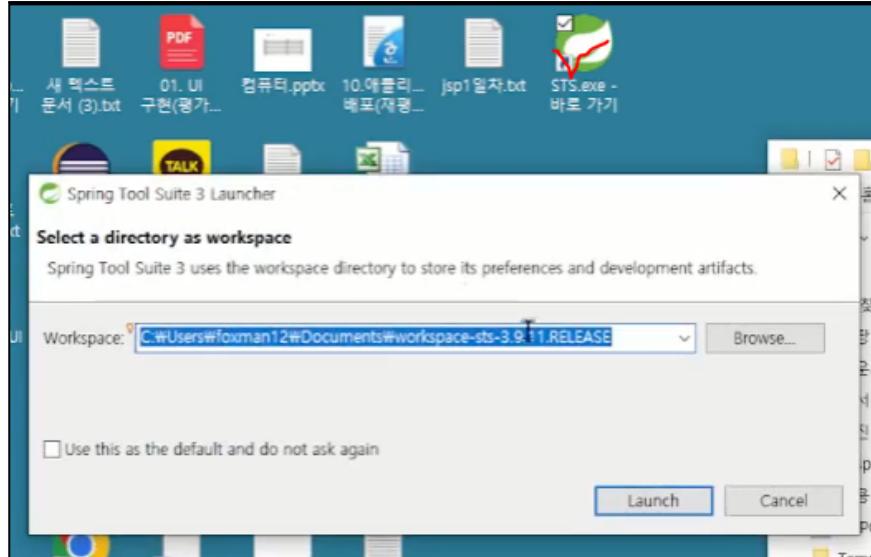
설치후 ini 파일을 열어보면 빨간줄 부분이 존재하지 않는다. -vm를 기술하고 다음 줄에

C:\mySTS\common\jre1.8.0\_333\bin\javaw.exe 를 넣자.  
common 폴더에 복사한 jre 관련 파일 중 javaw.exe 파일을 찾아가는 것이라 설치 위치가 다르면 적절히 찾아서 입력하자.

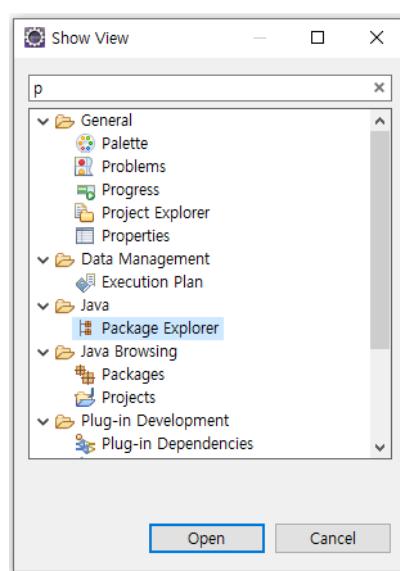
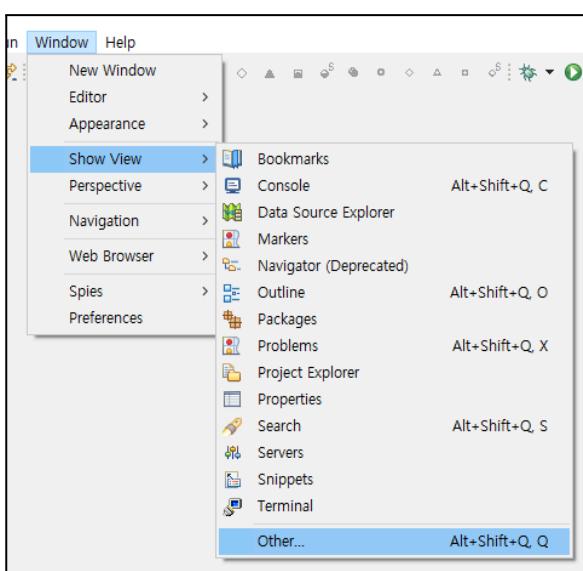
인코딩을 위해서 맨마지막 줄에 -Dfile.encoding=UTF-8를 넣자.  
파일이 열려 있어서 누군가 사용하려고 하면 문제가 발생할 수 있다. 인코딩이란? 텍스트를 컴퓨터에서 이해하기 위해 문자를 숫자로 변환하는 규칙이며, 올바른 문자 인코딩을 사용해야 텍스트가 정확하게 해석됩니다. 약속된 인코딩 utf-8를 사용해야 한글이 깨지지 않는다. 해당 파일을 저장 후 닫은 다음에 sts.exe를 실행하면

ini파일의 내용이 실행 될 때 적용 된다. 실행 할 가상 머신 설정과 툴에서 파일을 만들때 사용할 인코딩 설정을 한 것이다.

8. 바탕 화면의 바로가기 파일을 클릭하면 sts프로그램이 실행되고 workspace 작업 폴더를 설정 하라고 나온다. browser버튼을 클릭해서 이전에 만들어 놓은 mySTS폴더 안의 work 폴더를 찾아 설정하자.



9. sts실행하면 실행창에 여러가지 창들이 떠있다. 만약 실수로 닫아서 없다면 다음과 같은 방법으로 창을 찾을 수 있다.

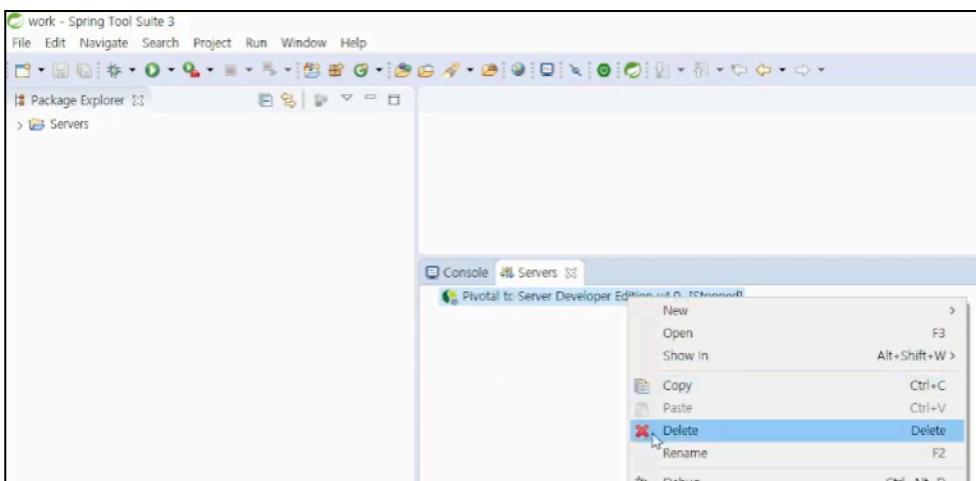


Package Explorer창 찾는 방법은 상단 메뉴에서 이미지 처럼 window >> show View >> other를 선택한 다음 edit창에 검색하고자 하는 창 이름을 입력해서 Package Explorer를 찾아 클릭하면 된다.

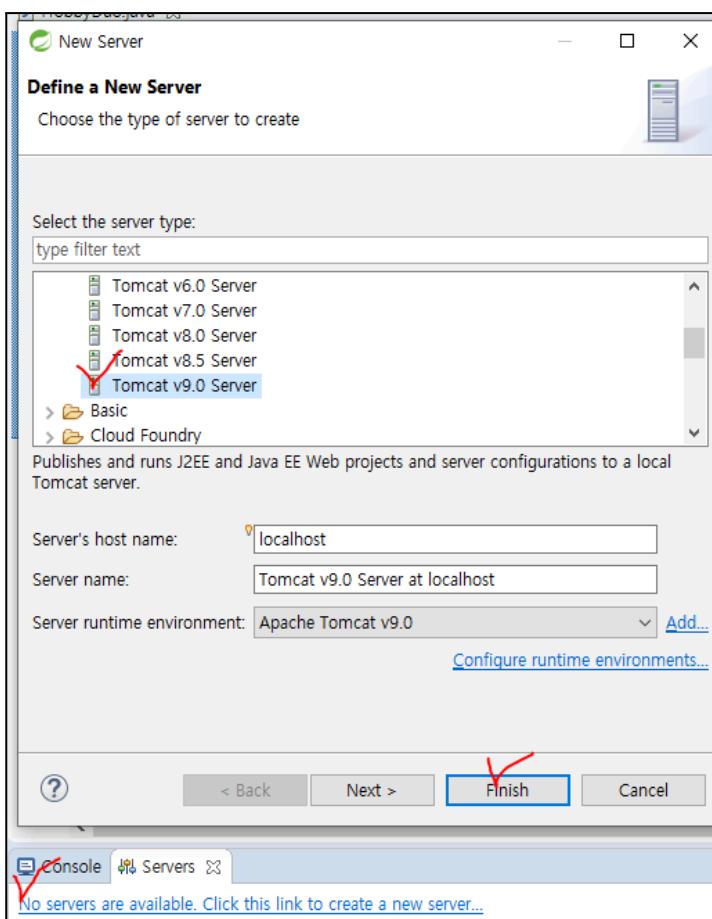
servers창을 찾는 방법은 왼쪽 이미지 처럼 server로 검색해서 servers창을 선택하면 된다.



다음은 톰캣을 툴에 연결하는 방법이다.

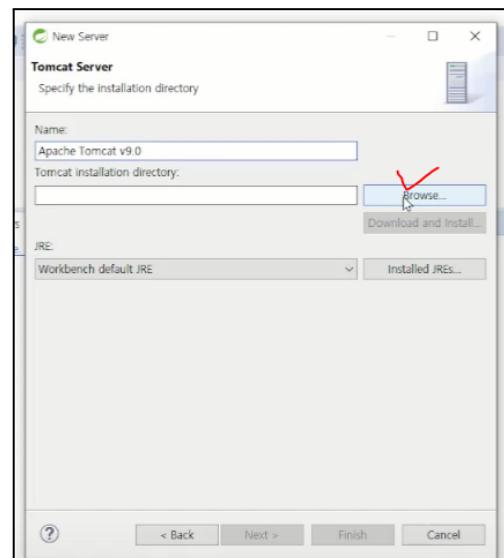


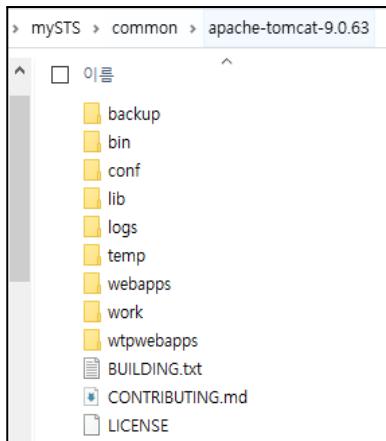
server창에 기본으로 등록되어 있는 server를 오른쪽 마우스 클릭해서 삭제를 선택해서 삭제 할 수 있다.



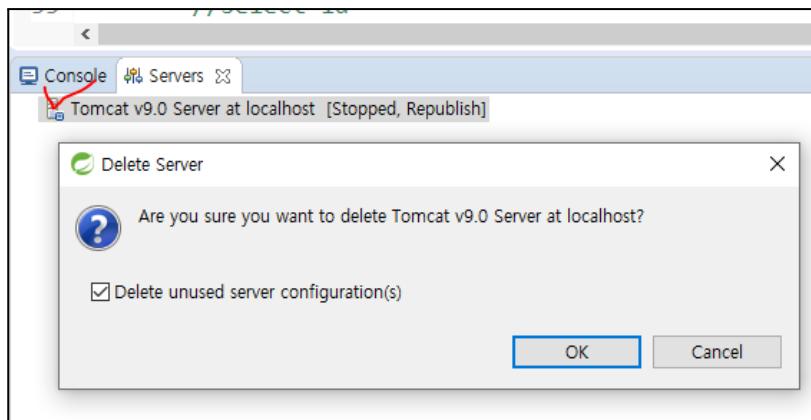
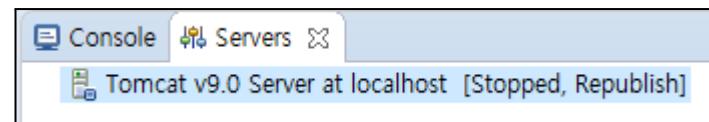
삭제하고 나면 왼쪽 이미지 하단부분처럼 no servers are available. 이라는 문자열이 생기고 클릭하면 팝업으로 생성된 창에서 스크롤을 이용하여 위쪽에 tomcat 풀더 밑에 Tomcatv9.0 Server 를 찾아 선택한후 넥스트 버튼을 클릭하면 톰캣 서버가 등록 된다.

만약 next나 finish 버튼등 원하는 버튼이 활성화 되어 있지 않아 누를 수 없는 경우가 있다. 그럴 때는 왼쪽 이미지의 add... 버튼을 누르면 다음 이미지와 같은 창이 뜬다.





browser를 눌러서 common 폴더에 복사해 넣은 tomcat폴더를 찾아서 선택한다. 하라는 대로 했으면 C:\mySTS\common\apache-tomcat-9.0.63 폴더가 선택된다. 성공적으로 등록 되었으면 아래 이미지 처럼 된다.

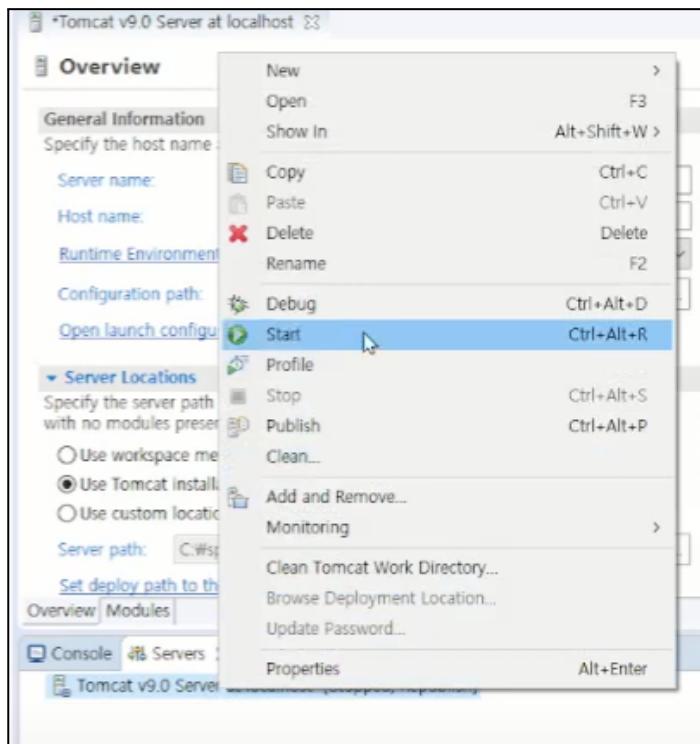
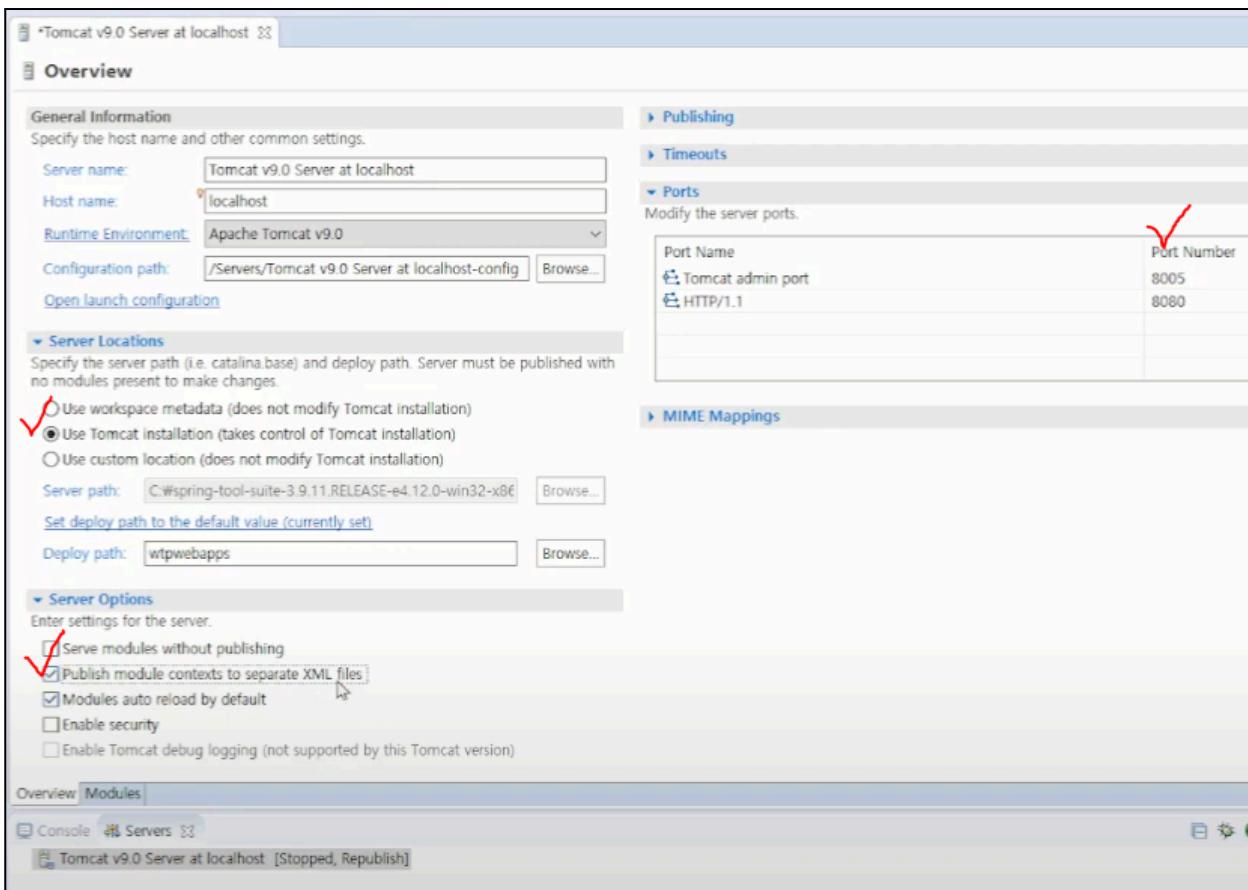


톰캣에 문제가 발생하면 삭제하고 다시 등록하는 것도 좋은 방법이다. 삭제하는 방법은 Tomcat 체크된 곳에서 오른쪽 마우스를 클릭한 다음 삭제를 클릭하면 왼쪽과 같은 이미지가 뜨는데 완전히 삭제하려면 반드시 delete unused server configurations를 체크 한 다음 삭제해야 한다.

톰캣 문제중 많이 발생 하는 원인은 이전 동작 시킨 톰캣이 툴버그로 인해서 종료되지 않아 작성한 프로그램을 톰캣에서 새로 실행할 때 해당 포트를 누군가가 사용하고 있는 메시지가 생성되는 경우 이다. 이럴 때는 다음에 배울 포트 번호 설정 방법을 이용해서 포트 번호들을 이전 사용했던 번호와 겹치지 않도록 2개다 바꾸면 해결 될 수 있다. 종종 컴퓨터를 리부팅 해서 해결되는 경우도 있다.

톰캣 설정을 변경하려고 할때 변경되지 않는 경우가 있는데 설정 변경은 톰캣을 실행하기전에 설정을 해야 한다. 실행후 설정이 불가능하니 변경하고 싶으면 삭제후 다시 등록해서 변경하면 된다. servers탭에서 생성된 톰캣을 더블 클릭하면 아래와 같은 창이 뜨는데 하단에 overview탭을 클릭한 다음 체크되어 있는 3부분을 변경하자. port 번호는 겹치면 안되기 때문에 겹치지 않는 숫자로 8000번 이후 번호를 마음대로 정해서 사용하면 된다. 사용 포트가 겹쳐진 상태에서 실행되면 문제가 발생 하는데 메시지를 잘 읽어보고 문제가 발생하면 포트 변경 후 다시 저장 하여 실행하면 된다.

종종 문제가 발생하면 톰캣이 돌아가지 않는다. 삭제하고 새로 등록한 다음에 톰캣을 사용하면 문제가 해결될 수 있다. 해결이 안되는 문제가 발생하면 톰캣을 완전히 지우고 다시 실행해 보자. 최악의 경우에 톰캣과 함께 package explorer에 있는 server 폴더까지 모두 지운 다음 다시 톰캣을 등록 해서 실행해 보자. 그래도 안되면 윈도우즈를 밀거나 프로그램을 다시 설치해 보자.



설정이 끝나면 servers창에 톰캣에서 오른쪽 마우스를 클릭한 다음 start를 클릭하면 톰캣이 실행된다.

방화벽 관련 질문이 나오면 액세스 허용을 선택하면 된다.

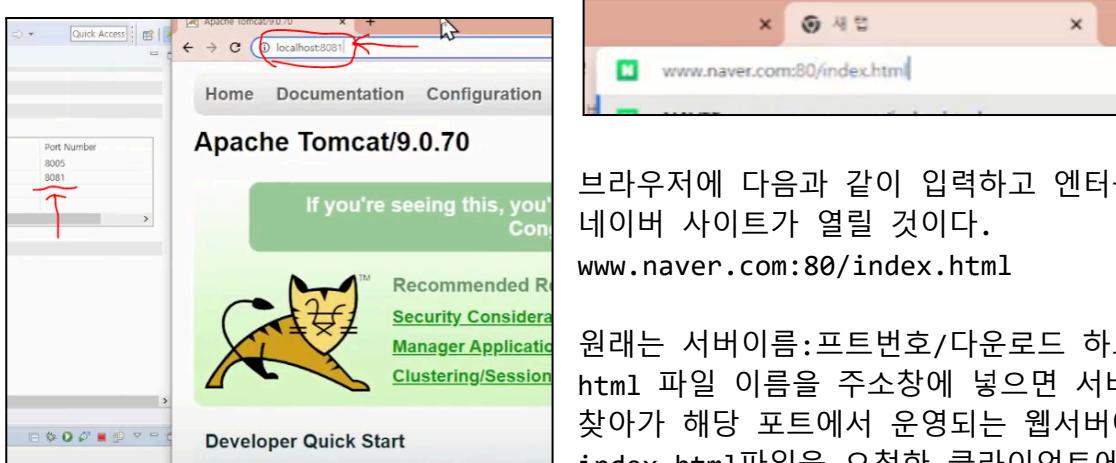


톰캣이 정상적으로 실행되었다면 상위 이미지처럼 창의 메시지가 started로 변경된 것을 확인 할 수 있다.  
톰캣을 실행하고 브라우저를 열어서 본인이 설정한 포트 번호로 주소창에 다음과 같이 입력해 보면 톰캣 이미지를 확인 할 수 있다.

`localhost:8081`

본인 컴퓨터에서 동작하는 톰캣 웹서버에 테스트로 제공하는 기본 html파일을 요청해서 본인 컴퓨터

브라우저에 보여준 것이다. 8081은 이전에 overview창에서 세팅한 포트번호와 일치해야 한다. 데이터를 요청하는 컴퓨터가 클라이언트이고 제공하는 컴퓨터가 서버이다. 네이버 사이트에서 html를 확인한다면 요청한 본인 컴퓨터가 클라이언트이고 html를 제공하는 컴퓨터가 서버이다.



브라우저에 다음과 같이 입력하고 엔터를 치면 네이버 사이트가 열릴 것이다.

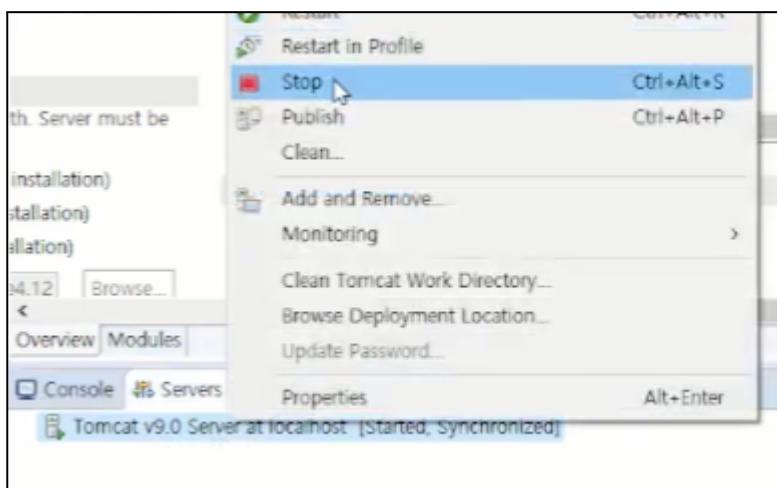
`www.naver.com:80/index.html`

원래는 서버이름:포트번호/다운로드 하고자하는 html 파일 이름을 주소창에 넣으면 서버 컴퓨터를 찾아가 해당 포트에서 운영되는 웹서버에 index.html파일을 요청한 클라이언트에게 제공해 준다.

특별히 80포트와 index.html은 생략 할 수 있다. 변경된 다른 포트와 index.html이 아닌 다른 파일 이름은 생략 할 수 없다.

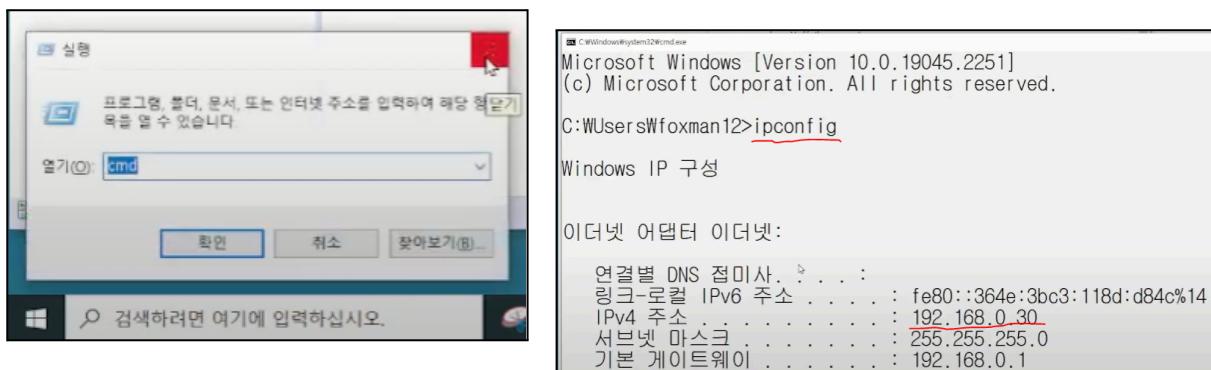
웹 프로그램 작업을 하다 보면 콘솔창에 에러 메시지가 생겨서 보기 안좋을때가 많다. 콘솔 창에서 마우스 오른쪽 클릭을 한 다음 clear 메뉴를 클릭하면 화면에 메시지들이 사라진다.

```
<terminated> Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Users\admin\Downloads\spring-  
10월 17, 2023 5:14:47 오후 org.apache.catalina.core.StandardServerInfo  
정보: 셧다운 포트를 통해 유효한 셧다운 명령을 받았습니다. 서버 인스턴스를 중지합니다.  
10월 17, 2023 5:14:47 오후 org.apache.coyote.AbstractProtocol  
정보: 프로토콜 핸들러 ["http-nio-8081"]을(를) 일시 정지 중  
10월 17, 2023 5:14:47 오후 org.apache.catalina.core.StandardService  
정보: 서비스 [Catalina]을(를) 중지시킵니다.  
10월 17, 2023 5:14:48 오후 org.apache.catalina.core.ApplicationContext  
정보: SessionListener: contextDestroyed()  
10월 17, 2023 5:14:48 오후 org.apache.catalina.core.ApplicationContext  
정보: ContextListener: contextDestroyed()  
10월 17, 2023 5:14:48 오후 org.apache.catalina.core.StandardEngine  
정보: 프로토콜 핸들러 ["http-nio-8081"]를 정지했습니다.  
10월 17, 2023 5:14:48 오후 org.apache.catalina.core.StandardService  
정보: 프로토콜 핸들러 ["http-nio-8081"]을 정지했습니다.  
No servers are available. Click this link to create one.
```



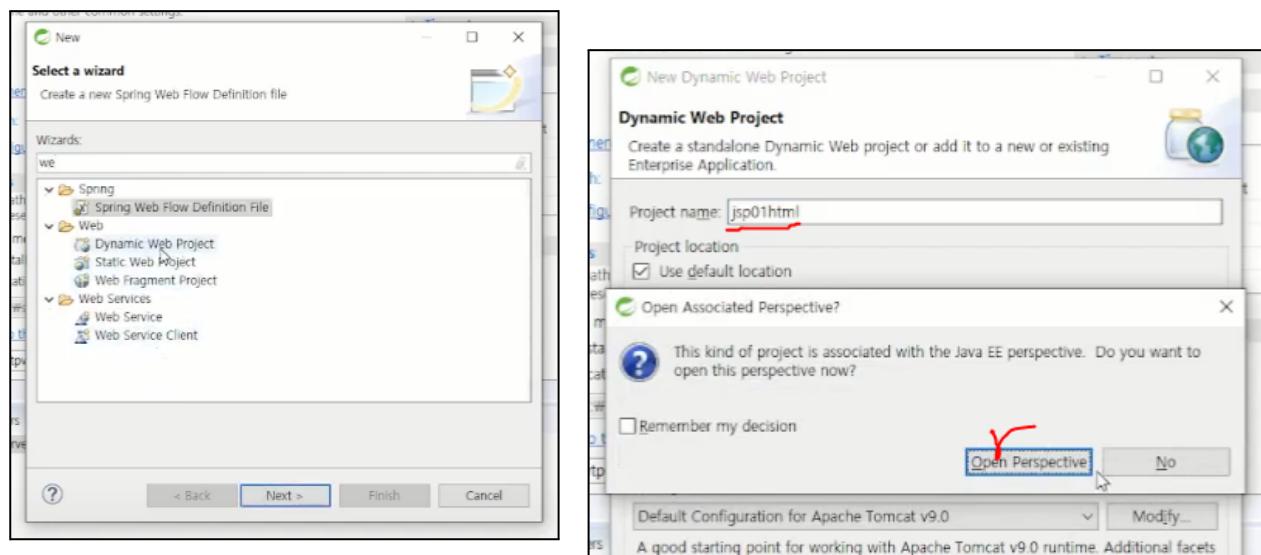
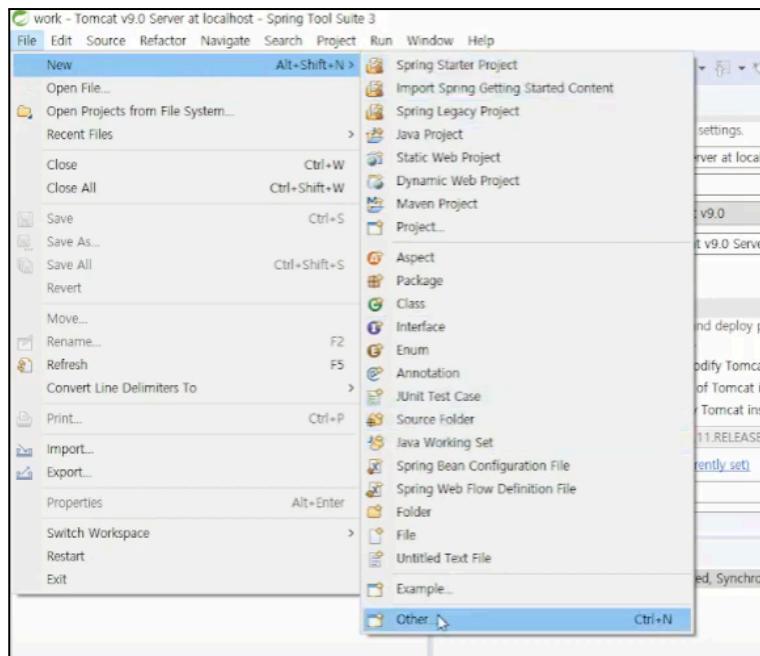
서버를 멈추고 싶다면 server << tomcat창에서 오른쪽 마우스를 클릭 한다음 stop메뉴를 선택하면 된다.

네트워크에서 본인 컴퓨터를 식별할 수 있는 ip주소를 얻고자 한다면 윈도우키+R를 누른 다음 cmd를 입력하고 엔터를 친 다음 콘솔창이 뜨면 ipconfig를 입력하면 본인 컴퓨터의 ip주소를 확인 할 수 있다.



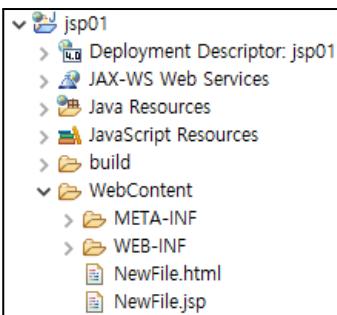
옆사람 컴퓨터에서 동작하고 있는 톰캣에 접속하고 싶다면 옆사람 ip주소와 톰캣에 설정한 포트 번호를 넣어주면 된다. ex) 192.168.0.30:8081 톰캣이 실행된 상태에서 요청해야 접근할 수 있다.

톰캣 설치가 마무리 되었다면 sts에서 web프로젝트를 만들어 볼 예정이다. package explorer 창을 열고 새 프로젝트를 만들어 보자. 다음 이미지 처럼 file >> new >> other를 선택한다음에 dynamic web project를 찾아서 선택해 보자.



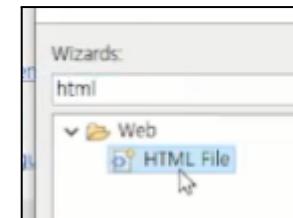
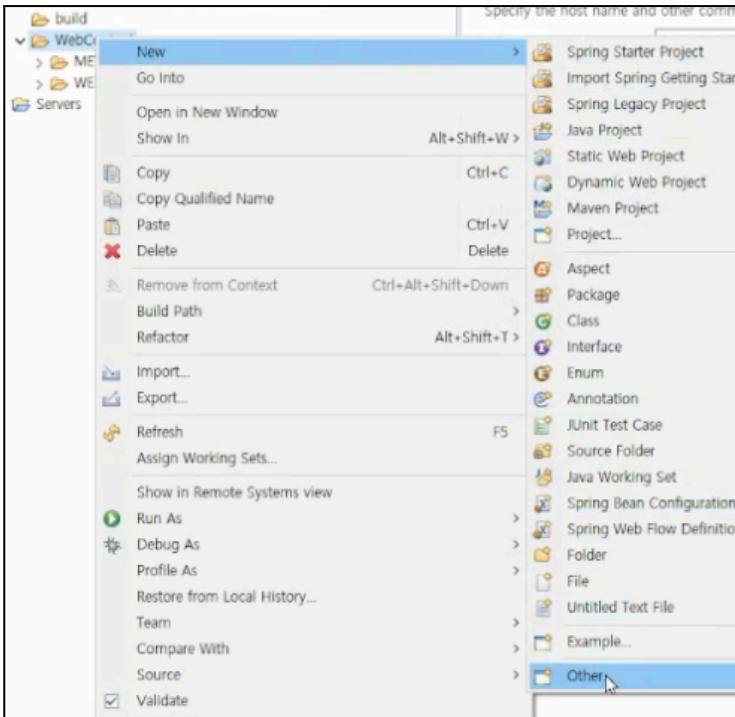
상위 오른쪽 이미지 처럼 창이 뜨면 java01html를 입력한다. 완료 버튼을 누르고 팝업 창이 뜨면 Open perspective버튼을 클릭한다.

다음 이미지 처럼 프로젝트 이름을 클릭 하고 src폴더 하위 폴더들을 열어 보면 다음과 같은 폴더 구조를 확인 할 수 있다. sts편집 툴에서 프로젝트를 작성하고 web server 톰캣에 작업한 코드를 넣어서 실행 할때 기준이 되는 root폴더가 webContent폴더가 된다.

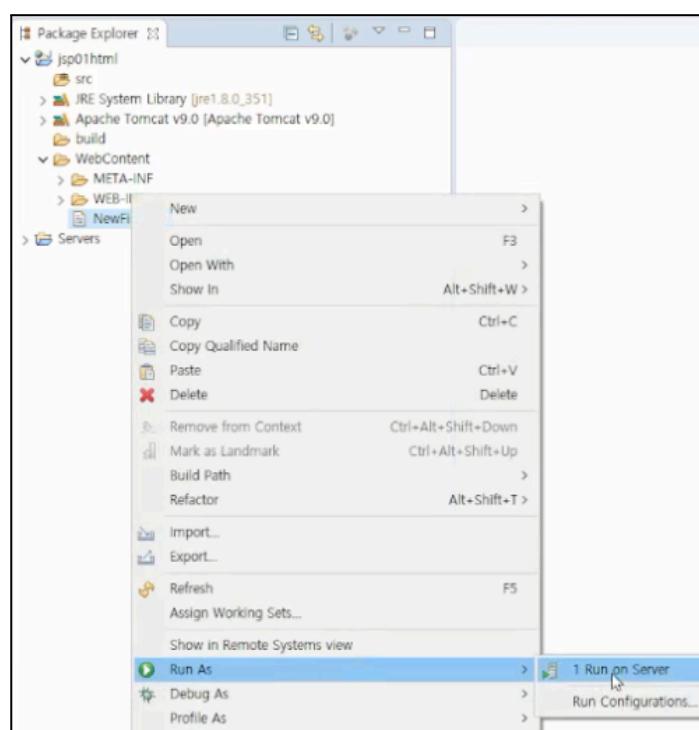


webContent를 기준으로 html를 작성하면 되는데 중요한점이 META-INF폴더와 WEB-INF폴더는 외부에서 접근할 수 없는 폴더이다. 종종 여기에 html파일을 만들어서 실행하면 찾을 수 없다는 에러가 난다 실수로 해당 폴더에 파일을 만들지 않도록 조심하자.

html파일을 하나 만들어 볼 예정이다. webContent폴더에서 하위 폴더를 만들고 싶으면 webContent폴더를 선택한 다음 new > folder를 선택하면 된다.



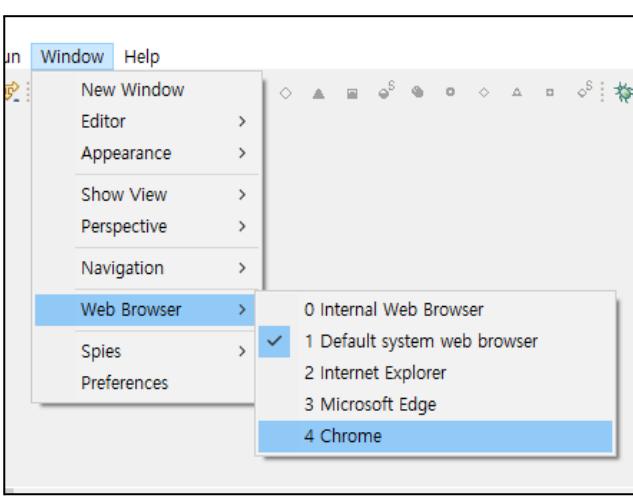
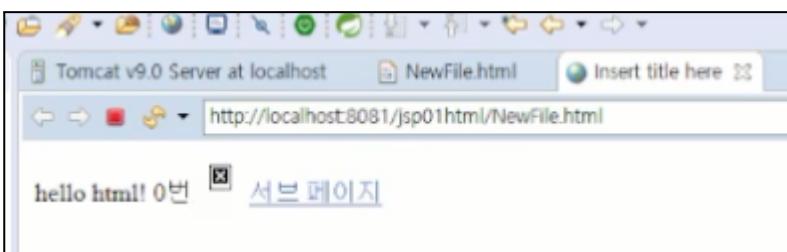
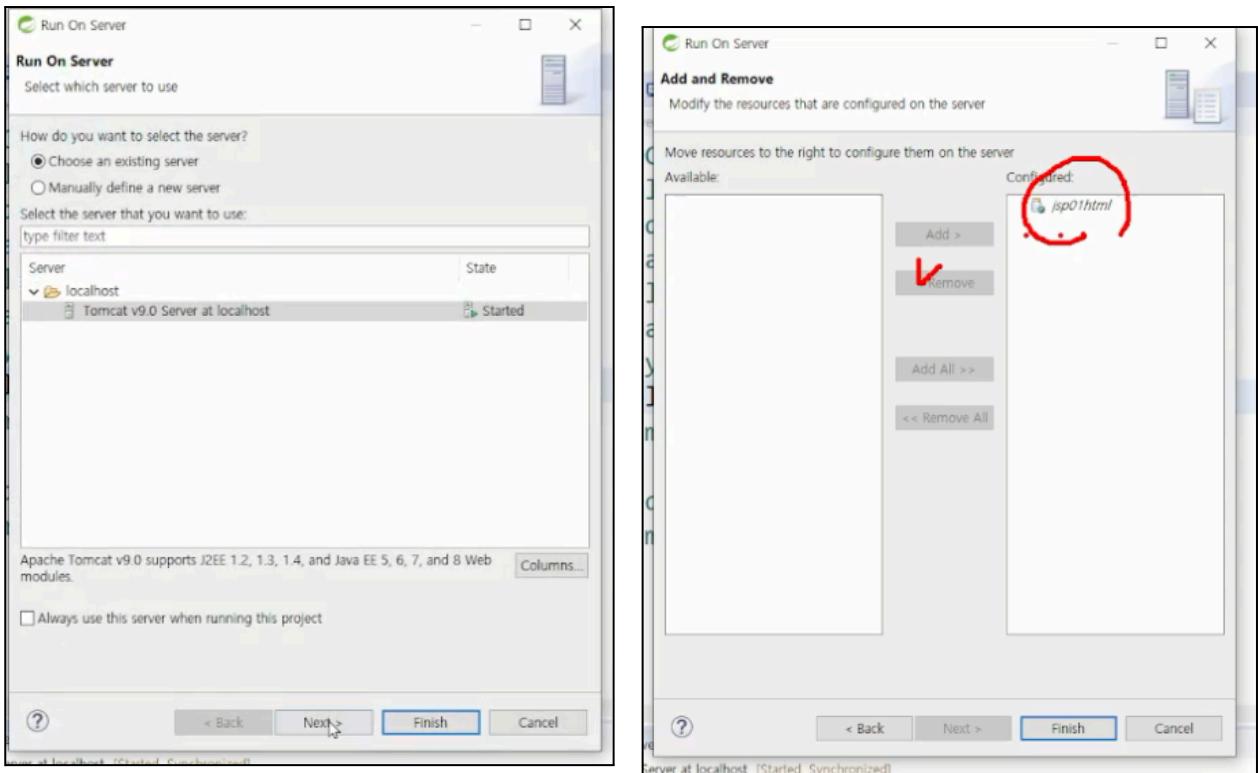
html파일을 만들고 싶으면 오른쪽 마우스를 클릭한 다음 new > other 를 선택한다음 html으로 검색해서 html file 메뉴를 선택하면 된다.



파일 이름을 넣고 실행하면 상의 이미지처럼 html파일이 만들어 지는데 charset= “UTF-8” 이 있는지 확인해 보자 없다면 sts.ini 파일 설정이 잘 안 된 것이다. ini설정 방법을 상위에서 참고해서 설정한 후 sts.exe를 다시 시작해 보자.

body 부분에 hello html! 를 입력후 저장 후 해당 html 파일에서 오른쪽 클릭 run as > run on server를 선택 한다.

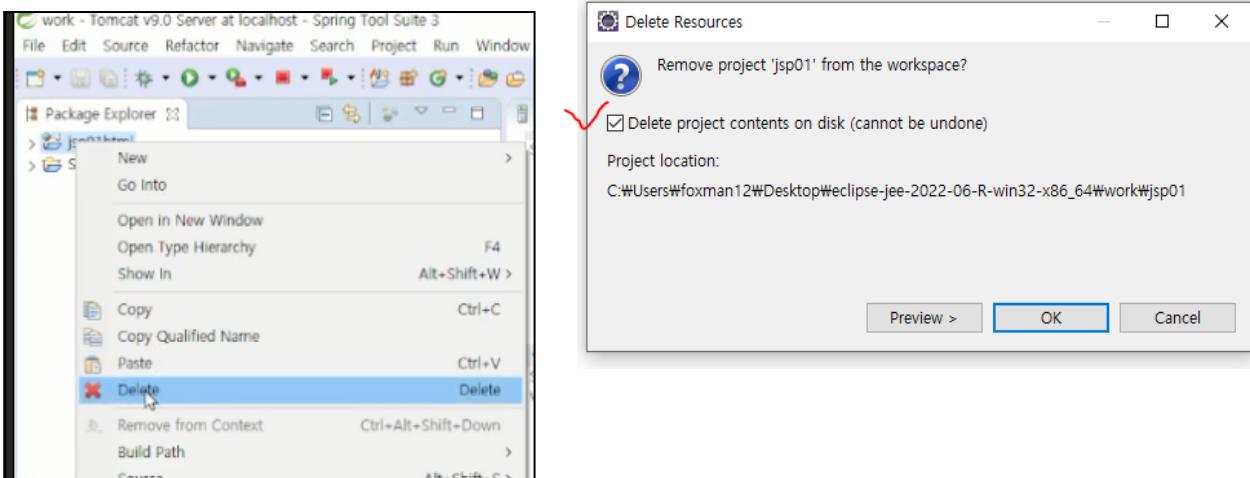
다음 페이지 왼쪽 이미지가 나오면 tomcat 9.0를 선택하고 next버튼을 누른다면 오른쪽 이미지처럼 모두 제거 버튼을 클릭해서 하나만 선택 되도록 만든 다음 finish버튼을 누른다. 여러개를 함께 실행 시키면 경로가 겹쳐서 문제가 될수 있으니 하나만 실행하는 습관을 가지자.



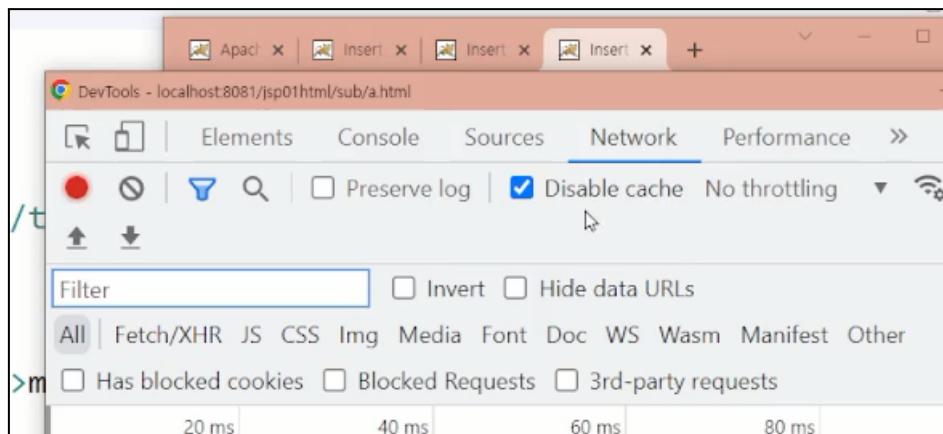
왼쪽 이미지처럼 실행 결과를 확인할 수 있는데 내부 브라우저로 실행 된다. 외부 브라우저 크롬을 사용하고 싶으면 다음 이미지처럼 window > web browser > chrome를 선택하면 실행 할때 내장 브라우저 대신 크롬을 사용할 수 있다.

내장 브라우저에 버그가 많으니 반드시 크롬을 사용하자.

생성한 프로젝트를 삭제하고 싶다면 다음 이미지처럼 해당 프로젝트 이름에서 오른쪽 마우스 클릭 delete를 선택한 다음 delete project... 항목에서 체크 박스를 반드시 체크하고 ok버튼을 누르면 삭제된다. 체크를 하지 않으면 sts에서만 사라지고 해당 폴더에는 파일이 남아 있어서 똑같은 이름으로 프로젝트를 만들면 문제가 발생한다.



웹프로그램을 구현하다 보면 수정하였으나 화면이 변경되지 않는 경우가 있다. 캐시를 사용하고 있기 때문이다. html를 변경한 즉시 적용시키고 싶으면 브라우저에서 f12를 누르고 아래처럼 network 탭에서 disable cache를 선택하면 변경한 내용이 바로 적용된다. 반드시 해당 창이 떠있어야 한다.



## > 02. html 작업하기

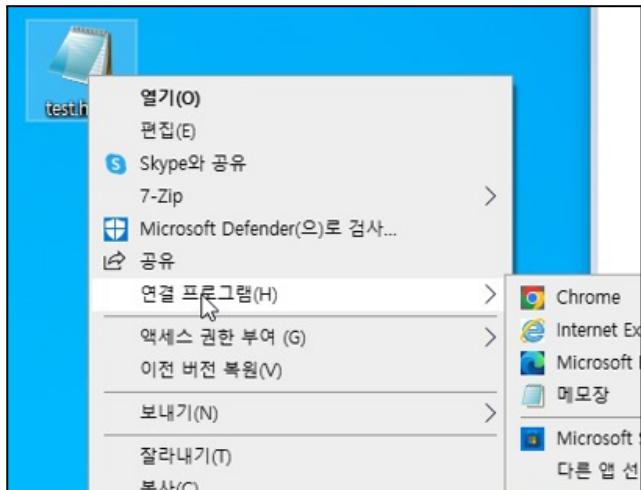
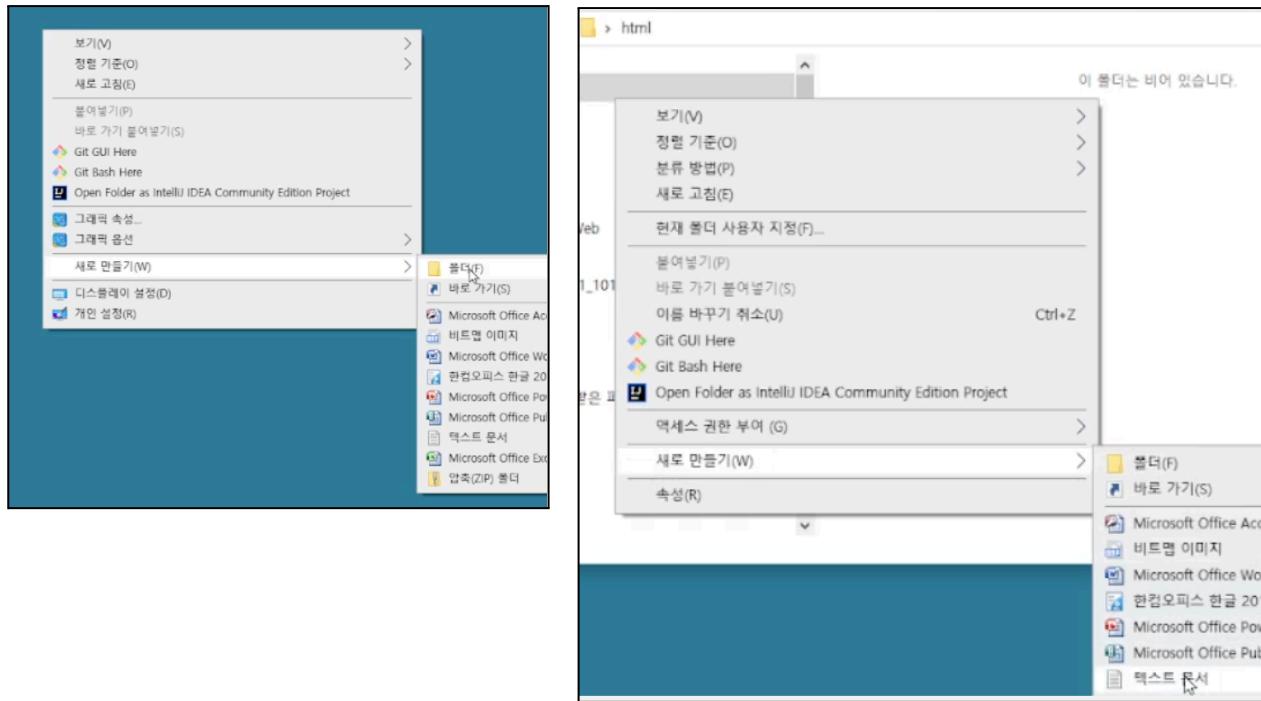
html은 브라우저로 네이버 사이트에 방문하면 브라우저에 네이버 사이트가 보이는데 이것을 html이라고 한다. 보통 사이트는 html+css+javascript로 되어 있다.

html은 내용과 구조에 해당하고 css는 색상 디자인에 해당하고 javascript는 움직이는 화면을 만드는 등의 일에 사용되는 브라우저에서 사용되는 프로그램 언어이다.

웹 개발자라면 반드시 알아야 하는 것들이다.

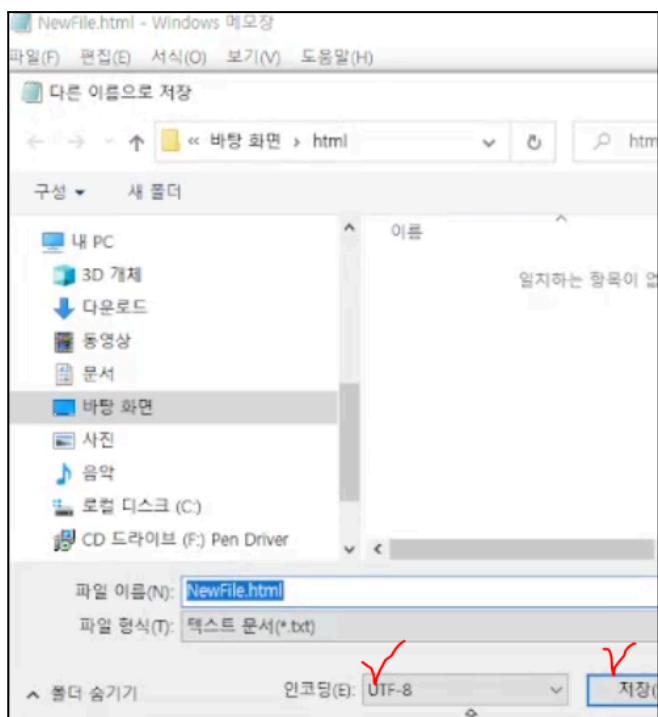
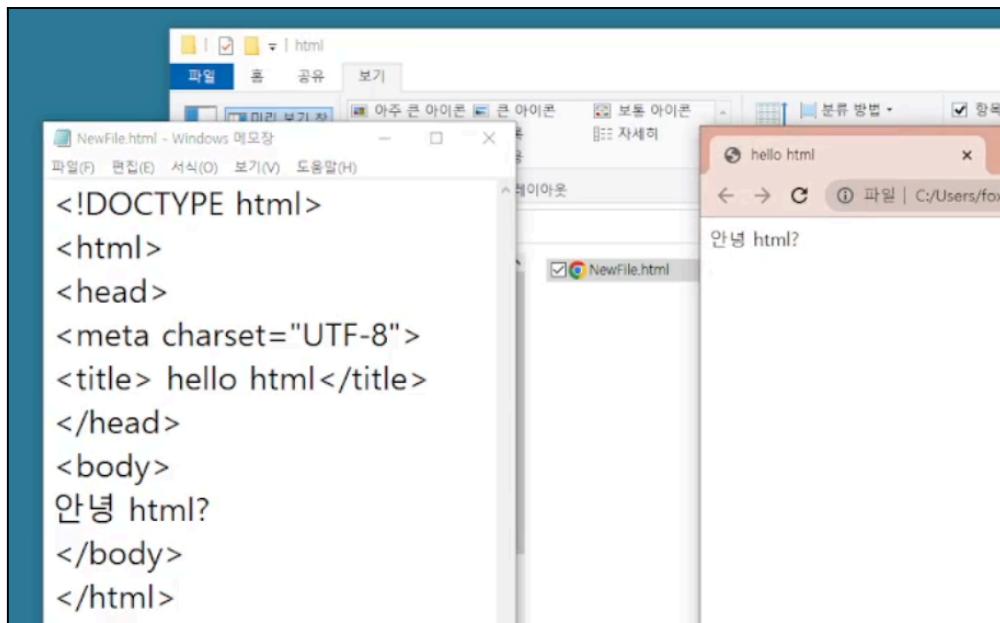
jsp는 사용자가 요청한 주소를 가지고 자바를 이용하여 html css javascript 파일을 만들어 요청한 사용자에게 보내 주는 작업을 한다.

간단하게 html 파일을 만들어 볼 예정이다.

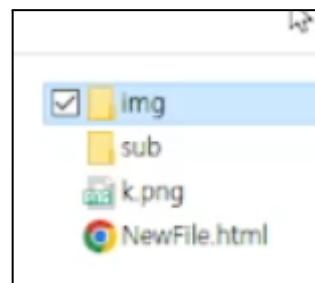


상위 왼쪽 이미지처럼 바탕화면에서 오른쪽 마우스 클릭을 하고 폴더를 선택한다. 폴더 이름을 html이라고 짓고 폴더를 더블 클릭해서 들어간다. 폴더 공간에 오른쪽 마우스 클릭 새로 만들기 텍스트 문서를 선택하고 문서 이름을 NewFile.html로 바꾼다. 확장자가 안보이면 확장자가 보이도록 설정하자. 이전 챕터에 설명이 있다. 해당 파일을 메모장으로 열려면 오른쪽 마우스 클릭 연결 프로그램을 메모장으로 선택하면 된다. 메모장이 열리면 다음 이미지처럼 입력하고 저장한 다음 마우스 오른쪽 프로그램 연결 프로그램

브라우저를 선택해서 브라우저로 열면 다음과 같다.



메모장 에디터로 html을 편집해서 프로그램을 구현한 다음 파일로 저장한 후 마우스 오른쪽 클릭 브라우저를 선택해서 실행하면 html이 해석되어 브라우저에 나타난다. 한글이 깨지면 왼쪽 이미지처럼 html를 저장할 때 파일 >> 다른 이름으로 저장 메뉴에서 인코딩을 UTF-8를 선택한 다음 저장하면 된다.



상위 이미지는 html폴더이다. 상위 이미지처럼 폴더를 구성하자. 이미지는 아무거나 하나 다운로드 받아 사용하자. 상위 이미지처럼 k.png이미지를 html폴더에 넣고

img라는 폴더를 만들고 안에 똑같은 이름의 이미지를 하나더 복사해 넣었다. sub폴더를 만들고 a.html를 만들었다. 두 html내용은 다음페이지와 같고 실행 결과 화면도 추가했다.

img 태그는 화면에 이미지를 보여주는 태그이고 src에 경로를 넣으면 된다.

경로로 같은 폴더에 있으면 k.png처럼 직접 쓰면 되고 하위 폴더로 가려면 img/k.png처럼 폴더 이름을 포함하여 기술하면 되고 ../를 이용해서 상위 폴더에 접근할 때 사용한다. body에 쓴 내용이 브라우저에 나타난다. a태그는 다른 페이지로 이동할때 사용하는 태그이고 href속성에 정의 된 html파일로 해당 태그를 클릭하면 이동한다.

```

NewFile.html - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> hello html</title>
</head>
<body>
안녕 html?
<br>
<img src='k.png' width="100">
<img src='img/k.png' width="100">
<br>
<a href="sub/a.html">서브페이지이동</a>
</body>
</html>

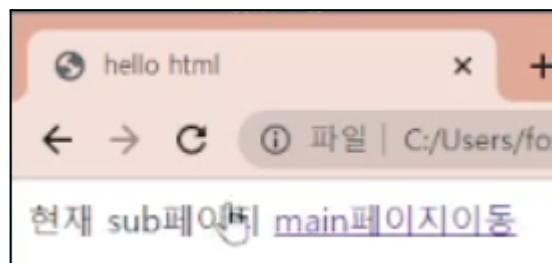
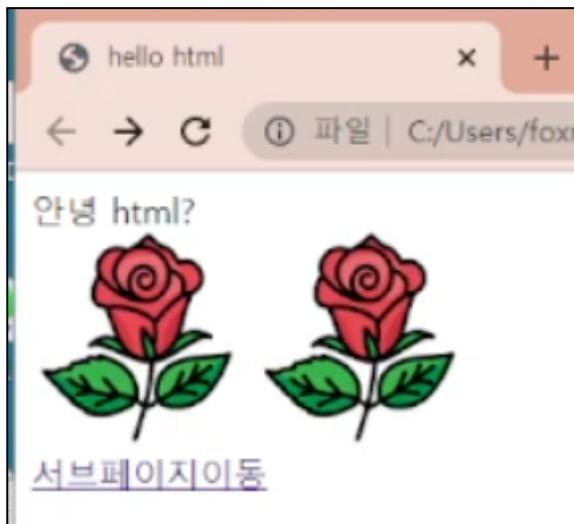
```

```

*a.html - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> hello html</title>
</head>
<body>
현재 sub페이지
<a href="../NewFile.html">main페이지이동</a>
</body>
</html>

```

완성후 NewFile.html를 클릭하면 아래 왼쪽 이미지처럼 나온다. 서브페이지 이동을 클릭하면 서브 페이지가 나오고 main 페이지 이동을 클릭하면 main페이지로 이동한다.

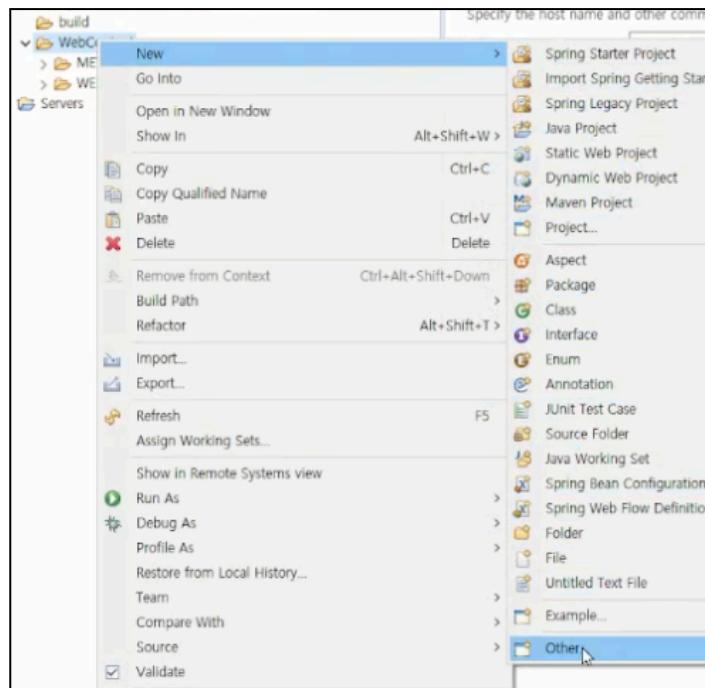


## > 01. html, jsp, servlet page 만들기

클라이언트 서버란? 데이터를 요청하는 컴퓨터를 클라이언트라 하고 데이터를 제공해주는 컴퓨터를 서버라 한다.

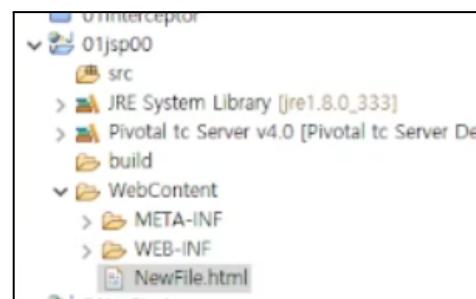
웹 서버는 인터넷에서 클라이언트가 요청한 주소를 확인해서 해당하는 html 웹 페이지를 제공해 주는 일을 한다. 이전에 설치하고 실행해 본 톰캣이 웹서버에 해당 한다.

html, jsp, servlet를 웹서버를 실행해서 웹 서버에 올려 놓으면 사용자가 요청한 주소에 해당하는 html파일을 제공해 준다. jsp, servlet은 결국 자바로 html문서를 만들어서 html를 제공해 준다. 결국 서버는 html만 제공 한다. 3가지 문서를 툴에서 만드는 방법을 확인해 볼 예정이다.

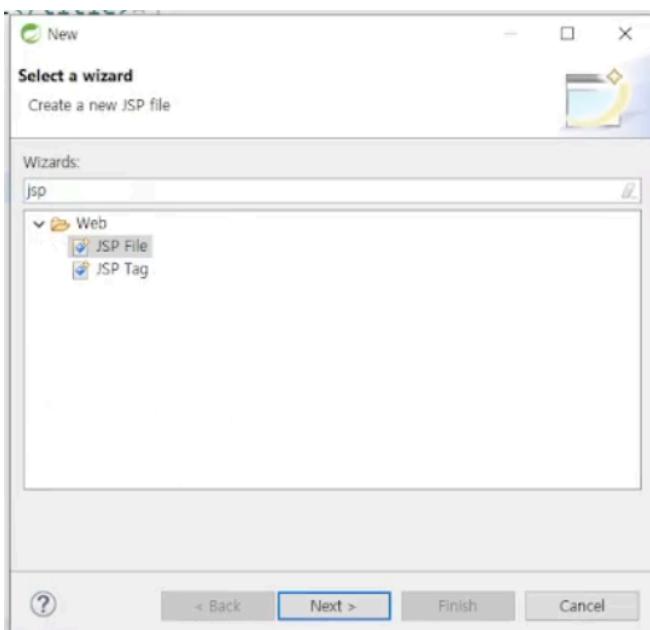


### 1. html 만들기

webContent 폴더에서 오른쪽 마우스클릭 new >> other 를 선택한다음 검색창에 html를 넣어서 html파일을 찾아 선택하면 다음 이미지 처럼 html파일이 생성 된다.

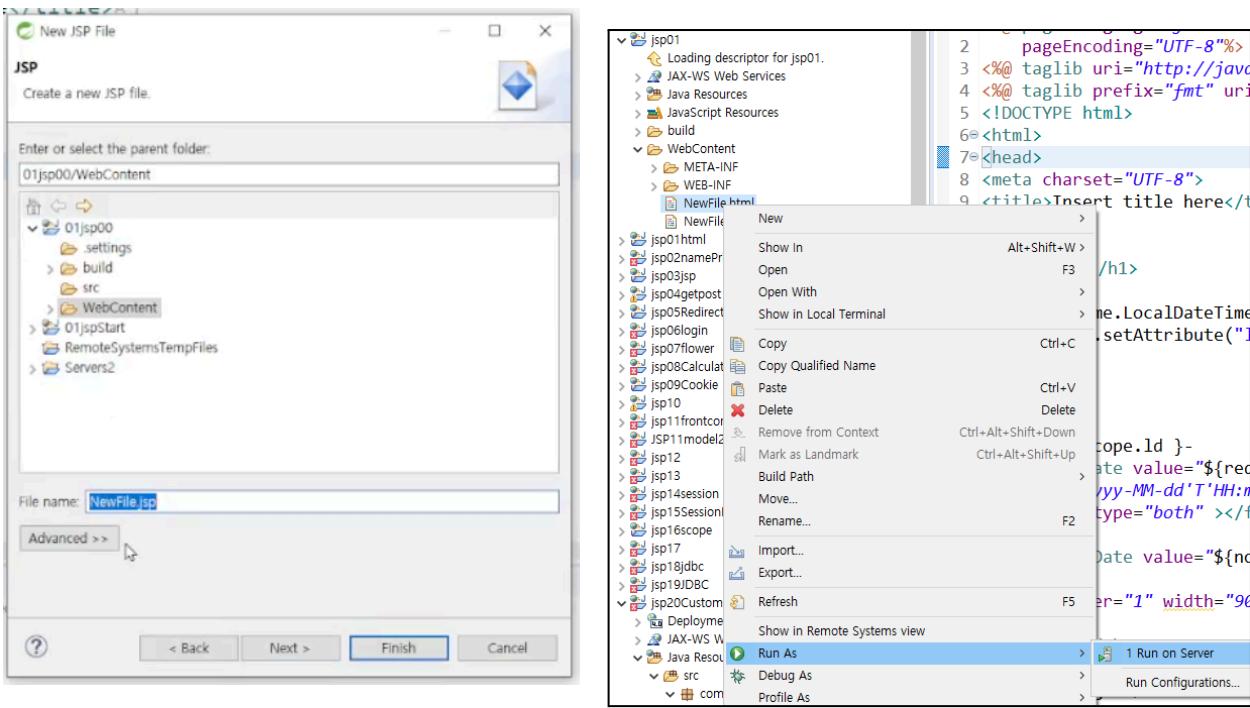


이전에 만들어 봤으니 코딩은 생략할 예정이다.



### 2. jsp 만들기

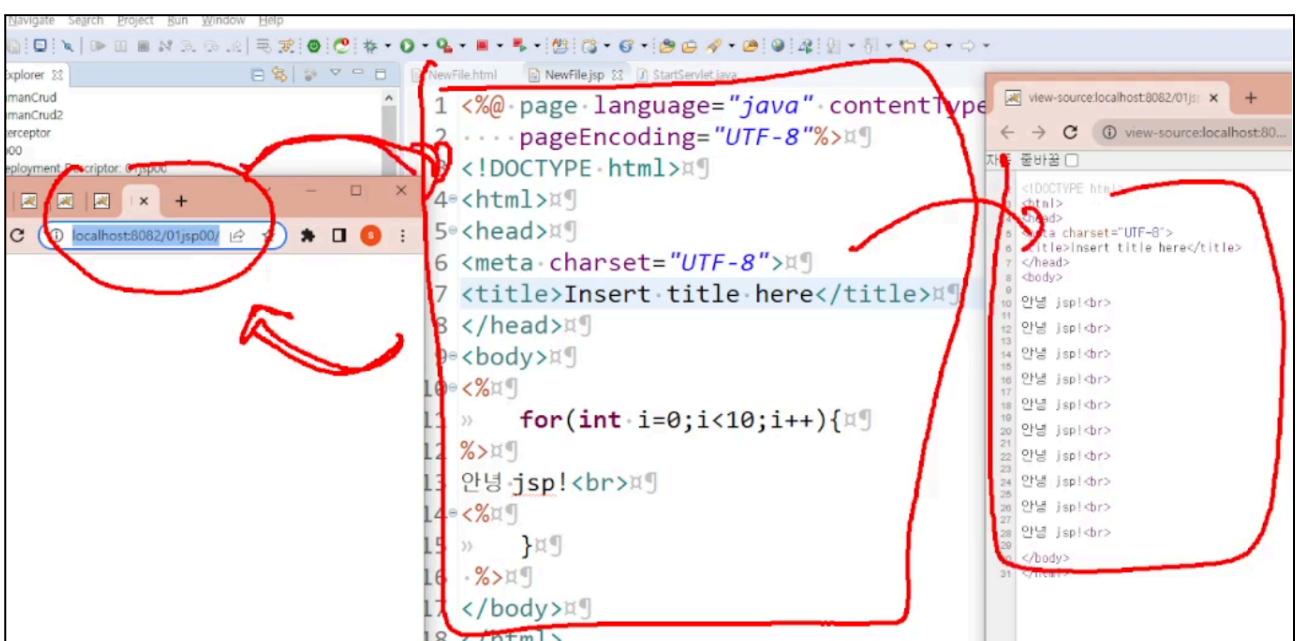
webContent폴더에서 오른쪽 마우스클릭 new >> other를 선택한 다음 jsp를 검색해서 jsp파일을 선택한 다음 NewFile.jsp파일이름을 입력한 다음 finish를 클릭하면 해당 파일이 만들어 진다.

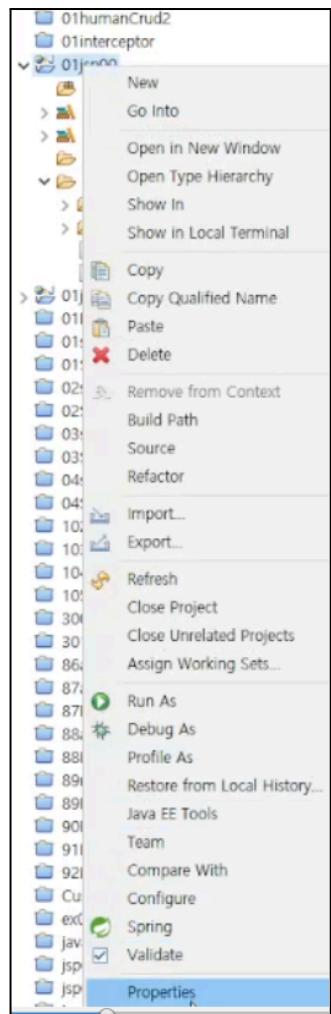


<% %> 스크립틀릿을 이용해서 아래 이미지 처럼 코딩한 후 만든 jsp파일 위에서 상위 왼쪽 이미지 처럼 마우스 오른쪽 클릭 >> run as >> run on server를 이용해서 해당 jsp파일을 실행하면 안녕 jsp!가 여러 개 찍힌것을 확인할 수 있다. 파일을 실행 한다는 의미는 본인이 틀에서 작업한 코드를 웹서버에 옮겨서 브라우저로 요청해서 확인한다는 의미가 된다. 이런 작업을 실제 서버에서는 일일이 작성한 파일을 서버에 올려야 하는데 자동으로 해준다고 생각 하면 된다. 실행 되는 것이 확인 되었다면 이전에 만든 html도 제대로 동작하는지 실행해 보자.

아래 이미지는 사용자가 만든 웹서버를 실행 시켜 주소창에 해당 jsp주소를 입력하면 주소에 해당하는 jsp 파일이 html로 만들어져서 요청한 브라우저에 전송하여 보내 준다.

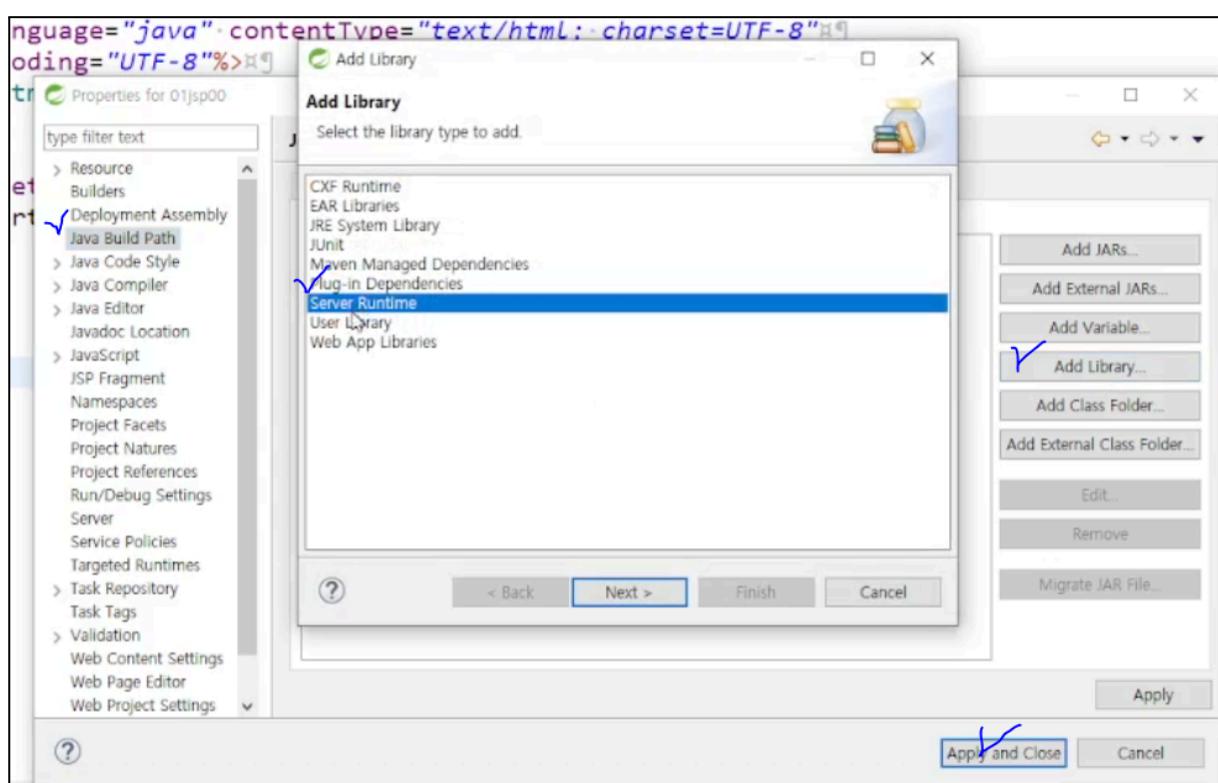
이모든 작업이 틀에서 해당 파일을 실행하면 자동으로 이루어 진다. 서버 실행후 웹서버에 올려 놓은 파일들은 모두 적절한 파일 이름 주소를 넣어서 웹서버에 요청할 수 있다.

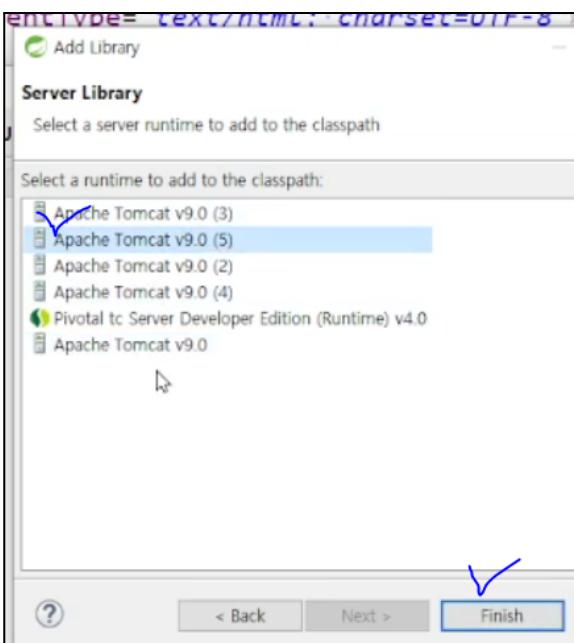




사용자가 jsp를 요청하면 자바 코드가 html파일을 만들어서 사용자에게 제공해 준다. 브라우저에서 오른쪽 마우스 클릭 페이지 소스 보기 선택하면 생성된 html을 확인 할 수 있다. 종종 jsp 파일을 <% 스크립 틀릿 부분에 여러 모양이 생길 때가 있는데 tomcat라이브러리가 등록되어 있지 않아서이다. 다음과 같은 방법으로 톰캣을 등록할 수 있다. 등록 방법은 다음과 같은 방법이 있다. 프로젝트에서 오른쪽 마우스를 클릭한 다음 맨아래 properties 메뉴를 클릭한다.

다음과 같은 메뉴가 나오면 java build path >> add library >> server runtime를 선택한 다음 next를 클릭하면 다음 이미지가 나온다. 등록되어 있는 tomcat중 하나를 선택한 다음 finish를 누르면 문제가 해결 된다. 대부분 한개의 톰캣이 등록 되어 있으나 아닐 수 도 있다.

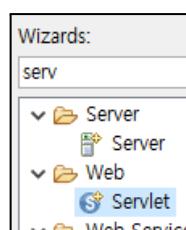




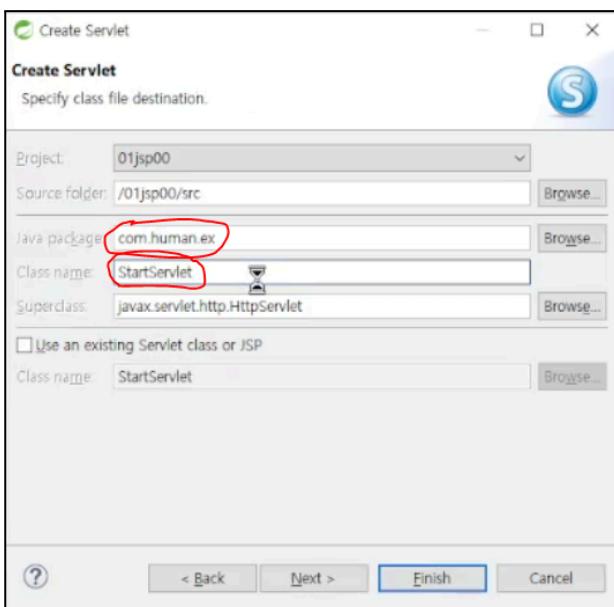
### 3. servlet 만들기

```

1 package com.human.ex;
2
3 import java.io.IOException;
11
12 /**
13 * Servlet implementation class StartServlet
14 */
15 @WebServlet("/StartServlet")
16 public class StartServlet extends HttpServlet {
17     private static final long
    
```



상위 이미지의 src폴더에서 마우스 오른쪽 클릭 new > other 를 선택한 다음 왼쪽 이미지와 같은 창에서 servlet를 검색해서 servlet를 선택한다.



왼쪽 이미지 처럼 입력창이 뜨면 java package는 com.human.ex class name은 StartServlet 를 입력 한다.

생성된 파일을 확인해 보면 상단에 @WebServlet("/StartServlet") 부분이 있는데 이부분이 해당 서블릿을 실행하고자 할때 사용하는 주소이다. 상위 이미지에서 확인해 보자.

기본적으로 만들어진 서블릿 자바 파일에는 모르는 내용이 많이 있겠지만 신경쓰지 말고 본인이 필요한 부분을 찾아서 코딩을 하면 된다.

주요 메소드로 다음 이미지처럼 doGet과 doPost가 있는데 사용자가 get방식으로

요청하면 doGet메소드가 post방식으로 요청 하면 doPost를 요청할 수 있는 방법은 없다. 나중에 배울 예정이니 일단 사용자 요청을 처리하고 싶다면 doGet메소드에 기술하면 된다.

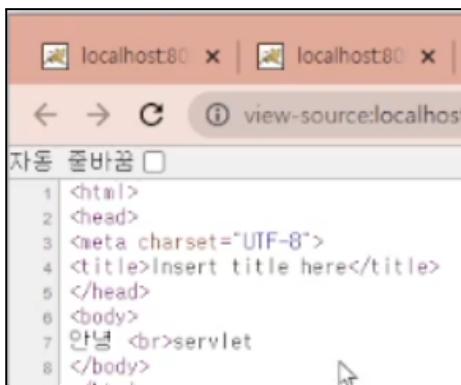
```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    // TODO Auto-generated method stub
    response.getWriter().append("Served at: ").append(request.getContextPath());
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    // TODO Auto-generated method stub
    doGet(request, response);
}
    
```

HTTP(하이퍼텍스트 전송 프로토콜)는 웹 상에서 클라이언트, 서버가 데이터(html)를 주고 받기 위한 통신 규약 중 하나입니다. 두개의 메시지로 구성되어 있는데 요청할때 request, 응답할때 사용하는 response을 사용한다. 다음 코드를 살펴보면 클라이언트에서 서버쪽에 가는 데이터는 request를 사용하고 서버에서 클라이언트로 가는 데이터는 response를 사용 한다. doGet부분을 다음과 같이 변경해보고 실행해 보자.

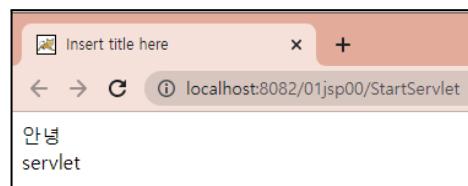
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("들어오면 콘솔창에 찍힘");
    //전송할 데이터 종류기술
    response.setContentType("text/html;charset=UTF-8");
    //클라이언트에게 전송할 out객체를 얻어와 println를 이용해서 전송해주는 작업
    PrintWriter out=response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<meta charset=\"UTF-8\">");
    out.println("<title>Insert title here</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("안녕 <br>servlet");
    out.println("</body>");
    out.println("</html>");
    out.close(); //사용이 끝나면 종료한다.
}
```



A screenshot of a web browser window titled 'localhost80'. The address bar shows 'view-source:localhost80'. The main content area displays the source code of a JSP page:

```
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
안녕 <br>servlet
</body>

```

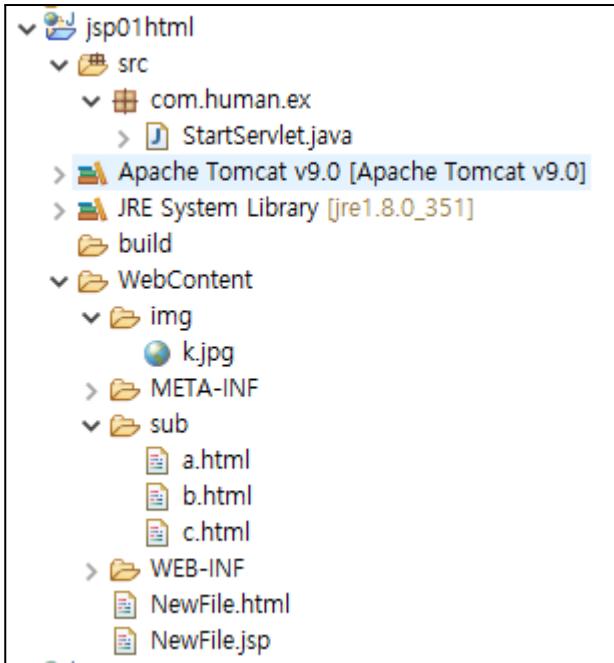


실행 방법은  
servlet파일에서  
마우스 오른쪽 클릭  
해서 run as >> run  
on server를

사용해서 실행한다.  
서블릿 실행 결과는 상위 이미지와 같고 브라우저에서  
소스 보기하면 왼쪽과 같이 html로 생성되어 있는 것을  
확인할 수 있다.

결국 웹서버에 있는 html, jsp, servlet파일은 사용자가 요청하면 html파일로 변환하여  
요청자에게 제공한다.

작업중 404, 500 에러가 날수 있다. 404에러는 요청한 주소가 서버에 없을때 생긴다.  
브라우저의 주소창에 주소를 확인해 보자. web-info폴더는 외부에서 접근할 수 없는  
폴더다. 해당 폴더에 파일을 만들어 주소로 요청하면 경로가 맞아도 에러를 생성한다.  
500에러는 문법 에러이다 자바 문법이 잘못된 곳이 있는지 확인해 보자.



왼쪽 이미지는 html,jsp,servlet를 이용해서 출력하는 페이지이다. 3가지 방식으로 만들어서 실행해보자.

3가지 방식 모두 webContent폴더를 기준으로 해서 상대경로가 생성된다. servlet으로 만든 자바 파일도 이미지 같은 리소스에 접근하려 할때 기준은 webContent이다.  
img폴더를 만들고 웹에서 아무 이미지 하나를 다운로드 해서 img/k.jpg로 넣는다.

1. html 를 사용해 보자.

WebContent/sub/a.html

NewFile.html

상위 2개의 파일로 구성되어 NewFile.html을 실행 시키면 서브페이지로 a.html로 이동하고 a.html에서 NewFile.html파일로 이동한다.

NewFile.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    hello html! 0번
    <a href='sub/a.html'>서브 페이지</a>
    <img src='img/k.jpg'>
</body>
</html>
```

sub/a.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    html sub page
    <a href="../NewFile.html">mainPage</a>
    
</body>
</html>
```

상위와 같이 만든 다음 NewFile.html를 실행 시키면 이전에 만든 html폴더안에서 동작하는 것과 동일하게 web서버에서 동작하게 된다.

html, jsp, servlet파일 모두 다 webcontent 폴더를 기준으로 파일 경로가 만들어 진다.  
NewFile.html로 만든 파일을 다음을 참고해서 NewFile.jsp와 StartServlet.java로 만들어서 각각에서 NewFile.html과 동일하게 동작하도록 만들어 보자.  
html의 서브페이지는 sub/a.html, jsp의 서브페이지는 sub/b.html, servlet의 서브페이지는 sub/c.html로 만들자.

## 2.jsp

WebContent/sub/b.html

NewFile.jsp

상위 2개의 파일로 구성되어 NewFile.jsp를 실행 시키면 서브페이지로 b.html로 이동하고 b.html에서 NewFile.jsp파일로 이동한다.

```
//NewFile.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    for(int i=0;i<5;i++){
%>
    안녕하세요<br>
<%} %>
<%
    out.println("hello jsp?<br>");
%>
    hello jsp! 0번
    <a href='sub/b.html'>서브 페이지</a>
    <img src='img/k.jpg'>
</body>
</html>
// sub/b.html
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    jsp sub page
    <a href="../NewFile.jsp">mainPage</a>
    
```

```
</body>
</html>
```

3.servlet으로 만들어 보자.

WebContent/sub/c.html

src/com.human.ex.StartServlet.java

상위 2개의 파일로 구성되어 StartServlet.java를 실행 시키면 서브페이지로 c.html로 이동하고 c.html에서 StartServlet.java파일로 이동한다.

서블릿 주소 매핑시 다음과 같은 경우 문제가 된다.

- 1.@WebServlet("myname") /으로 시작해야 한다.
- 2.@WebServlet("/\*.do") 여러 개의 주소를 하나로 받을 경우 /를 생략해야 한다.
- 3.같은 주소 매핑이 2개 이상 존재하면 404 에러가 발생 한다.

```
//StartServlet.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/StartServlet")
public class StartServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public StartServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.println("화면에 찍힘");
        //브라우저 스트림에 문자열을 찍으면 브라우저에 나타나다.
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("hello servlet1<br>");
        int sum=0;
        for(int i=1;i<10;i++) {
            out.println(String.format("sum=%d<br>", sum=sum+i));
        }
        out.println("<img src='img/k.jpg'>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
}
}

//sub/c.html
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
servlet sub page
<a href="../StartServlet">mainPage</a>

</body>
</html>
```

servlet안의 html에서 사용하는 경로 기준은 webContent입니다.

다음 3개는 상대경로이다.

현재 내용을 담고 있는 파일의 상위폴더에 있는 k.jpg 파일을 의미함  
현재 내용을 담고 있는 파일의 img파일 상위폴더에 있는 k.jpg  
현재 내용을 담고 있는 파일과 같은 위치에 있는 파일

다음은 절대 경로이다.

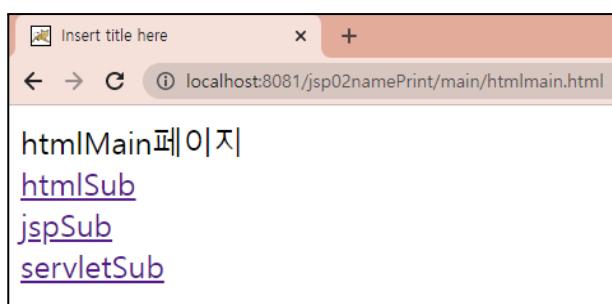
 현재 내용을 담고 있는 파일의 서버 경로 + /k.jpg 파일 이름을 실제 경로에 해당한다. ex) <http://localhost:8082/k.jpg>

## > 02. html,jsp,servlet 경로 이해하기

다음 예제를 실습해보자.

3가지 방법(html,jsp,servlet)으로 메인페이지 3개와 서브페이지 3개를 만든 다음 각각의 메인 화면에서 각각의 서브 화면으로 갈 수 있고 각각의 서브파일에서 각각의 메인 화면으로 갈 수 있는 페이지를 만든다.

### 1. 각각의 메인페이지 만들기



왼쪽 이미지는 htmlMain페이지이다.

<http://localhost:8081/jsp02namePrint/main/htmlmain.html> 과

같은 주소를 가진다.

htmlMain페이지에서 htmlSub 링크를 클릭하면 htmlSub페이지로 이동한다.

main과 sub 폴더에 html,jsp파일이 구성되어 있음을 확인해 보자.

The screenshot shows the Eclipse IDE interface with three panes:

- Package Explorer:** Shows the project structure:
  - jsp01html
  - jsp02namePrint
    - src
      - com.human.ex
        - MyName.java
      - com.human.main
        - ServletMain.java
    - JRE System Library [jre1.8.0\_35]
    - Apache Tomcat v9.0 [Apache]
    - build
    - WebContent
      - main
        - htmlmain.html
        - jspmains.jsp
      - META-INF
      - sub
        - myname.html
        - myname.jsp
    - WEB-INF
- Browser 1:** Address: localhost:8081/jsp02namePrint/main/htmlmain.html. Content: htmlMain페이지  
[htmlSub](#)  
[jspSub](#)  
[servletSub](#)
- Browser 2:** Address: localhost:8081/jsp02namePrint/main/htmlmain.html. Content: servletMain 페이지  
[htmlSub](#)으로 이동  
[jspSub](#)으로 이동  
[servletSub](#)으로 이동

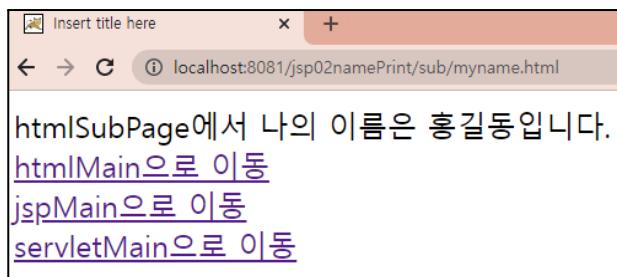
### 2. 각각의 sub페이지 만들기

3가지 방법(html,jsp,servlet)으로 본인 이름 출력 하는 sub 페이지를 만들고 각각의 메인 페이지에서 3개의 sub페이지로 이동하고 각각의 서브페이지에서 각각의 메인 페이지로 이동하도록 웹사이트를 만들어 보자.

//프로젝트 이름은 [jsp02namePrint](#)로 하자.

//총페이지 개수는 6개가 된다.

다음은 각각의 서브페이지 화면이다.

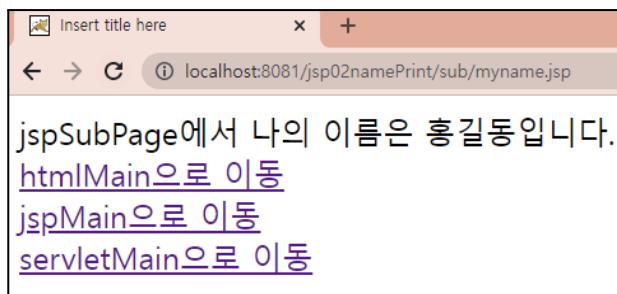


왼쪽 이미지는 htmlSubPage페이지이다.

<http://localhost:8081/jsp02namePrint/sub/mynname.html>

과 같은 주소를 가진다.

htmlMain으로 이동 링크를 클릭하면 이전 htmlMain페이지로 이동한다.



왼쪽 이미지는 jspSubPage페이지이다.

<http://localhost:8081/jsp02namePrint/main/jspmain.jsp>와 같은 주소를 가진다.

htmlMain으로 이동 링크를 클릭하면 이전 htmlMain페이지로 이동한다.



왼쪽 이미지는 ServletSubPage페이지이다.

<http://localhost:8081/jsp02namePrint/sub/mynname.html>와

같은 주소를 가진다. htmlMain으로 이동 링크를 클릭하면 이전 htmlMain페이지로 이동한다.

jsp와 servlet으로도 관리 페이지를 만들면 총 6개의 화면이 만들어 진다.

다음은 최종 결과물이다. 스스로 작성해 보고 확인해 보자.

```
//com.human.ex.MyName.java //servleSub 페이지
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
//@WebServlet("sub/mynname") 중간에 /를 사용하면 안된다.
//@WebServlet("mynname") /으로 시작해야 한다.
@WebServlet("/mynname")
public class MyName extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public MyName() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

```

        System.out.println("화면에 찍힘");
        //브라우저 스트림에 문자열을 찍으면 브라우저에 나타난다.
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("servletSubPage에서에서 나의 이름은 흥길동 입니다.<br>");
        out.println("<a href='main/htmlmain.html'>htmlMain으로 이동</a><br>");
        out.println("<a href='main/jspmain.jsp'>jspMain으로 이동</a><br>");
        out.println("<a href='servletmain'>servletMain으로 이동</a><br>");
        out.println("</body>");
        out.println("</html>");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}

//com.human.ex.main.ServletMain.java //servlet main파일
package com.human.main;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/servletmain")
public class ServletMain extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public ServletMain() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        System.out.println("화면에 찍힘");
        //브라우저 스트림에 문자열을 찍으면 브라우저에 나타난다.
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("servletMain 페이지<br>");
        out.println("<a href='sub/myname.html'>htmlSub</a>으로 이동<br>");
        out.println("<a href='sub/myname.jsp'>jspSub</a>으로 이동<br>");
        out.println("<a href='myname'>servletSub</a>으로 이동<br>");
        out.println("</body>");
    }
}

```

```

        out.println("</html>");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request, response);
    }
}

// main/html      main.html  //html main페이지
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
htmlMain페이지<br>
<a href="../sub/myname.html">htmlSub</a><br>
<a href="../sub/myname.jsp">jspSub</a><br>
<a href="../myname">servletSub</a><br>
</body>
</html>

//main/jspmain.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
jspMain페이지<br>
<a href="../sub/myname.html">htmlSub</a><br>
<a href="../sub/myname.jsp">jspSub</a><br>
<a href="../myname">servletSub</a><br>
</body>
</html>

// sub/myname.html  //html sub페이지
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>

```

```

</head>
<body>
    htmlSubPage에서 나의 이름은 흥길동입니다.<br>
    <a href="../main/htmlmain.html">htmlMain으로 이동</a><br>
    <a href="../main/jspmain.jsp">jspMain으로 이동</a><br>
    <a href="../servletmain">servletMain으로 이동</a><br>
</body>
</html>

//sub/myname.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    jspSubPage에서 나의 이름은 흥길동입니다.<br>
    <a href="../main/htmlmain.html">htmlMain으로 이동</a><br>
    <a href="../main/jspmain.jsp">jspMain으로 이동</a><br>
    <a href="../servletmain">servletMain으로 이동</a><br>
</body>
</html>

//myname.jsp      //jsp sub페이지
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    jspSubPage에서 나의 이름은 흥길동입니다.<br>
    <a href="../main/htmlmain.html">htmlMain으로 이동</a><br>
    <a href="../main/jspmain.jsp">jspMain으로 이동</a><br>
    <a href="../servletmain">servletMain으로 이동</a><br>
</body>
</html>

//jspmain.jsp  jsp페이지
<!DOCTYPE html>
<html>

```

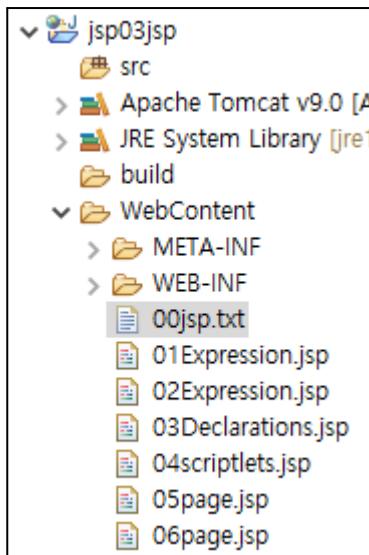
```

<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
htmlMain페이지|<br>
<a href="../sub/myname.html">htmlSub</a><br>
<a href="../sub/myname.jsp">jspSub</a><br>
<a href="../myname">servletSub</a><br>
</body>
</html>

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
jspMain페이지|<br>
<a href="../sub/myname.html">htmlSub</a><br>
<a href="../sub/myname.jsp">jspSub</a><br>
<a href="../myname">servletSub</a><br>
</body>
</html>

```

## > 03. jsp 문법 정리



<!-- --> //html 주석  
<%-- --%> //jsp 주석  
// /\* \*/ 자바주석

jsp 태그는 3가지  
표현식 <%= %> 화면에 출력  
선언식 <%! %> 전역변수나 메소드 선언  
스크립틀릿 <% %> 지역변수나 java로직 사용

jsp 태그 문법들 보다는 나중에 배울 el과 jstl을 사용하니  
다음 예제를 확인해서 간단히 어떻게 사용하는 확인해 보자.

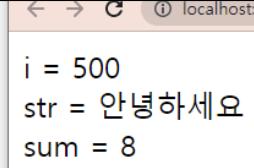
```
// 01Expression.jsp 다음 표현식 예제를 확인해 보자.  
<%@ page language="java" contentType="text/html; charset=UTF-8"  
    pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="UTF-8">  
<title>Insert title here</title>  
</head>  
<body>  
<%!  
    String str1="MyString1<br>";//전역변수 선언  
    public String strFunction(){//함수 선언  
        return "<br>functionString<br>";  
    }  
%>  
<%  
    String str2="MyString2<br>";//지역변수 선언  
%>  
<%  
    out.print(str1);  
    out.print(str2);  
/*  
    나눠서 여러개로 기술할수 있다. ; 사용하지 않음  
*/  
%>  
<%=str1 %>  
<%=str2 %>
```

```

<%"직접출력<br>" %>
<%=50 %>
<%=50+20 %>
<%=strFunction() %>
</body>
</html>

```

문제1) 다음과 같이 출력 되도록 jsp코드들을 추가하고 아래의 코드를 표현식으로 변경해보자.

<pre> &lt;%     out.print("i = "+i +&lt;br&gt;);     out.print("str = "+str +&lt;br&gt;);     out.print("sum = "+sum(3,5) +&lt;br&gt;); %&gt; </pre>	
--	--

```

//02Expression.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<% int i=10; %>
<% String str="ABCDE"; %>
<%!
public int sum(int n1, int n2){
    return n1+n2;
}
%>
<%
    out.print("i = "+i +<br>);
    out.print("str = "+str +<br>);
    out.print("sum = "+sum(3,5) +<br>);
%>
<%="i = "+i +<br>%>
<%="str = "+str +<br>%>
<%="sum = "+sum(3,5) +<br>%>
</body>
</html>

```

다음은 선언 문관련 예제이다.

다음 코드를 확인해서 선언문에서 선언된 변수와 스크립틀립에서 선언된 변수의 차이를 이해해 보자. 선언문에 선언하면 java static과 같아서 여러 사람이 같은 사이트에 들어 올

때 변수 값을 공유한다. 스크립트릿에 선언한 변수는 java 지역 변수와 같아서 각각의 사용자가 들어올 때마다 값이 초기화된다. 결과적으로 `str1=str1+"1";` 의 결과로 `str1`의 결과는 “1111111” 형태로 새로 고침 할 때마다 하나씩 증가하고 `str2=str2+"1";`은 항상 “11”이 `str2`의 값이다.

```
//03Declarations.jsp 다음 선언문 예제를 확인해 보자.
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
// 클래스 멤버로 사용되 함수나 클래스 변수를 선언할 때 사용
    String str1="1";
%>
<%
    String str2="1";
    str1=str1+"1";
    str2=str2+"1";
%>
<%=str1 %>
<br>
<%=str2 %>
</body>
</html>
```

스크립틀립에 대해서 이해해 보자.

// 표현식, 선언문, 스크립틀립은 한 파일에 여러개 선언할수 있다.  
// 스크립틀립은 자바코드를 쪼개서 사이에 html을 넣을 수 있다.

// 스크립틀립을 이용하여 5단을 출력하는 프로그램을 작성해 보자.

```
//04scriptlets.jsp //scrip
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    int dan=5;
%
```

```

        for(int i=1;i<10;i++){
%>
    <%= dan+"*"+ i + "="+ dan*i %> 출력<br>
<%
    }
%>
</body>
</html>

```

페이지 지시자에 대해서 알아보자.

//page 지시자는 현재 페이지 정보나 특정 클래스를 import할때 사용한다.

```

//import java.time.LocalDateTime;
//import java.time.format.DateTimeFormatter;

```

//05page.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.time.LocalDateTime" %>
<%@ page import="java.time.format.*" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
String formatDate = LocalDateTime.now()
    .format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
String formatDate2 = java.time.LocalDateTime.now()
    .format(java.time.format.DateTimeFormatter.ofPattern("yyyy-MM-dd"));
%>
<%=formatDate %><br>
<%=formatDate2 %>
</body>
</html>

```

<!--

문제1)

```

Import java.util.Arrays;
int[] iArr = {10, 20, 30};
System.out.println( Arrays.toString(iArr) );
다음 자바코드를 jsp로 구현하시오
-->

```

다음 코드는 문제에 대한 최종 결과물이다. 다음 코드에서 주석을 확인해 보자.

//06page.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.util.Arrays" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    int[] iArr={10,20,30};
%>
<%=Arrays.toString(iArr) %>
주석 사용 방법은 다음과 같다.
<%-- jsp 주석 --%>
<!-- html 주석 -->
// /* */ 자바 주석

</body>
</html>

<!--
문제1) 스크립틀릿 테이블에 1~25 봉어 출력 하는 프로그램을 스크립틀립으로 만들어 보자.
-->

```

## 내장 객체

JSP 내장 객체는 JSP 페이지에서 자동으로 생성되는 객체로, 개발자가 따로 객체를 생성하거나 초기화할 필요 없이 JSP 페이지에서 바로 사용할 수 있습니다. JSP 내장 객체는 다음과 같습니다.

**request:** HttpServletRequest 인스턴스를 참조하는 객체로, HTTP 요청과 관련된 정보를 제공합니다.

**response:** HttpServletResponse 인스턴스를 참조하는 객체로, HTTP 응답과 관련된 정보를 제공합니다.

**out:** JSP 페이지에서 출력을 담당하는 JspWriter 객체를 참조하는 객체로, HTML 코드를 생성하는 등의 출력 작업을 수행합니다.

**session:** HttpSession 인스턴스를 참조하는 객체로, 클라이언트와 서버 간 세션 정보를 관리합니다.

**application:** ServletContext 인스턴스를 참조하는 객체로, 웹 어플리케이션 전역에서

사용되는 정보를 저장하고 관리합니다.

`pageContext`: JSP 페이지와 관련된 정보를 담고 있는 `PageContext` 객체를 참조하는 객체로, 다른 내장 객체를 참조할 수 있는 메서드를 제공합니다.

`config`: `ServletConfig` 인스턴스를 참조하는 객체로, JSP 페이지와 관련된 서블릿 설정 정보를 제공합니다.

`exception`: JSP 페이지에서 발생한 예외 정보를 담고 있는 `Exception` 객체를 참조하는 객체로, 예외 처리에 사용됩니다.

Servlet에서 다음과 같이 사용하던 `out` 객체를

```
PrintWriter out = response.getWriter();
out.println("Hello, world!");
```

jsp에서는 내장 객체를 이용해서 객체 선언 없이 사용한다.

```
<%
    out.println("Hello, world!");
%>
```

나머지 내장 객체도 객체를 얻어오는 작업없이 jsp에서 선언 없이 바로 사용할 수 있다.

## > 04. GET, POST를 이용한 사용자 입력처리

GET과 POST는 웹 프로그래밍에서 사용되는 HTTP 요청 방식입니다. 이 두 방식은 서버에 데이터를 전송할 때 사용되며, 각각의 특징과 용도가 다릅니다.

### get방식 특징

1. 주소창에 사용자가 전송하는 데이터를 표시한다.
2. 255자 이하만 전송가능
3. 데이터 표시방법은 쿼리스트링을 사용한다.

쿼리스트링이란? 물음표 뒤에 key=value형태로 여러개의 데이터를 &로 연결해서 만든다.

ex) ?where=nexearch&sm=top\_hty&fbm=0&ie=utf8&query=강아지

### post방식 특징

1. 전송데이터가 주소창에 보이지 않는다.
2. id, pass 원드처럼 보안이 필요하거나 전송데이터가 대용량일때 사용한다.

### get방식 요청방법

1. 브라우저에 직접 기술하면 get방식
2. a태그 사용 get방식
3. html form태그에서 method 속성을 get으로 한다.

### post방식 요청방법

- 1.html form태그에서 method 속성을 post으로 한다.

다음 예제는 jsp에서 get방식 3개와 post방식 1개 총 4개를 jsp로 처리하는 방식이다.

jsp전송 get방식

1.form태그의 method get

data1:  
data2:  
data2:

전송

2번방법 a태그 사용

a태그 get방식 전송 3번방법 브라우저 사용 사용  
브라우저에 다음과 같이 전송

<http://localhost:8081/jsp04getpost/GetServlet.jsp?data1=a&data2=b&data2=c>

jsp전송 post

data1:  
data2:  
data2:

전송

data1값은 :  
a  
data2값은 :  
b  
c

왼쪽 이미지 는  
4개의 요청  
화면이고 오른쪽  
이미지는 결과  
화면이다.

입력화면에서의 GET 및 POST 요청을 사용한 내용을 요약하면 다음과 같다.

GET 방식 - 폼(form) 사용:

HTML <form> 요소를 사용하여 GET 방식으로 데이터를 서버로 전송합니다.

method="get"을 사용하여 GET 요청을 지정합니다.

사용자가 입력한 데이터는 URL의 쿼리 문자열에 노출됩니다.

GET 방식 - 하이퍼링크(a 태그) 사용:

<a> 태그(하이퍼링크)를 사용하여 GET 요청을 생성합니다.

링크의 href 속성에 데이터를 포함한 URL을 직접 지정합니다.

사용자가 링크를 클릭하면 데이터가 URL에 포함되어 전송됩니다.

GET 방식 - 브라우저 주소 표시줄 사용:

사용자는 브라우저 주소 표시줄을 직접 편집하여 GET 요청을 생성합니다.

데이터가 URL에 직접 포함되어 서버에 전송됩니다.

POST 방식 - 폼(form) 사용:

HTML <form> 요소를 사용하여 POST 방식으로 데이터를 서버로 전송합니다.

method="post"를 사용하여 POST 요청을 지정합니다.

사용자가 입력한 데이터는 HTTP 요청 본문에 포함되며 URL에 노출되지 않습니다.

이렇게 데이터를 전송하는 방식은 요청의 목적과 보안 요구 사항에 따라 다르게 사용됩니다.

GET은 데이터를 URL에 노출하므로 정보를 요청할 때 주로 사용되며, POST는 데이터를 숨기고 보안적으로 민감한 정보를 전송할 때 주로 사용됩니다.

사용자가 입력해서 전송한 데이터를 처리하려면 request 객체의 getParameter와 getParameterValues를 이용하면 된다.

**getParameter:**

getParameter 메서드는 지정된 이름의 파라미터에서 단일 값을 추출합니다.

만약 지정된 이름의 파라미터가 여러 개인 경우(예: 체크박스 그룹), 이 메서드는 첫 번째 값을 반환합니다.

반환 값은 문자열 형태이고 매개변수는 쿼리 스트링에서 받고자하는 key 부분을 문자열로 기술 하면된다.

예:

```
String username = request.getParameter("username"); // 단일 값 추출
```

**getParameterValues:**

getParameterValues 메서드는 지정된 이름의 파라미터에서 모든 값을 배열로 추출합니다.

여러 값을 가지는 파라미터에 유용하며, 반환 값은 문자열 배열로, 각 요소는 파라미터 값에 해당합니다.

여러개가 리턴 될 수 있으므로 반드시 문자열 배열로 받는다는 걸 잊지 말자.

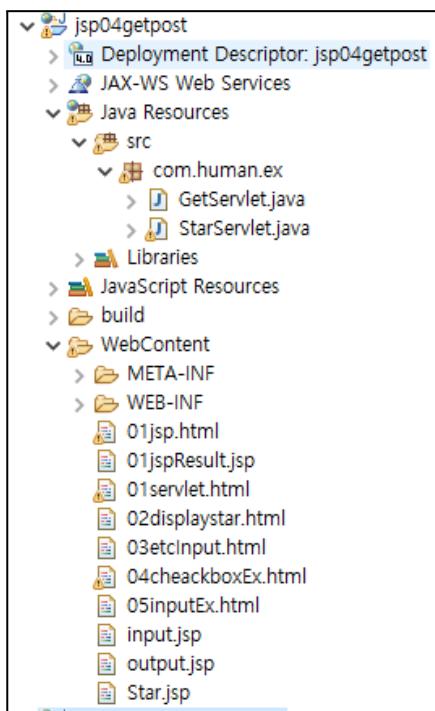
매개변수는 쿼리 스트링에서 받고자하는 key 부분을 문자열로 기술 하면된다.

예:

```
String[] selectedColors = request.getParameterValues("colors"); // 여러 값 추출
```

간단히 말해서, getParameter는 단일 값을 반환하고, getParameterValues는 모든 값을 배열로 반환합니다. 이러한 메서드를 적절히 사용하여 HTTP 요청에서 필요한 데이터를 추출할 수 있습니다.

//01.jsp.html 입력화면



```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    jsp전송 get방식<br>
    1.form태그의 method get<br>
    <form action="GetServlet.jsp" method="get">
        data1:<input type="text" name="data1"><br>
        data2:<input type="text" name="data2"><br>
        data2:<input type="text" name="data2"><br>
        <input type="submit" value="전송"><br>
    </form>

    2번방법 a태그 사용<br>
    <a href=

```

"<http://localhost:8081/jsp04getpost/GetServlet.jsp?data1=a&data2=b&data2=c>">  
a태그 get방식 전송 </a>

3번방법 브라우저 사용 사용<br>  
브라우저에 다음과 같이 전송<br>

<http://localhost:8081/jsp04getpost/GetServlet.jsp?data1=a&data2=b&data2=c<br><br>>

```

        jsp전송 post<br>
        <form action="GetServlet.jsp" method="post">
            data1:<input type="text" name="data1"><br>
            data2:<input type="text" name="data2"><br>
            data2:<input type="text" name="data2"><br>
            <input type="submit" value="전송"><br>
        </form>
    </body>
</html>

```

//01.jspResult.jsp 처리

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%

```

```

String data1=request.getParameter("data1");
String []data2=request.getParameterValues("data2");

%>
data1값은 :<br>
<%=data1 %> <br>
data2값은 :<br>
<%
if(data2!=null) {
    for(String str:data2) {
%>
    <%=str %> <br>
<%
}
}else {
%>
    값이 없음 <br>
<%           }      %>
</body>
</html>

```

다음 예제는 결과화면 처리를 jsp에서 하던것을 서블릿으로 한 것말고는 크게 차이가 없다.  
01servlet.html 파일과 GetServlet.java 서블릿 파일을 확인해 보자.

|  |   |
|--|---|
| <p>servlet get방식으로 데이터전송<br/>1번방법 form태그 사용</p> <p>data1: <input type="text"/></p> <p>data2: <input type="text"/></p> <p>data2: <input type="text"/></p> <p><input type="button" value="전송"/></p> <p>2번방법 a태그 사용<br/><a href="#">a태그 get방식 전송</a></p> <p>3번방법 브라우저 사용 사용</p> <p>브라우저에 다음과 같이 전송<br/><a href="http://localhost:8081/jsp04getpost/GetServlet?data1=a&amp;data2=b&amp;data2=c">http://localhost:8081/jsp04getpost/GetServlet?data1=a&amp;data2=b&amp;data2=c</a></p> <p>servlet post방식으로 데이터전송<br/>1번방법 form태그 사용</p> <p>data1: <input type="text"/></p> <p>data2: <input type="text"/></p> <p>data2: <input type="text"/></p> <p><input type="button" value="전송"/></p> | <p>data1값은 :<br/>a</p> <p>data2값은 :<br/>b<br/>c</p> |
|--|---|

다음 예제를 확인해서 이해해 보자.

```

//01servlet.html
<!DOCTYPE html>
<html>
<head>

```

```

<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <BR> 다음은 servlet으로 사용자 입력 데이터를 처리하는 방법이다.
    servlet get방식으로 데이터 전송<br>
    1번방법 form태그 사용<br>
    <form action="GetServlet" method="get">
        data1:<input type="text" name="data1"><br>
        data2:<input type="text" name="data2"><br>
        data2:<input type="text" name="data2"><br>
        <input type="submit" value="전송"><br>
    </form>
    2번방법 a태그 사용<br>
    <a href="http://localhost:8081/jsp04getpost/GetServlet?data1=a&data2=b&
data2=c">
        a태그 get방식 전송 </a>
    3번방법 브라우저 사용 사용<br>
    <br>
        브라우저에 다음과 같이 입력하고 전송 버튼을 누름<br>
        http://localhost:8081/jsp04getpost/GetServlet?data1=a&data2=b&data2=c<br>

```

```

servlet post방식으로 데이터전송<br>
1번방법 form태그 사용<br>
<form action="GetServlet" method="post">
    data1:<input type="text" name="data1"><br>
    data2:<input type="text" name="data2"><br>
    data2:<input type="text" name="data2"><br>
    <input type="submit" value="전송"><br>
</form>

</body>
</html>

```

```

//GetServlet.java
package com.human.ex;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

```

```

/**
 * Servlet implementation class GetServlet
 */
@WebServlet("/GetServlet")
public class GetServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public GetServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
     HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //요청에 대한 한글처리
        //사용자 입력 한글 깨짐 처리를 위한 인코딩
        request.setCharacterEncoding("UTF-8");

        String data1=request.getParameter("data1");
        String []data2=request.getParameterValues("data2");
        //브라우저 스트림에 문자열을 찍으면 브라우저에 나타나다.
        //응답에 대한 한글처리
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("data1값은 :<br>");
        out.println( data1+"<br>");
        out.println("data2값은 :");
        if(data2!=null) {
            for(String str:data2) {
                out.println( str+"<br>");
            }
        }else {
            out.println("값이 없음 <br>");
        }
        out.println("</body>");
        out.println("</html>");
    }
}

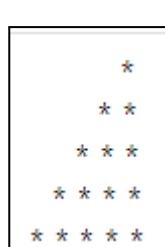
```

```

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    System.out.println("doPost 시작");
    doGet(request, response);
    System.out.println("doPost 종료");
}
}

```



```

<!--
다음 문제를 확인해 보자.
1. 숫자와 문자를 입력 받아 삼각형 형태에 모양으로 출력해보자.
    servlet get, post , jsp 3가지에서 요청에 대해서 동작 하도록
구현해 보자.
-->
//02displaystar.html

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action=StarServlet method=post >
    문자 입력 : <input type=text name=char1 ><br>
    숫자 입력 : <input type=text name=num1 ><br>
    <input type=submit value=post전송 /><br>
</form>
<br>
<form action=StarServlet method=get >
    문자 입력 : <input type=text name=char1 ><br>
    숫자 입력 : <input type=text name=num1 ><br>
    <input type=submit value=get전송 /><br>
</form>
<br>
<form action=Star.jsp method=get >
    문자 입력 : <input type=text name=char1 ><br>
    숫자 입력 : <input type=text name=num1 ><br>
    <input type=submit value=get전송 /><br>
</form>
<br>
</body>
</html>

```

```

//Star.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
String char1=request.getParameter("char1");
int num1=Integer.parseInt(request.getParameter("num1"));

for(int i=0;i<num1;i++) {
    //공간 찍기
    for(int j=i;j<num1;j++) {
%>
        &nbsp;
<%
}
//별찍는 for
for(int j=0;j<=i;j++) {
%>
    *
<%    }    %>
    <br>
<%    }    %>
</body>
</html>
```

```

//StarServlet.java
package com.human.ex;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class StarServlet
```

```

/*
@WebServlet("/StarServlet")
public class StarServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String char1=request.getParameter("char1");
        int num1=Integer.parseInt(request.getParameter("num1"));

        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<body>");
        for(int i=0;i<num1;i++) {
            //공간 찍기
            for(int j=i;j<num1;j++) {
                out.println(" ");
            }
            //별찍는 for
            for(int j=0;j<=i;j++) {
                out.println("*");
            }
            out.println("<br>");
        }
        out.println("</body>");
        out.println("</html>");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}

```

다음 3개의 html코드에서 사용자 입력을 받아 처리하는 서블릿을 만들어 보자.

```

//문제1) 03etcInput.html
<form action="FormEx1" method="get">
    이름 : <input type="text" name="name" size="10"><br>
    아이디 : <input type="text" name="id" size="10"><br>
    비밀번호 : <input type="password" name="pw" size="10"><br>
    취미 : <input type="checkbox" name="hobby" value="read">독서
          <input type="checkbox" name="hobby" value="cook">요리
          <input type="checkbox" name="hobby" value="run">조깅
          <input type="checkbox" name="hobby" value="swim">수영

```

```

<input type="checkbox" name="hobby" value="sleep">취침<br/>
과목 :<input type="radio" name="major" value="kor">국어
<input type="radio" name="major" value="eng" checked>영어
<input type="radio" name="major" value="mat">수학
<input type="radio" name="major" value="des">디자인<br/>
<input type="radio" name="major1" value="kor">국어
<input type="radio" name="major1" value="eng" checked>영어
<input type="radio" name="major1" value="mat">수학
<input type="radio" name="major1" value="des">디자인<br/>
<select name="protocol">
    <option value="http">http</option>
    <option value="ftp" selected = "selected">ftp</option>
    <option value="smtp">smtp</option>
    <option value="pop">pop</option>
</select>
<input type="submit" value="전송"><input type="reset"
value="초기화">
</form>

```

//문제2) 04checkboxEx.html

```

<form action="FormEx2" method="post">
    이름 : <input type="text" name="name"/><br>
    취미 : <input type="checkbox" name="hobby" value="cook">요리
        <input type="checkbox" name="hobby" value="book">독서
        <input type="checkbox" name="hobby" value="run">조깅
    <br>
    좋아하는 나무 :
    <input type="checkbox" name="trees" value="tree1">나무1
    <input type="checkbox" name="trees" value="tree2">나무2
    <input type="checkbox" name="trees" value="tree3">나무3
    <input type="submit" values="전송"/>
</form>

```

//문제3) 05inputEx.html

```

<!-- Label for속성에 기술된 id는 해당 라벨을 클릭하면 for에 기술된 id 컨트롤이 선택된다.
multiple 여러 파일을 선택할 수 있다.
textarea 여러줄 입력

```

전송하는 데이터가 없을 때 null 체크  
 데이터가 몇 개 올지 모를 때 배열 처리(getParameterValues)  
 null 체크와 문자열 배열 크기 확인해서 출력 가능(.Length 사용하여 크기 확인 가능)

```

-->
<form action="FormEx3" method="get">
<div>
    <label for="file">파일 여러개 선택</label>
    <input type="file" id="file" name="myfile" multiple /><br>

```

```

생일선택:<input type="datetime-local" name="birthday"><br>
여러줄 입력:
<textarea name="multiline" rows="4" cols="50">hi~textarea</textarea>
</div>
<div>
    <button>Submit</button>
</div>
</form>

```

답안

```

//FormEx1.java

String name=request.getParameter("name");
String id= request.getParameter("id");
String pw= request.getParameter("pw");
String[] hobby= request.getParameterValues("hobby");
String major = request.getParameter("major");
String major1 = request.getParameter("major1");
String protocol = request.getParameter("protocol");
String br = "<br>";
//브라우저 스트림에 문자열을 찍으면 브라우저에 나타난다.
response.setContentType("text/html; charset=UTF-8");
PrintWriter out=response.getWriter();
out.println("이름 : "+name+br);
out.println("아이디 : "+id+br);
out.println("비밀번호 : "+pw+br);
out.println("취미 : ");
for(String items : hobby) {
    out.println(items+" ");
}
out.println(br);
out.println("과목 : "+major+", "+major1+br);
out.println("프로토콜 : "+protocol+br);

}

//FormEx02.java
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
request.setCharacterEncoding("UTF-8");
String name =request.getParameter("name");
String[] hobbies= request.getParameterValues("hobby");
String[] trees = request.getParameterValues("trees");
String br = "<br>";
//브라우저 스트림에 문자열을 찍으면 브라우저에 나타난다.
response.setContentType("text/html; charset=UTF-8");

```

```

PrintWriter out=response.getWriter();
out.println("이름 : "+name);
out.println("취미 "+ br);
for(String hobby : hobbies) {
out.println(hobby+" ");
}
out.println(br);
out.println("좋아하는 나무 "+ br);
for(String tree : trees) {
out.println(tree+" ");
}
}

//FormEx03.java
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
String[] myfiles=request.getParameterValues("myfile");
String birthday= request.getParameter("birthday");
String multiline = request.getParameter("multiline");
String br = "<br>";
//브라우저 스트림에 문자열을 찍으면 브라우저에 나타난다.
response.setContentType("text/html; charset=UTF-8");
PrintWriter out=response.getWriter();
out.println("파일 이름 : " + br);
for(String items : myfiles) {
out.println(items);
}
out.println(br);
out.println("생년월일 : "+ birthday+br);
out.println("글 내용 "+ br);
out.println(multiline);
}

```

### 1. 물건 이름과 가격 4개값 입력

여자는 5% 추가할인

5%10%15%20% 쿠폰 중복사용가능

결과 페이지에 구입 물건의 정가, 할인가, 할인 내용을 출력 하시오.

다 한 사람은 카페에 올려주세요.

### 2. 회원가입 페이지를 만들어서 사용자가 입력한 정보를 서블릿,jsp로 출력해보자.

### 3. 사용자가 글자로 되어 있는 꽃, 강아지, 고양이, 건물 중 여러 개를 선택해서 선택된 글자에 해당하는 이미지를 화면에 보여주는 프로그램을 서블릿,jsp페이지로 만드시오.

### 4. input.jsp에서 취미 추가 자격증 추가 버튼을 눌러서 원하는 만큼 입력창을 추가한 다음 output.jsp에서 입력한 내용을 출력하는 프로그램을 구현해 보자.

<b>추미 추가</b>
• <input type="text" value="야구"/>
• <input type="text" value="배구"/>
• <input type="text" value="축구"/>
<b>자격증 추가</b>
• <input type="text" value="정보처리"/>
• <input type="text" value="sql전문가"/>
<b>전송</b>

- 추미**
- 야구
  - 배구
  - 축구
- 자격증**
- 정보처리
  - sql전문가

```
//input.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script>
    function addhb() {
        document.getElementById("div1").innerHTML += "<li><input
type='text' name='hobby'></li>";
    }
    function addsp() {
        document.getElementById("div2").innerHTML += "<li><input
type='text' name='spec'></li>";
    }
</script>
</head>
<body>
    <form action="output.jsp" method="post">
        <input type="button" value="추미 추가" onclick="addhb()">
        <div>
            <ul id="div1">
                <li><input type="text" name="hobby"></li>
            </ul>
        </div>
        <input type="button" value="자격증 추가" onclick="addsp()">
        <div>
            <ul id="div2">
                <li><input type="text" name="spec"></li>
            </ul>
        </div>
        <input type="submit" value="전송"/>
    </form>

```

```

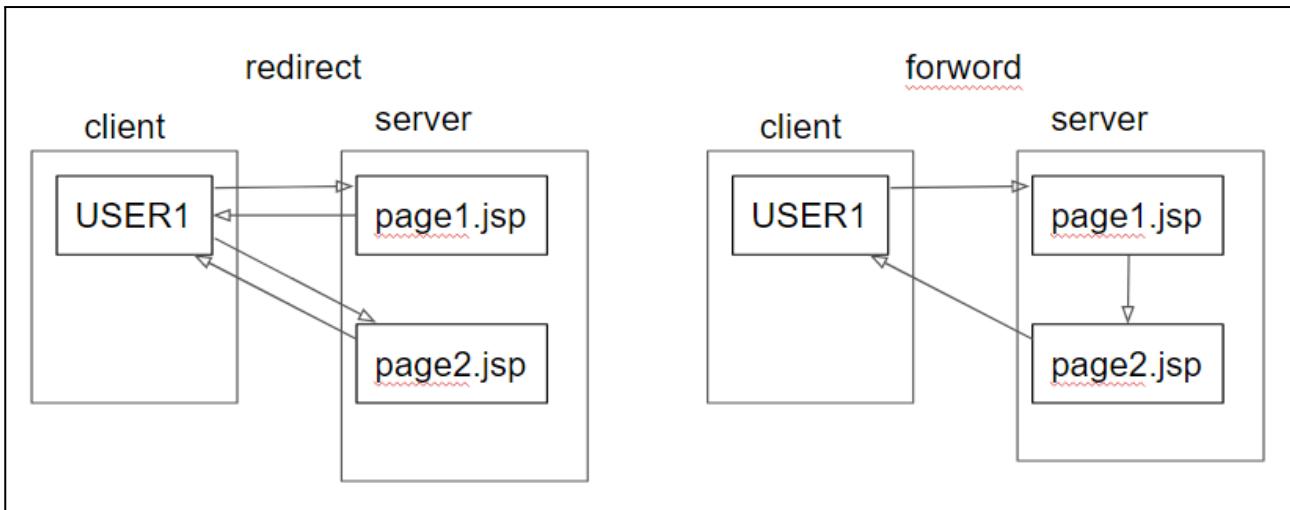
</body>
</html>

//output.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    //사용자 입력 한글 깨짐 처리를 위한 인코딩
    request.setCharacterEncoding("UTF-8");

    String [] hobbys = request.getParameterValues("hobby");
    String [] specs = request.getParameterValues("spec");
%>
취미
<ul>
<%
    for(int i=0; i<hobbys.length; i++){
%>
        <li><%=hobbys[i] %></li>
<%
    }
%>
</ul>
자격증
<ul>
<%
    for(int j=0; j<specs.length; j++){
%>
        <li><%=specs[j] %></li>
<%
    }
%>
</ul>
</body>
</html>

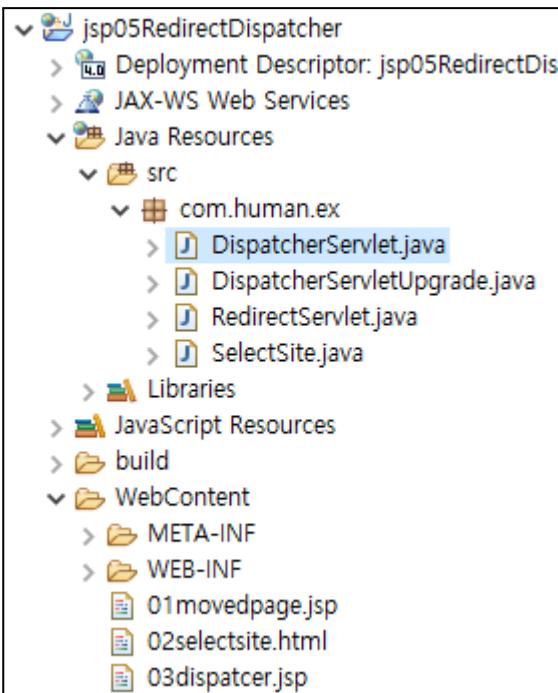
```

## > 05. Redirect 와 forward



redirect와 Dispatcher는 프로그램에서 다른 페이지로 이동할 때 사용한다.

redirect의 경우 요청을 마무리하고 다른 페이지로 이동하고 forward의 경우는 요청을 종료하지 않고 다른 페이지로 이동 한다.



```
//redirect
//1. 요청한 페이지를 종료하고 클라이언트에게
//새로운 페이지를 제공함
//2. 본인이 입력한 주소 대신에 주소창에 새로
//제공된 페이지 주소로 변경됨
//3. 새로 제공된 페이지에서 처음 요청한
//페이지의 request 객체에 접근할 수 없다.
//4. 다음과 같이 response 객체를 사용한다.
ex)
response.sendRedirect("01redirect.jsp");
```

```
//Dispatcher
//1. 요청한 페이지를 종료하지 않고
```

클라이언트에게 새로운 페이지를 제공함

```
//2. 요청한 브라우저 주소 입력창에 페이지가 변경되지 않고 처음 요청한 페이지가 유지됨
//3. 포워드 하여 이동된 페이지에서 처음 요청한 request 객체에 접근할 수 있다.
//4. 다음과 같이 response 객체를 사용한다.
ex)
RequestDispatcher dispatcher= request.getRequestDispatcher("03dispatcher.jsp");
dispatcher.forward(request,response);
```

**Redirect (리다이렉트):**

리다이렉트는 클라이언트의 요청을 다른 페이지로 보내는 방법입니다. 이때, 웹 애플리케이션은 새로운 페이지로 이동한 후 요청 객체가 새롭게 생성되며, 이전 페이지에서

설정한 데이터는 사라집니다. 다시 말해, 이동한 페이지에서는 이전 페이지의 요청 객체에 저장된 데이터를 읽을 수 없습니다. 리다이렉트는 주로 사용자를 다른 페이지로 보내는 데 사용되며, 이전 페이지와는 독립적인 요청이 발생합니다.

#### Forward (포워드):

포워드는 현재 페이지에서 다른 페이지로 요청을 전달하는 방법입니다. 이때, 요청 객체는 그대로 유지되며, 데이터가 유지됩니다. 따라서 이동한 페이지에서는 이전 페이지에서 요청 객체에 저장한 데이터를 읽을 수 있습니다. 포워드는 주로 동일한 요청에 대한 처리를 다른 페이지에서 계속하거나, 데이터를 공유해야 할 때 사용됩니다.

다음 예제는 리다이렉트 예제이다. /RedirectServlet 주소를 입력하면 해당 페이지에서 01redirect.jsp페이지로 이동한다.

```
//RedirectServlet.java
package com.human.ex;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/RedirectServlet")
public class RedirectServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public RedirectServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.sendRedirect("01movedpage.jsp");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}

//01movedpage.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
```

```

<title>Insert title here</title>
</head>
<body>
redirect나 forward로 이동된 페이지입니다.<br>
</body>
</html>

```

다음 예제는 DispatcherServlet주소를 요청하면 01movedpage.jsp페이지로 forward를 이용해서 이동하는 예제이다.

dispatcher객체 생성시 이동할 주소를 넣고 실제 이동하기 위한 forward메소드 매개변수에 request와 response 객체를 담아서 이동한다.

DispatcherServlet.java

```

@WebServlet("/DispatcherServlet")
public class DispatcherServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public DispatcherServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        RequestDispatcher dispatcher=
            request.getRequestDispatcher("01movedpage.jsp");
        dispatcher.forward(request,response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request, response);
    }
}

```

사용자 요청 객체에 데이터를 담아 이동한 페이지에서 처리하는 예제를 만들어 보자. 일단 redirect를 이용하면 요청한 객체가 종료된 상태에서 새로운 페이지로 이동 하므로 요청 객체를 통해 이동한 페이지에서 데이터를 받을 수 있는 방법은 없다. forward를 이용해서 가능한데 request객체에 원하는 데이터를 담는 방법은 setAttribute(key,value)이고, 읽어오는 방법은 getAttribute(key)이다.

다음 예제를 확인해 보면 /DispatcherServletUpgrade으로 요청 한 페이지의 request 객체에 데이터를 담은후 03dispatcer.jsp 페이지로 이동하여 이전에 담은 데이터를 읽어와서 보여 준다.

```

//DispatcherServletUpgrade.java
package com.human.ex;

import java.io.IOException;
import javax.servlet.RequestDispatcher;

```

```

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/DispatcherServletUpgrade")
public class DispatcherServletUpgrade extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public DispatcherServletUpgrade() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        /*
         * request 객체에 사용자가 원하는 데이터를 담을 수 있다.
         * setAttribute(key, value)를 이용하여 데이터를 담는다.
         * getAttribute(key)를 이용하여 데이터를 읽어온다.
         */
        request.setAttribute("name", "홍길동");

        RequestDispatcher dispatcher=
request.getRequestDispatcher("03dispatcer.jsp");
        dispatcher.forward(request,response);
        //forward 이후에 이부분의 코드는 의미가 없다.

    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}

//03dispatcer.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
dispatcher된 페이지입니다.<br>

```

```
setAttribute한 name값은 <%=String)request.getAttribute("name") %>입니다.  
</body>  
</html>  
  
//forward나 sendRedirect 다음에 기술한 코드는 접근할 수 없는 코드 블럭이 되므로  
forward나 sendRedirect 다음에는 어떤 코드도 기술하지 말아야 한다.
```

다음 문제를 redirect로 만들어 보자.

문제1)

사용자에게 naver, google, 다음 중 하나를 선택 하게 해서 선택된 페이지로 이동하는 서블릿을 만들어 보자.

입력페이지는 라디오 박스로 만들자.

redirect를 이용해서 사용자가 선택한 페이지로 이동하게 만들어 보자.

문제2) 나이를 입력 받아 나이별로 적절한 페이지로 이동하는 프로그램을 구해 보자.

문제3) 일반고객인지, vip고객인지 판별하여 출력해 보자.

답안 1)

```
//02selectsite.html  
<form action="SelectSite" method="get">  
    <input type="radio" name="site" value="daum"/> 다음  
    <input type="radio" name="site" value="naver"/> 네이버  
    <input type="radio" name="site" value="google"/> 구글  
    <input type="submit" value="전송">  
</form>
```

```
//SelectSite.java  
...  
@WebServlet("/SelectSite")  
...  
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    String site=request.getParameter("site");  
  
    String siteMove="";  
    if(site.equals("daum")) {  
        siteMove="https://www.daum.net";  
    }else if(site.equals("naver")) {  
        siteMove="https://www.naver.com";  
    }else if(site.equals("google")) {  
        siteMove="https://www.google.com";  
    }  
    response.sendRedirect(siteMove);  
}
```

다음 문제를 forward하여 jsp로 출력 하시오.

1.

숫자 2개 연산자 1개를 입력 받아서 연산결과를 화면에 출력하는 프로그램을 구현해 보자.

2. 체크 박스에서 꽃이름을 선택하여 선택된 이미지들을 화면에 출력해 보자.

3. 다음과 같은 로그인 프로그램을 구현하시오.

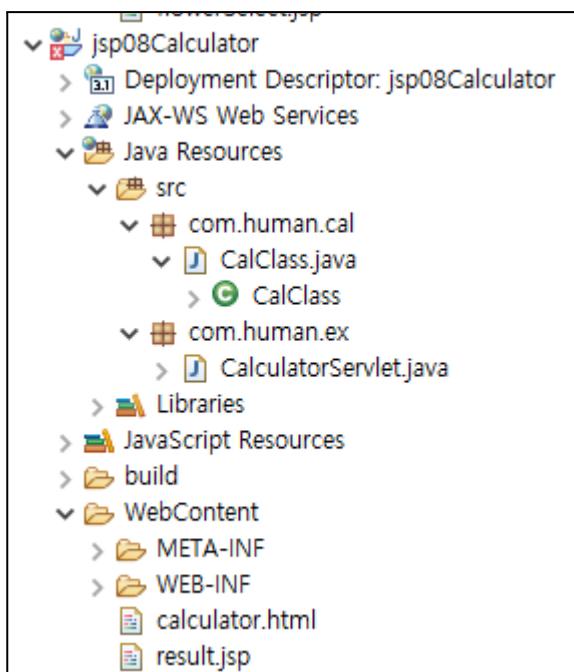
사용자의 아이디와 비밀번호를 login.html에서 입력받아(java1,1234)(java2,2345) LoginServlet 서블릿에서 틀리면 입력페이지(login.html)로 redirect하고 맞으면

success.jsp로 forward하여 성공 메시지와 사용자 이름을 출력하시오.

프로젝트 이름은 jsp06login으로 한다.

답안

1.



```
//CalClass.java
package com.human.cal;

public class CalClass {
    //멤버변수
    private double num1,num2;
    private String operator="+";
    private String result="";

    public String resultString() {
        if(operator.equals("+")) {
            result=num1+"+"+num2+"=(num1+num2)";
        }else if(operator.equals("-"))
            result=num1+"-"+num2+"=(num1-num2);
    }else if(operator.equals("*")) {
        result=num1+"*"+num2+"=(num1*num2);
    }else if(operator.equals("/")) {
        result=num1+"/"+num2+"=(num1/num2);
    }
    return result;
}

@Override
public String toString() {
    return resultString();
}

public double getNum1() {
    return num1;
}
```

```

public void setNum1(double num1) {
    this.num1 = num1;
}
public double getNum2() {
    return num2;
}
public void setNum2(double num2) {
    this.num2 = num2;
}
public String getOperator() {
    return operator;
}
public void setOperator(String operator) {
    this.operator = operator;
}
}

//CalculatorEx.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.human.cal.CalClass;
@WebServlet("/CalculatorEx")
public class CalculatorServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public CalculatorServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        myCalculator(request, response);
    }
    public void myCalculator(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        double num1 = Double.parseDouble(request.getParameter("num1"));
        double num2 = Double.parseDouble(request.getParameter("num2"));
        String operator = request.getParameter("operation");
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");

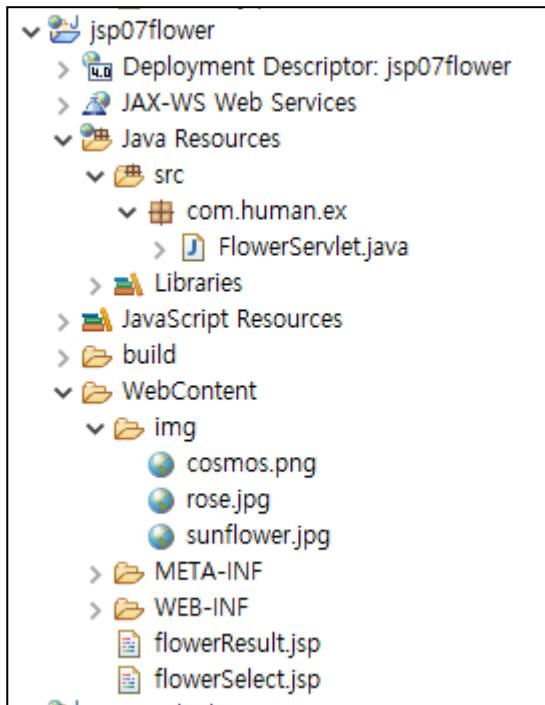
```

```

        out.println("num1,num2 계산 결과<br>");
        /*if(operator.equals("+")) {
            out.println(num1+"+"+num2+"="+ (num1+num2));
        }else if(operator.equals("-")) {
            out.println(num1+"-"++num2+"="+ (num1-num2));
        }else if(operator.equals("*")) {
            out.println(num1+"*"+num2+"="+ (num1*num2));
        }else if(operator.equals("/")) {
            out.println(num1+"/"++num2+"="+ (num1/num2));
        }*/
        CalClass calClass=new CalClass();
        calClass.setNum1(num1);
        calClass.setNum2(num2);
        calClass.setOperator(operator);
        System.out.println(calClass);
        out.println(calClass);
        //주소가 리턴 toString제정의하면 제정의한 문자열이 리턴
        out.println(calClass.resultString());
        //10+20=30
        out.println("</body>");
        out.println("</html>");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}

```

## 2.



```

//flowerSelect.jsp
<%@ page language="java"
contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
출력을 원하는 꽃을 선택하세요.
<form action="FlowerServlet" method="get">
<input type="checkbox" name="flower"
value="cosmos.png">코스모스
<input type="checkbox" name="flower"
value="rose.jpg">장미
<input type="checkbox" name="flower"

```

```

value="sunflower.jpg">해바라기
    <input type="submit" value="확인">
</form>
</body>
</html>

//FlowerServlet.java
package com.human.ex;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/FlowerServlet")
public class FlowerServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public FlowerServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        String[] flowers = request.getParameterValues("flower");
        if (flowers != null) {

            request.setAttribute("flowers", flowers);

            RequestDispatcher dispatcher =
request.getRequestDispatcher("flowerResult.jsp");
            dispatcher.forward(request, response);
        } else {
            response.sendRedirect("main.html");
        }
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}

```

```

}

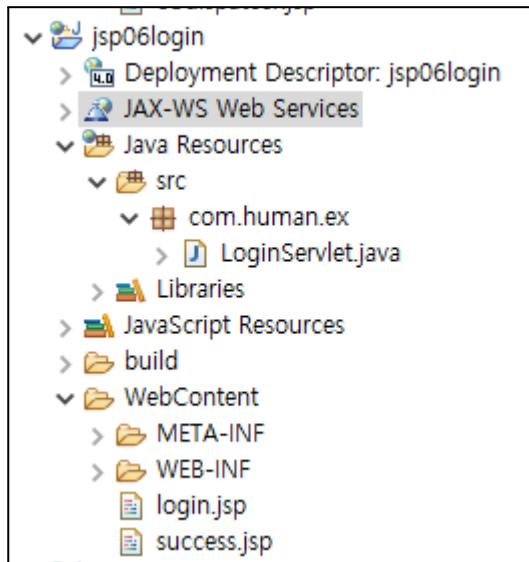
//flowerResult.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    String []flowers = (String [])request.getAttribute("flowers");

    for(int i=0; i<flowers.length;i++){
%>
        <%=flowers[i] %><br>
        <br>
<%
    }
%>
</body>
</html>

```

3.

//login.jsp



```

<%@ page language="java"
contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script>
    if(<%=request.getParameter("isSuccess") %>==false){
        alert('아이디를 잘못입력하였습니다.');
    }

```

```

    if(<%=request.getParameter("isLogin") %>==false){
        alert('로그인이 해야 사용할수 있습니다.');
    }

```

```

        }
    </script>
</head>
<body>
로그인<br>
<form action="LoginServlet" method="post">
    id : <input type="text" name="id"><br>
    password: <input type="password" name="pw"><br>
    <input type="submit" value="전송">
</form>
</body>
</html>

//LoginServlet.java
package com.human.ex;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

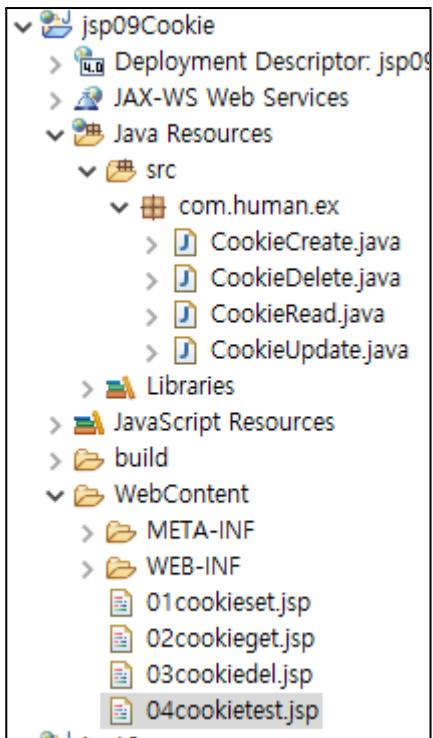
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public LoginServlet() { super(); }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        String id = request.getParameter("id");
        String pw = request.getParameter("pw");
        if ((id.trim().equals("java1") && pw.trim().equals("1234")) ||
            (id.trim().equals("java2") && pw.trim().equals("2345"))) {
            request.setAttribute("id", id);
            // 로그인 성공
RequestDispatcher dispatcher = request.getRequestDispatcher("success.jsp");
            dispatcher.forward(request, response);
        } else {
            // 로그인 실패
            response.sendRedirect("login.jsp?isSuccess=false");
        }
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

```

```
        doGet(request, response);
    }
}

//success.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    로그인 성공 페이지
    <br>
    <%
        if(request.getAttribute("id")==null){
            //response.sendRedirect("/jsp06Login/Login.jsp?isLogin=false");
out.println("<script>location.href='/jsp06login/login.jsp?isLogin=false'</scr
ipt>");
        }
    %>
    <%=request.getAttribute("id") %>님이 로그인 하였습니다.
</body>
</html>
```

## > 06. cookie 사용 방법



쿠키는 사용자의 정보를 클라이언트에 저장하는 방법이다. 클라이언트에서 사용 하지만 클라이언트에서 직접 조작하지 않고 클라이언트의 모든 쿠키를 서버에 보내서 서버에서 조작한 다음 클라이언트에 정보를 보내서 클라이언트에 저장 한다.

쿠키 사용방법은 다음과 같다.

1. 생성

```
Cookie cookie=new Cookie("name","psm");
cookie.setMaxAge(600); //쿠키 사용시간 설정 600초  
동안 쿠키 유지
```

2. 쿠키 수정 : 키 입력 부분에 기존에 들어 있는 같은 키 값을 사용하면 데이터가 수정된다.

```
Cookie cookie=new Cookie("name","updatePsm");
```

3. 쿠키 삭제

```
cookie=new Cookie("name","deletePsm");
cookie.setMaxAge(0); //삭제됨      // 0 삭제 // -1
```

문한대

4. 쿠키 읽어오기 : 모든 쿠키를 읽어와 반복문을 이용해서 확인할 수 있다.

```
Cookie cookies[] = request.getCookies();
```

5. 서버에서 작업한 쿠키 정보를 response 객체를 이용해서 클라이언트에 저장 한다.

```
response.addCookie(cookie);
```

```
//CookieCreate.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
@WebServlet("/CookieCreate")
public class CookieCreate extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public CookieCreate() {
        super();
    }
```

```

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //쿠키 생성 작업순서
        //1. 클라이언트에서 생성된 쿠키를 가지고 서버로 이동한다.
        //2. 쿠키 생성 작업 페이지에서 쿠키를 만든다.
        //3. response 객체에 담아서 클라이언트에 저장한다.

        // 쿠키 생성
        Cookie cookie=new Cookie("name","psm");
        cookie.setMaxAge(600); //쿠키 사용시간 설정 600초 동안 쿠키 유지
        // 0 삭제 // -1 문한대
        response.addCookie(cookie);

        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println(cookie.getName()+"<br>");
        out.println(cookie.getValue()+"<br>");
        out.println(cookie.getMaxAge()+"<br>");
        out.println("</body>");
        out.println("</html>");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request, response);
    }
}

//CookieRead.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/CookieRead")
public class CookieRead extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public CookieRead() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse

```

```

response) throws ServletException, IOException {
        Cookie cookies[] = request.getCookies();
        // ...
        // ...
        // ...
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        for (Cookie cookie : cookies) {
            out.println(cookie.getName() + "<br>");
            out.println(cookie.getValue() + "<br>");
            out.println(cookie.getMaxAge() + "<br>");
        }
        out.println("</body>");
        out.println("</html>");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}

//CookieUpdate.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/CookieUpdate")
public class CookieUpdate extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public CookieUpdateDelete() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // 쿠키 수정 : 같은 키값에 새로운 이름을 넣는다.
        Cookie cookie = new Cookie("name", "updatePsm");
        cookie.setMaxAge(600); // 쿠키 사용시간 설정 600초 동안 쿠키 유지
        response.addCookie(cookie);
    }
}

```

```

        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println(cookie.getName()+"<br>");
        out.println(cookie.getValue()+"<br>");
        out.println(cookie.getMaxAge()+"<br>");
        out.println("</body>");
        out.println("</html>");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}

//CookieDelete.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/CookieDelete")
public class CookieDelete extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public CookieUpdateDelete() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        //쿠키 삭제
        Cookie cookie=new Cookie("name","deletePsm");
        cookie.setMaxAge(0); //삭제됨
        response.addCookie(cookie);
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println(cookie.getName()+"<br>");
        out.println(cookie.getValue()+"<br>");
        out.println(cookie.getMaxAge()+"<br>");
        out.println("</body>");
    }
}

```

```

        out.println("</html>");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request, response);
    }
}

//01cookieset.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    쿠키 생성됨
    <%
        Cookie cookie=new Cookie("cookieN","cookieV");
        cookie.setMaxAge(60*60); //1시간
        response.addCookie(cookie);
    %>
    <a href="02cookieget.jsp">cookie get</a>
</body>
</html>

//02cookieget.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    Cookie[] cookies=request.getCookies();
    for(int i=0;i<cookies.length;i++){
        String str=cookies[i].getName();
        if(str.equals("cookieN")){
            out.println("쿠키 name :" +cookies[i].getName()+"<br>");
            out.println("쿠키 value :" +cookies[i].getValue()+"<br>");
        }
    }
%>
</body>
</html>

```

```

        out.println("-----"+<br>);
    }
}

%>
<a href="03cookiedel.jsp">cookie delete</a>
</body>
</html>

//03cookiedel.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
Cookie[] cookies=request.getCookies();
for(int i=0;i<cookies.length;i++){
    String str=cookies[i].getName();
    if(str.equals("cookieN")){
        out.println("name : "+cookies[i].getName()+"<br>");
        cookies[i].setMaxAge(0);
        response.addCookie(cookies[i]);
    }
}
%>
<a href="04cookietest.jsp">쿠키확인</a>
</body>
</html>

//04cookietest.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
Cookie[] cookies=request.getCookies();

```

```

if(cookies!=null){
    for(int i=0;i<cookies.length;i++){
        out.println(cookies[i].getName()+"<br/>");
        out.println(cookies[i].getValue()+"<br/>");
    }
}
%>
</body>
</html>

```

클라이언트에서 생성한 쿠키를 작업하는 방법은 다음과 같다. f12 >> application >> cookies 를 선택한 다음 현재 작업중인 주소를 선택하면 해당 주소에서 사용하는 쿠키를 확인 할 수 있다.

The screenshot shows the Google Chrome DevTools interface with the Application tab selected. On the left, the Storage section is expanded, showing Local storage, Session storage, IndexedDB, Web SQL, and Cookies. Under Cookies, there is an entry for the domain <http://localhost:8081>. The main panel displays a table of cookies:

Name	Value	D...	P...	Ex...	Si...	H...
Idea-b5bbac...	fa4ad4c6-9052-4e64-9...	lo...	/	2...	49	✓
cookieN	cookieV	lo...	/j...	2...	14	
JSESSIONID	6A0AC2162D9BCF5027...	lo...	/j...	S...	42	✓

A message at the bottom right says "Select a cookie to preview its value".

## > 07. 쿠키관리 프로그램 구현

아래 이미지는 main.html

**쿠키 생성**

name :   
value :

**쿠키 조회**

**쿠키 업데이트**

쿠키 이름 :   
변경할 value :

**쿠키 삭제**

삭제할 쿠키 이름 :

아래 이미지는 create.jsp

**coockie setting page**

name : park  
value : 123  
age : 600

[메인으로 이동](#)

아래 이미지는 update.jsp

name : park  
변경 전 value : 123  
변경 된 value : 1234  
age : 100

[메인으로 이동](#)

다음 이미지는 select.jsp

**쿠키 불러오기**

cookies[0] name : JSESSIONID  
cookies[0] value : 7CEBC2E87110632318F0DD6E7F989087  
cookies[0] age : -1

cookies[1] name : park  
cookies[1] value : 123  
cookies[1] age : -1

cookies[2] name : Idea-b5bbacf3  
cookies[2] value : fa4ad4c6-9052-4e64-91e6-f8689b7b4855  
cookies[2] age : -1

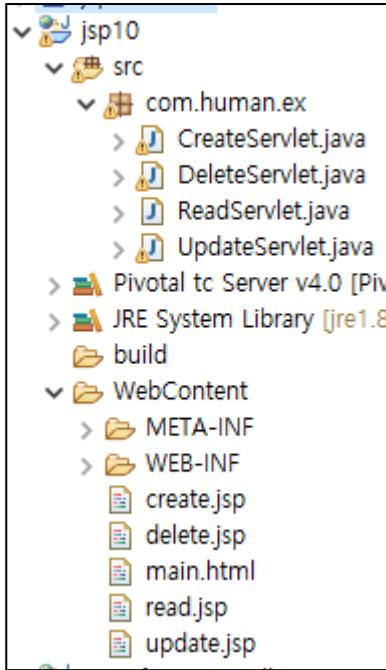
[메인으로 이동](#)

다음 이미지는 delete.jsp

삭제된 name : park  
삭제된 value : 1234

[메인으로 이동](#)

다음 웹프로그램은 main페이지에서 원하는 쿠키 작업을 선택해서 사용자 입력 및 전송 버튼을 눌러서 각 페이지에 쿠키 작업 결과를 보여주는 예제이다.



메인 화면 main.html 파일을 실행시키고

메인 화면에 사용자 입력을 하고 쿠키 생성 버튼을 누르면 CreateServlet.java를 통해서 쿠키가 생성되고 create.jsp로 결과 화면을 보여 준다.

쿠키조회 버튼을 누르면 ReadServlet.java를 통해서 쿠키를 조회하여 read.jsp로 결과 화면을 보여 준다.

쿠키업데이트 버튼을 누르면 UpdateServlet.java를 통해서 쿠키를 조회하여 update.jsp로 결과 화면을 보여준다.

쿠키삭제 버튼을 누르면 DeleteServlet.java를 통해서 쿠키를 조회하여 delete.jsp로 결과 화면을 보여준다.

```
//main.html
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>쿠키 생성</h1>
    <form action="CreateServlet">
        name : <input type="text" name="name"> <br>
        value : <input type="text" name="value"> <br>
        <input type="submit" value="쿠키생성">
    </form>
    <hr>
    <h1>쿠키 조회</h1>
    <form action="ReadServlet">
        <input type="submit" value="쿠키조회">
    </form>
    <hr>
    <h1>쿠키 업데이트</h1>
    <form action="UpdateServlet">
        쿠키 이름 : <input type="text" name="name"><br>
        변경할 value : <input type="text" name="updateValue"><br>
        <input type="submit" value="변경하기">
    </form>
    <hr>
    <h1>쿠키 삭제</h1>
    <form action="DeleteServlet">
        삭제할 쿠키 이름 : <input type="text" name="delName"><br>
```

```

        <input type="submit" value="삭제하기">
    </form>
    <br>
</body>
</html>

//CreateServlet.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/CreateServlet")
public class CreateServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public CreateServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String name = request.getParameter("name");
        String value = request.getParameter("value");
        Cookie cookie=new Cookie(name,value);
        cookie.setMaxAge(600);
        response.addCookie(cookie);
        request.setAttribute("name", cookie.getName());
        request.setAttribute("value", cookie.getValue());
        request.setAttribute("age", cookie.getMaxAge());
        RequestDispatcher dispatcher = request.getRequestDispatcher("create.jsp");
        dispatcher.forward(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        doGet(request, response);
    }
}

//DeleteServlet.java
package com.human.ex;
import java.io.IOException;

```

```

import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/DeleteServlet")
public class DeleteServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public DeleteServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Cookie cookies[] = request.getCookies();
        String delName = request.getParameter("delName");
        boolean check = false;
        String name = "";
        String value = "";
        for (int i = 0; i < cookies.length; i++) {
            if (delName.equals(cookies[i].getName())) {
                check = true;
                name = cookies[i].getName();
                value = cookies[i].getValue();
                cookies[i].setMaxAge(0); // '삭제' 맵
                response.addCookie(cookies[i]);
                request.setAttribute("name", name);
                request.setAttribute("value", value);
            }
        }
        request.setAttribute("check", check);
        RequestDispatcher dispatcher = request.getRequestDispatcher("delete.jsp");
        dispatcher.forward(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}

//ReadServlet.java
package com.human.ex;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;

```

```

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/ReadServlet")
public class ReadServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public ReadServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Cookie cookies[] = request.getCookies();
        request.setAttribute("cookies", cookies);
        RequestDispatcher dispatcher = request.getRequestDispatcher("read.jsp");
        dispatcher.forward(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}

//UpdateServlet.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/UpdateServlet")
public class UpdateServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public UpdateServlet() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Cookie cookies[] = request.getCookies();
        String name = request.getParameter("name");
        String updateValue = request.getParameter("updateValue");
        String value = "";

```

```

        for (int i = 0; i < cookies.length; i++) {
            if(name.equals(cookies[i].getName())) {
                value = cookies[i].getValue();
                Cookie cookie = new Cookie(name, updateValue);
                cookie.setMaxAge(600);
                response.addCookie(cookie);
                request.setAttribute("name", name);
                request.setAttribute("value", value);
                request.setAttribute("updateValue", updateValue);
                request.setAttribute("age", 100);
            }
        }
    RequestDispatcher dispatcher = request.getRequestDispatcher("update.jsp");
    dispatcher.forward(request, response);
}
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}

//create.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>coockie setting page</h1>
    <%    request.setCharacterEncoding("utf-8");    %>
    name : <%= request.getAttribute("name") %><br>
    value : <%= request.getAttribute("value") %> <br>
    age : <%= request.getAttribute("age") %> <br>
    <br>
    <a href="/jsp10/main.html">메인으로 이동</a>
</body>
</html>

//delete.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
```

```

<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>coockie setting page</h1>
    <%    request.setCharacterEncoding("utf-8");    %>
    name : <%= request.getAttribute("name") %><br>
    value : <%= request.getAttribute("value") %> <br>
    age : <%= request.getAttribute("age") %> <br>
    <br>
    <a href="/jsp10/main.html">메인으로 이동</a>
</body>
</html>

//read.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>쿠키 불러오기</h1>
    <%
        Cookie[] cookies = (Cookie[])request.getAttribute("cookies");

        for (int i = 0; i < cookies.length; i++) {
            String name = cookies[i].getName();
            String value = cookies[i].getValue();
            int age = cookies[i].getMaxAge();

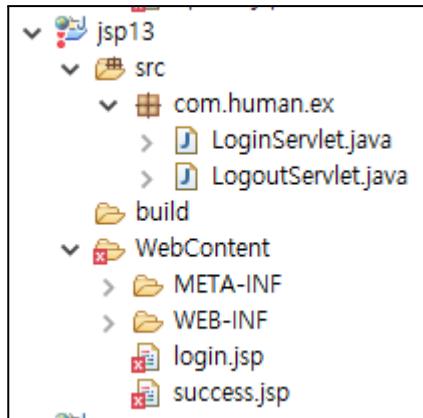
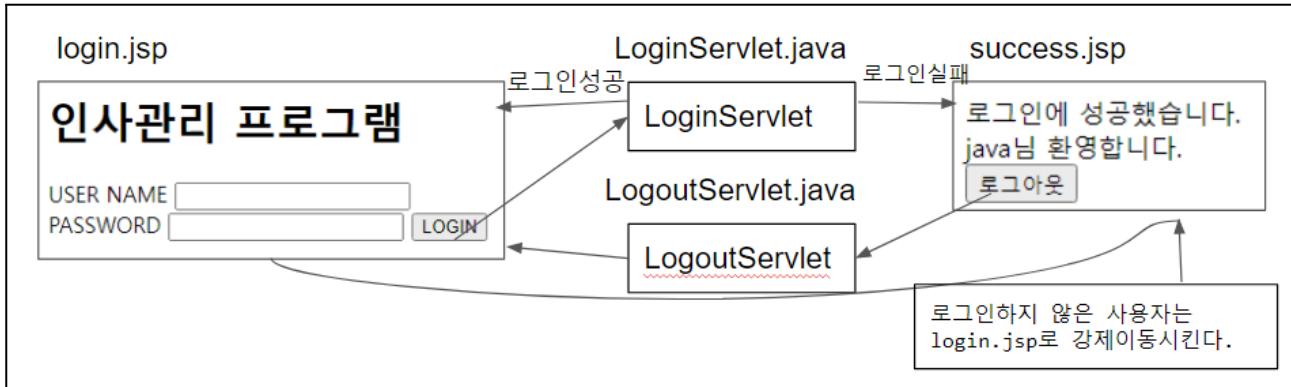
            out.println("cookies[" + i + "] name : " + name + "<br>");
            out.println("cookies[" + i + "] value : " + value + "<br>");
            out.println("cookies[" + i + "] age : " + age + "<br>");
            out.println("<br/>");
        }
    %><br>
    <a href="/jsp10/main.html">메인으로 이동</a>
</body>
</html>

//update.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>

```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    request.setCharacterEncoding("utf-8");
    if(request.getAttribute("name") == null){
        out.println("<h1>없는 ID 입니다.</h1>");
    }else{
%
    name :<%= request.getAttribute("name") %><br>
    변경 전 value :<%= request.getAttribute("value") %><br>
    변경 된 value :<%= request.getAttribute("updateValue") %><br>
    age : <%= request.getAttribute("age") %><br>
<%
    }
%
<br>
<a href="/jsp10/main.html">메인으로 이동</a>
</body>
</html>
```

## > 08. 쿠키를 사용한 로그인



<!--

쿠키를 이용해서 로그인 만들기

1. **login.jsp**에서 사용자 입력 아이디와 패스워드를 받아서 **login**버튼을 눌러서 **LoginServlet**으로 이동한다.

2. **LoginServlet**에서는 아이디와 패스워드 일치 여부를 확인해서 일치하면 **success.jsp** 일치하지 않으면 다시 로그인 할 수 있도록 **login.jsp**파일로 이동하게 한다.

3. **success.jsp**는 로그인 한 사람만 들어가게 하고, 해당 화면에 **로그아웃** 버튼이 있어 **로그아웃**을 클릭하면 **LogoutServlet**으로 이동하여 **로그아웃**되고 다시 로그인 할 수 있도록 **login.jsp**로 이동하여 로그인 화면을 보여 준다.

4. **success.jsp**에 로그인 하지 않은 사용자가 들어 오면 **login.jsp**로 이동해서 로그인을 유도한다.

5. **success.jsp**에 로그인 한 사람만 들어올 수 있게 하는 방법은 쿠키에 **id**정보가 있는 사용자만 **success.jsp**페이지에 들어갈 수 있도록 하면된다.

6. 쿠키에 **id**를 등록하는 방법은 **LoginServlet**에서 로그인에 성공하면 쿠키에 **id**를 등록하고 **success.jsp** 화면으로 이동한다.

6. 로그아웃 하는 방법은 **LogoutServlet**으로 사용자가 들어 오면 **id**관련 쿠키를 삭제하면 된다.

로그인의 의미는 쿠키에 **id**값등록

로그아웃의 의미는 쿠키에 **id**값 삭제.

쿠키를 이용한 로그인 구현 방법은 다음과 같습니다.

login.jsp 파일에서 사용자가 입력한 아이디와 패스워드를 받아서 LoginServlet으로 전송합니다.

LoginServlet에서는 입력받은 아이디와 패스워드를 데이터베이스와 비교하여 일치하는지 확인합니다.

만약 일치하는 경우, success.jsp 파일로 이동하도록 리다이렉트합니다. 이때, 사용자의 아이디와 패스워드를 쿠키로 저장합니다.

success.jsp 파일에서는 사용자가 로그인 한 경우에만 접속할 수 있는 페이지를 보여줍니다. 또한, 로그아웃 버튼을 클릭하면 LogoutServlet으로 이동하여 쿠키를 삭제하고 로그아웃 처리를 합니다.

LogoutServlet에서는 쿠키를 삭제하고 login.jsp 파일로 이동하도록 리다이렉트합니다. 이렇게 쿠키를 이용하여 로그인 기능을 구현하면 사용자가 다시 로그인할 필요 없이 일정 기간동안 로그인 상태를 유지할 수 있습니다. 하지만 보안적인 측면에서 쿠키를 사용한 로그인은 취약점이 존재할 수 있으므로, 보안에 대한 추가적인 조치가 필요합니다. 예를 들어, 쿠키에 저장되는 사용자의 정보를 암호화하거나 쿠키의 유효 기간을 짧게 설정하는 등의 방법이 있습니다.

```
-->
//login.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="jquery-3.4.1.js"></script>
    <script>
        if(<%=request.getParameter("isSuccess")%>==false){
            alert('USER NAME 혹은 PASSWORD가 잘못 입력되었습니다.');
        }else if(<%=request.getParameter("isLogout")%>==true){
            alert('로그아웃되었습니다. 로그인페이지로 이동합니다.');
        }else if(<%=request.getParameter("isLogin")%>==false){
            alert('로그인해야 이용할 수 있는 페이지입니다. 로그인페이지로 이동합니다.');
        }
    </script>
</head>
<body>
    <h1>인사관리 프로그램</h1>
    <form action="LoginServlet" method="get">
        <label for="user_id">USER NAME</label>
        <input type="text" name="user_id" id="user_id" autocomplete="off"
required>
        <label for="user_pw">PASSWORD</label>
        <input type="password" name="user_pw" id="user_pw" autocomplete="off"
required>
        <button type="submit" id="btn">LOGIN</button>
    </form>
</body>
```

```

</html>

//LoginServlet.java
package com.human.ex;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public LoginServlet() {
        super();
        // TODO Auto-generated constructor stub
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String user_id= request.getParameter("user_id");
        String user_pw= request.getParameter("user_pw");
        if(user_id.trim().equals("java")&&user_pw.trim().equals("1234"))
{
            Cookie cookie = new Cookie("user_id",user_id);
            cookie.setMaxAge(600);
            response.addCookie(cookie);

            RequestDispatcher dispatcher =
request.getRequestDispatcher("success.jsp");
            dispatcher.forward(request, response);
        } else {
            response.sendRedirect("login.jsp?isSuccess=false");
        }
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}

```

```

//success.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
Cookie[] cookies=request.getCookies();

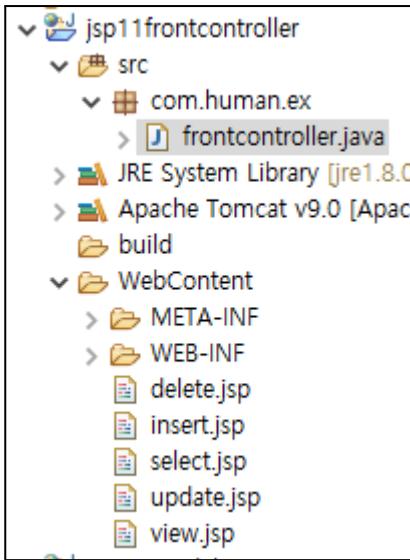
String user_id="";
for(int i=0;i<cookies.length;i++){
    String str=cookies[i].getName();
    if(str.equals("user_id")){
        user_id=cookies[i].getValue();
    }
}
if(user_id.equals("")){
    response.sendRedirect("login.jsp?isLogin=false");
}else{
    out.println("로그인에 성공했습니다.<br>"+user_id+"님 환영합니다.");
}
%>
<form action="LogoutServlet" method="get">
<button type = "submit">로그아웃</button>
</form>
</body>
</html>

//LogoutServlet.java
package com.human.ex;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/LogoutServlet")
public class LogoutServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public LogoutServlet() {
        super();
    }
}

```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String user_id=(String) request.getAttribute("user_id");
    Cookie cookie = new Cookie("user_id", user_id);
    cookie.setMaxAge(0);
    response.addCookie(cookie);
    response.sendRedirect("login.jsp?isLogout=true");
}
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    doGet(request, response);
}
}
```

## > 08. frontcontroller



매핑 주소는 보통 1:1 매칭이지만 \*를 이용해서 여러 매핑 주소를 한번에 받아서 처리 할 수 있다. \*.do 라고 매핑 주소를 만들면 .do로 끝나는 여러 개의 주소를 한번에 받을 수 있다.

이 방식을 이용해서 여러 요청을 받아서 처리하는 컨트롤러를 만들 수 있다.

이 코드는 간단한 웹 애플리케이션을 구현한 코드입니다. `frontcontroller.java` 파일에서 모든 \*.do 주소를 받아와서 해당하는 view 파일을 렌더링합니다. `view.jsp`, `insert.jsp`, `select.jsp`, `delete.jsp`, `update.jsp`는 각각 메뉴를 표시하며 링크를 클릭하면 해당하는 \*.do 주소를 호출합니다. 이 코드에서는 jsp를 이용하여 구현했으며, MVC(Model-View-Controller) 패턴을 적용하고 있습니다.

.코드를 살펴보면 .do를 모두 받을 수 있도록 `frontcontroller.java` 서블릿에 `@WebServlet("*.do")`로 기술 하였다.

`view.jsp`를 실행하면 `insert.do`, `update.do`, `select.do`, `delete.do` 매핑을 처리할 수 있는 메뉴가 나온다. 본인 컴퓨터의 8081포트에서 실행하면 결국에 모두 서버상의 `frontcontroller.java`로 가게 설정되어 있다. 실행 후 클릭한 후 브라우저상에 기록된 주소를 확인해 보자.

```
<a href="insert.do">insert</a>
<a href="/jsp11frontcontroller/select.do">select</a>
<a href="http://localhost:8081/jsp11frontcontroller/delete.do">delete</a>
<a href="http://localhost:8081
<%=request.getContextPath()%>/update.do">update</a>
```

`frontcontroller.java` 에서는 .do 로 끝나는 모든 주소를 받아서 처리해야 하기 때문에 사용자가 요청한 마지막 주소 `insert.do`, `update.do`, `select.do`, `delete.do`를 찾아서 해당 부분에 맡는 로직을 실행하고(콘솔에 어떤 주소로 들어 왔는지 출력됨) 해당하는 `view`화면 .jsp파일을 정해서 해당 페이지로 forward하도록 구현되어 있음.

```
//view.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
```

```

<body>
view메뉴<br>
<a href="insert.do">insert</a>
<a href="/jsp11frontcontroller/select.do">select</a>
<a href="http://localhost:8081/jsp11frontcontroller/delete.do">delete</a>
<a href="http://localhost:8081
<%=request.getContextPath()%>/update.do">update</a>
</body>
</html>

//select.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
select메뉴<br>
<a href="insert.do">insert</a>
<a href="/jsp11frontcontroller/select.do">select</a>
<a href="http://localhost:8081/jsp11frontcontroller/delete.do">delete</a>
<a href="http://localhost:8081
<%=request.getContextPath()%>/update.do">update</a>
<a href="view.do">view</a>
</body>
</html>

//insert.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
insert메뉴<br>
<a href="insert.do">insert</a>
<a href="/jsp11frontcontroller/select.do">select</a>
<a href="http://localhost:8081/jsp11frontcontroller/delete.do">delete</a>
<a href="http://localhost:8081
<%=request.getContextPath()%>/update.do">update</a>
<a href="view.do">view</a>

```

```

</body>
</html>

//delete.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
Delete메뉴<br>
<a href="insert.do">insert</a>
<a href="/jsp11frontcontroller/select.do">select</a>
<a href="http://localhost:8081/jsp11frontcontroller/delete.do">delete</a>
<a href="http://localhost:8081
<%=request.getContextPath()%>/update.do">update</a>
<a href="view.do">view</a>
</body>
</html>

//update.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
update메뉴<br>
<a href="insert.do">insert</a>
<a href="/jsp11frontcontroller/select.do">select</a>
<a href="http://localhost:8081/jsp11frontcontroller/delete.do">delete</a>
<a href="http://localhost:8081
<%=request.getContextPath()%>/update.do">update</a>
<a href="view.do">view</a>
</body>
</html>

```

```

//frontcontroller.java
package com.human.ex;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * 매핑주소를 *.do 로 하면 .do로 끝나는 모든 주소를 받아올 수 있다.
 * 매핑주소 맨앞에 "/"은 없어야 한다.
 */
@WebServlet("*.do")
public class frontcontroller extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public frontcontroller() {
        super();
    }
    protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
        // URI:/jsp10/hello.do
        // conPath:/jsp10
        // command:/hello.do
        String uri = request.getRequestURI();
        System.out.println("URI:" + uri);
        // 원하는 주소에 대한 처리 방법
        String conPath = request.getContextPath();
        System.out.println("conPath:" + conPath);

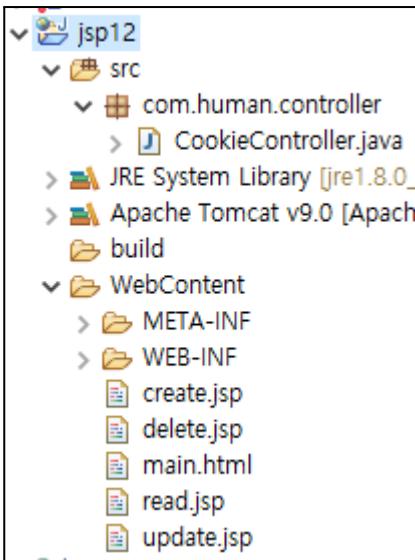
        String command = uri.substring(conPath.length());
        System.out.println("command:" + command);

        String viewPage = "view.jsp";
        if (command.equals("/insert.do")) {
            viewPage = "insert.jsp";
            System.out.println("insert.do");
        } else if (command.equals("/select.do")) {
            viewPage = "select.jsp";
            System.out.println("select작업");
            System.out.println("insert.do");
        } else if (command.equals("/delete.do")) {
            viewPage = "delete.jsp";
            System.out.println("delete작업");
            System.out.println("insert.do");
        } else if (command.equals("/update.do")) {
            viewPage = "update.jsp";
        }
    }
}

```

```
        System.out.println("update작업");
        System.out.println("insert.do");
    }
    System.out.println(viewPage);
    RequestDispatcher dispatcher =
request.getRequestDispatcher(viewPage);
    dispatcher.forward(request, response);
    //jsp10번 예제를 frontcontroller를 이용해서 jsp12으로 만들어 보자.
}
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}
```

## > 09. cookie 관리 프로그램 to frontcontroller



이전에 만든 쿠키 관리 프로그램을 frontcontroller형태로 만들어 보자.

main.html이 view.jsp에 해당하는 시작 메뉴이고 해당 메뉴에서 쿠키의 create, read, update, delete 작업을 할 수 있는 사용자 입력 화면이 있고 입력후 버튼을 눌러서 create.do, read.do, update.do, delete.do로 이동 하여 각각의 쿠키 작업을 한후 각각의 결과물 화면 .jsp로 이동 한다.

```
//main.html
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
```

```
<title>Insert title here</title>
</head>
<body>
<h1>쿠키 생성</h1>
<form action="Create.Servlet">
    name : <input type="text" name="name"> <br>
    value : <input type="text" name="value"> <br>
    <input type="submit" value="쿠키생성">
</form>
<hr>
<h1>쿠키 조회</h1>
<form action="Read.Servlet">
    <input type="submit" value="쿠키조회">
</form>
<hr>
<h1>쿠키 업데이트</h1>
<form action="Update.Servlet">
    쿠키 이름 : <input type="text" name="name"><br>
    변경할 value : <input type="text" name="updateValue"><br>
    <input type="submit" value="변경하기">
</form>
<hr>
<h1>쿠키 삭제</h1>
<form action="Delete.Servlet">
    삭제할 쿠키 이름 : <input type="text" name="delName"><br>
    <input type="submit" value="삭제하기">
</form>
<br>
```

```

</body>
</html>

//create.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>coockie setting page</h1>
<%
request.setCharacterEncoding("utf-8");
%>
name : <%= request.getAttribute("name") %><br>
value : <%= request.getAttribute("value") %> <br>
age : <%= request.getAttribute("age") %> <br>
<br>
<a href="/jsp12/main.html">메인으로 이동</a>
</body>
</html>

//read.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>쿠키 불러오기</h1>
<%
Cookie[] cookies = (Cookie[])request.getAttribute("cookies");

for (int i = 0; i < cookies.length; i++) {
    String name = cookies[i].getName();
    String value = cookies[i].getValue();
    int age = cookies[i].getMaxAge();

    out.println("cookies[" + i + "] name : " + name + "<br>");
    out.println("cokies[" + i + "] value : " + value + "<br>");
```

```

        out.println("cokies[" + i + "] age : " + age + "<br>");
        out.println("<br/>");
    }
%><br>
<a href="/jsp12/main.html">메인으로 이동</a>
</body>
</html>

//update.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    request.setCharacterEncoding("utf-8");
    if(request.getAttribute("name") == null){
        out.println("<h1>없는 ID 입니다.</h1>");
    }else{
%
        name :<%= request.getAttribute("name") %><br>
        변경 전 value :<%= request.getAttribute("value") %><br>
        변경 된 value :<%= request.getAttribute("updateValue") %><br>
        age : <%= request.getAttribute("age") %><br>
<%
    }
%
<br>
<a href="/jsp12/main.html">메인으로 이동</a>
</body>
</html>

//delete.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%

```

```

request.setCharacterEncoding("utf-8");
boolean check = (boolean)request.getAttribute("check");
if(check == false){
    out.println("<h1>삭제에 실패하였습니다.(없는 ID입니다.)</h1>");
}else{
%>
삭제된 name :<%= request.getAttribute("name") %><br>
삭제된 value :<%= request.getAttribute("value") %><br>
<%
}
%>
<br>
<a href="/jsp12/main.html">메인으로 이동</a>
</body>
</html>

```

```

//CookieController.java
package com.human.controller;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("*.Servlet")
public class CookieController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public CookieController() {
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // URI:/jsp10/hello.do
        // conPath:/jsp10
        // command:/hello.do
        String uri = request.getRequestURI();
        System.out.println("URI:" + uri);
        // 원하는 주소에 대한 처리 방법
        String conPath = request.getContextPath();
        System.out.println("conPath:" + conPath);

        String command = uri.substring(conPath.length());
        System.out.println("command:" + command);
    }
}

```

```

String viewPage="main.html";
if(command.equals("/Create.Servlet")) {
    viewPage="create.jsp";
    String name = request.getParameter("name");
    String value = request.getParameter("value");
    Cookie cookie=new Cookie(name,value);
    cookie.setMaxAge(600); //쿠키 사용시간 설정 600초 동안 쿠키 유지
    // 0 삭제 // -1 문한대
    response.addCookie(cookie);
    request.setAttribute("name", cookie.getName());
    request.setAttribute("value", cookie.getValue());
    request.setAttribute("age", cookie.getMaxAge());
} else if(command.equals("/Read.Servlet")) {
    Cookie cookies[] = request.getCookies();
    request.setAttribute("cookies", cookies);

    viewPage="read.jsp";
} else if(command.equals("/Update.Servlet")) {
    Cookie cookies[] = request.getCookies();

    String name = request.getParameter("name");
    String updateValue = request.getParameter("updateValue");
    String value = ""; //기존값

    for (int i = 0; i < cookies.length; i++) {
        if(name.equals(cookies[i].getName())) {
            value = cookies[i].getValue(); //기존값

            // 쿠키 수정
            Cookie cookie = new Cookie(name, updateValue);
            cookie.setMaxAge(600); // 쿠키 사용시간 설정 600초 동안 쿠키 유지
            response.addCookie(cookie);

            request.setAttribute("name", name);
            request.setAttribute("value", value);
            request.setAttribute("updateValue", updateValue);
            request.setAttribute("age", 100);
        }
    }
    viewPage="update.jsp";
} else if(command.equals("/Delete.Servlet")) {
    viewPage="delete.jsp";
    Cookie cookies[] = request.getCookies();
    String delName = request.getParameter("delName");
    boolean check = false;
}

```

```

        String name = "";
        String value = "";
        for (int i = 0; i < cookies.length; i++) {
            if (delName.equals(cookies[i].getName())) {
                check = true;
                name = cookies[i].getName();
                value = cookies[i].getValue();
                cookies[i].setMaxAge(0); // 삭제됨
                response.addCookie(cookies[i]);

                request.setAttribute("name", name);
                request.setAttribute("value", value);
            }
        }
        request.setAttribute("check", check);
    }
    RequestDispatcher dispatcher = request.getRequestDispatcher(viewPage);
    dispatcher.forward(request, response);
}
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}

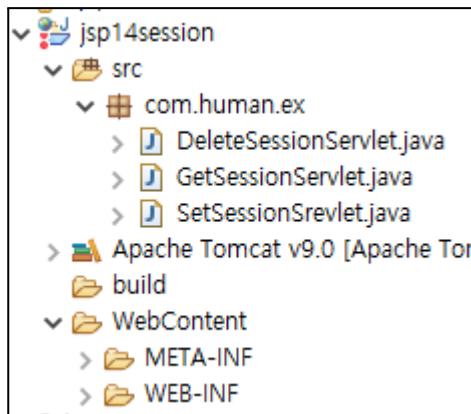
```

## > 10. session

세션은 클라이언트가 원하는 개인정보를 서버에 저장할 때 사용한다.

서버는 클라이언트 A와 클라이언트 B를 구별할 수 없습니다. 요청이 들어오면 누가 보냈는지 알 수 없다. 그런데 세션을 사용하기 원하면 사용자에게 세션 ID를 발급하고, 이를 쿠키로 클라이언트로 저장 한다. 그 후 사용자가 세션을 사용하려면 세션 ID를 이용하여 이 사용자가 이전에 세션을 사용한 사용자인지를 판단하여 만약 동일한 사용자라면 이전 세션을 계속 사용하고, 다른 사용자라면 새로운 세션 ID를 생성하여 클라이언트에게 제공합니다.

세션(Session)은 웹 서버와 클라이언트(브라우저) 간의 상호 작용에서 발생하는 정보를 일시적으로 서버에 저장하는 방법 중 하나입니다. 클라이언트가 서버에 접속하면 서버는 그 클라이언트에 대한 고유한 세션 ID를 생성하고, 이 ID를 쿠키(cookie)나 URL 파라미터(parameter) 등의 방법으로 클라이언트에 전달합니다. 이후 클라이언트가 서버에 요청을 보낼 때마다 서버는 해당 세션 ID를 이용하여 세션에 저장된 정보를 참조하거나 업데이트 합니다. 이를 통해 사용자 인증 정보, 장바구니 정보 등을 유지할 수 있습니다. 세션은 일정 시간 동안 유지되며, 일반적으로 사용자가 로그아웃하거나 브라우저를 종료할 때 종료됩니다.



세션은 세션 객체에 있는 메소드로 작업을 한다. 다음과 같은 세션 객체의 메소드가 있다.  
`setAttribute()` 메서드는 HttpSession 객체에 데이터를 저장하는 역할을 하며,  
`getAttribute()` 메서드로 세션에 저장한 값을 읽어올수 있다.  
`getMaxInactiveInterval()` 메서드는 세션이 유효한 시간을 반환하고, `setMaxInactiveInterval()` 메서드는 세션의 유효시간을 설정하는 역할을 합니다.

### 세션 저장 방법

`SetSessionServlet.java` 서블릿에서는 세션을 어떻게 저장하는지 보여준다.

```
HttpSession session=request.getSession();
session.setAttribute("name", "park");
session.setMaxInactiveInterval(600);
```

이 코드는 JSP에서 HttpSession 객체를 생성하고, "name"이라는 이름으로 "park"라는 값을 세션에 저장하며, 이 세션의 유효시간을 600초로 설정하는 코드입니다.

HttpSession 객체는 클라이언트와 서버 간의 통신에서 세션을 유지하기 위한 객체이며, 이를 사용하여 클라이언트가 웹 애플리케이션과 상호 작용하는 동안 필요한 데이터를 저장하고 공유할 수 있습니다.

세션을 수정하는 방법은 저장하는 방법과 동일한데 같은 키로 새로운 값을 넣으면 된다.

```
session.setAttribute("name", "newValue");
```

세션 읽어오는 방법은 다음과 같다.

```

//GetSessionServlet.java 코드는 저장한 세션을 불러오는 코드이다.
HttpSession session=request.getSession();
String name=(String)session.getAttribute("name");

//DeleteSessionServlet.java 코드는 세션을 삭제하는 코드이다.
HttpSession session=request.getSession();
session.removeAttribute("name");//부분 삭제//name만 삭제
session.invalidate();//전체삭제//사용자에 모든 세션을 삭제

//SetSessionServlet.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
@WebServlet("/setSessionServlet")
public class SetSessionServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public SetSessionServlet() { super(); }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        HttpSession session=request.getSession();
        session.setAttribute("name", "park");
        session.setMaxInactiveInterval(600);

        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out=response.getWriter();
        out.println("<html>"); out.println("<body>");
        out.println("<h1>세션 저장 완료:"+session.getMaxInactiveInterval()+"</h1>");
        out.println("</body>"); out.println("</html>");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}

//GetSessionServlet.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;

```

```

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
@WebServlet("/getSessionServlet")
public class GetSessionServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public GetSessionServlet() { super(); }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpSession session=request.getSession();
        String name=(String)session.getAttribute("name");

        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out=response.getWriter();
        out.println("<html>"); out.println("<body>");
        out.println("<h1>세션에서 name값 읽어오기:"+name+"</h1>");
        out.println("</body>"); out.println("</html>");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request, response);
    }
}
//DeleteSessionServlet.java
package com.human.ex;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
@WebServlet("/DeleteSessionServlet")
public class DeleteSessionServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public DeleteSessionServlet() { super(); }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpSession session=request.getSession();
        session.removeAttribute("name");//name만 삭제 //부분 삭제
        session.invalidate();//사용자에 모든 세션을 삭제//모두 삭제

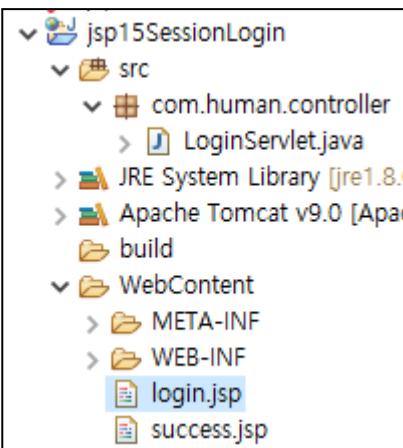
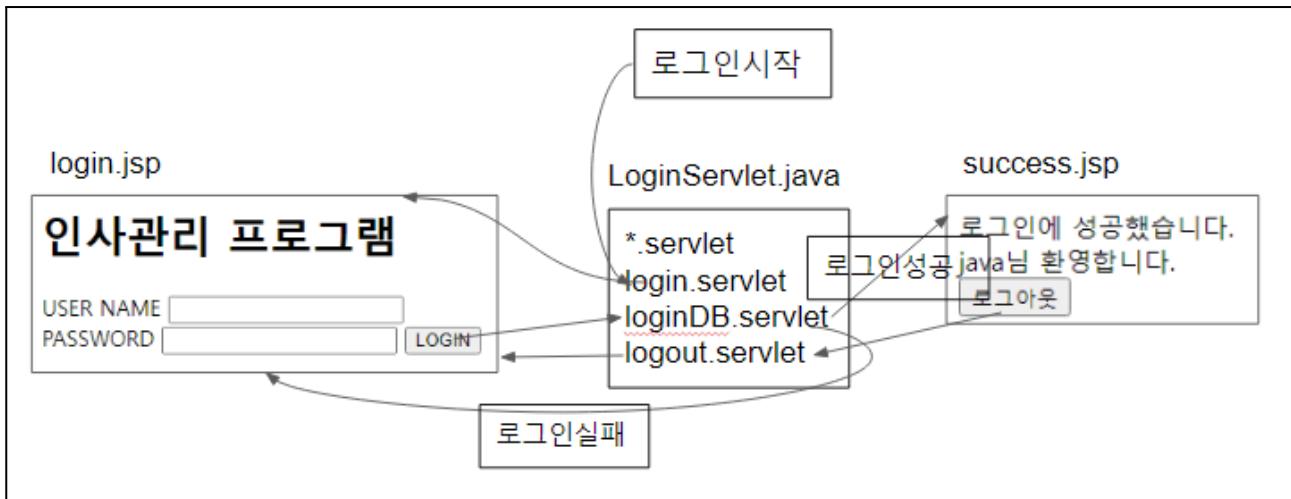
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out=response.getWriter();
        out.println("<html>"); out.println("<body>");

```

```
        out.println("<h1>세션 삭제됨</h1>");
        out.println("</body>"); out.println("</html>");
        //session과 frontcontroller를 이용하여 로그인을 만들어 보자.
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}
```

jsp를 이용한 세션은 서블릿이랑 동일해서 생략 했다. 차이점이 있다면 세션은 내장 객체여서 HttpSession session=request.getSession(); 과 같이 선언하여 생성할 필요 없이 바로 session 인스턴스로 접근하여 사용하면 된다.

## > 11. sessionLogin to frontcontroller



로그인 프로그램에서 로그인의 의미는 다음과 같이 세션을 등록하는 것이고  
session.setAttribute("user\_id", user\_id);  
로그아웃의 의미는 다음과 같이 로그인과 관련된 모든 세션을 지우는 것이다. session.invalidate();

프로그램을 시작하려면 login.servlet를 주소창에 입력하면 된다. login.jsp가 나타나면 id password를 입력하고 login버튼을 누르면 loginDB.servlet으로 이동하고 해당 서블릿에서 로그인에 성공하면 success.jsp로 이동하고 실패하면 login.jsp로 이동 시킨다. success.jsp화면에 로그인 없이 들어오면

login.servlet으로 이동시키고 로그아웃 버튼을 누르면 logout.servlet으로 이동하여 로그 아웃시킨 다음 login.jsp페이지로 이동시킨다.

다음 전체코드를 확인해서 로그인 프로젝트를 만들어 보고 이해해 보자.

```
//LoginServlet.java
package com.human.controller;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
@WebServlet("*.servlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

```

public LoginServlet() {
    super();
}
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // URI:/jsp10/hello.do
    // conPath:/jsp10
    // command:/hello.do
    String uri = request.getRequestURI();
    System.out.println("URI:" + uri);
    String conPath = request.getContextPath();
    System.out.println("conPath:" + conPath);
    String command = uri.substring(conPath.length());
    System.out.println("command:" + command);
    String viewPage = "login.jsp";
    if (command.equals("/login.servlet")) {
        viewPage = "login.jsp";
    }else if (command.equals("/loginDB.servlet")) {
        System.out.println("login.servletDB");
        viewPage = "success.jsp";
        String user_id= request.getParameter("user_id");
        String user_pw= request.getParameter("user_pw");
    if(user_id.trim().equals("java")&&user_pw.trim().equals("1234")) {
        HttpSession session=request.getSession();
        session.setAttribute("user_id",user_id);
        session.setMaxInactiveInterval(600);
    } else {
        viewPage="login.jsp";
    }
//response.sendRedirect("Login.jsp?isSuccess=false");
    }
} else if (command.equals("/logout.servlet")) {
    viewPage = "login.jsp";
    HttpSession session=request.getSession();
    session.invalidate();
    System.out.println("logout.servlet");
}
System.out.println(viewPage);
RequestDispatcher dispatcher = request.getRequestDispatcher(viewPage);
dispatcher.forward(request, response);
}
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    doGet(request, response);
}
}

```

```

//login.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="jquery-3.4.1.js"></script>
    <script>
        if(<%=request.getParameter("isSuccess")%>==false){
            alert('USER NAME 혹은 PASSWORD가 잘못 입력되었습니다.');
        }else if(<%=request.getParameter("isLogout")%>==true){
            alert('로그아웃되었습니다. 로그인페이지로 이동합니다.');
        }else if(<%=request.getParameter("isLogin")%>==false){
            alert('로그인해야 이용할 수 있는 페이지입니다. 로그인페이지로 이동합니다.');
        }
    </script>
</head>
<body>
    <h1>인사관리 프로그램</h1>
    <form action="loginDB.servlet" method="get">
        <label for="user_id">USER NAME</label>
        <input type="text" name="user_id" id="user_id" autocomplete="off"
required>
        <label for="user_pw">PASSWORD</label>
        <input type="password" name="user_pw" id="user_pw" autocomplete="off"
required>
        <button type="submit" id="btn">LOGIN</button>
    </form>
</body>
</html>

//success.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
</head>
<body>
<%
if(session.getAttribute("user_id")==null){

```

```
    response.sendRedirect("login.servlet?isLogin=false");
}else{
    String user_id=(String)session.getAttribute("user_id");
    out.println("로그인에 성공했습니다.<br>"+user_id+"님 환영합니다.");
}
%>
<form action="logout.servlet" method="get">
    <button type = "submit">로그아웃</button>
</form>
</body>
</html>
```

## > 12. scope

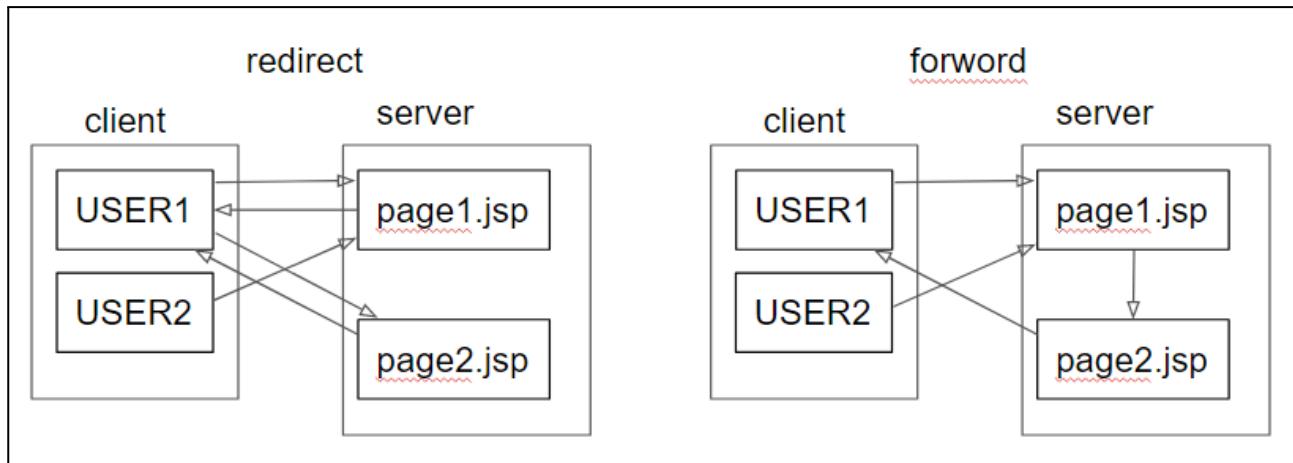
java 변수는 전역 변수와 지역변수로 되어 있는데, jsp같은 경우 서버 클라이언트와 같은 복잡한 구조여서 jsp에서는 특정 범위에서 데이터를 저장할 수 있는 4개의 객체를 제공한다. 사용자가 사용할 범위에 맞춰서 적절히 사용하면 된다.

페이지 스코프: JSP 페이지 내에서만 유효한 스코프입니다. 변수를 선언하면 해당 페이지에서만 사용할 수 있습니다.

요청 스코프: HTTP 요청 단위로 유효한 스코프입니다. 클라이언트가 서버에 요청을 보낼 때 생성되고, 응답을 받을 때까지 유지됩니다.

세션 스코프: 웹 애플리케이션 사용자 세션 단위로 유효한 스코프입니다. 사용자가 로그인한 후 로그아웃할 때까지 유지됩니다.

애플리케이션 스코프: 웹 애플리케이션 전체에 걸쳐 유효한 스코프입니다. 웹 애플리케이션이 시작되고 종료될 때까지 유지됩니다.



page는 해당 page에서 만 접근 가능 하다. 현재 요청한 page1.jsp에서 사용하는 변수를 현재 요청한 page1.jsp에서만 사용할 수 있다.

request인 경우 요청이 끝날때 까지 접근 가능하다. 현재 요청한 page1.jsp에서 사용하는 변수를 redirect인 경우 page2.jsp에서 사용할 수 없고 forward 인경우 page2.jsp에서 사용할수 있다.

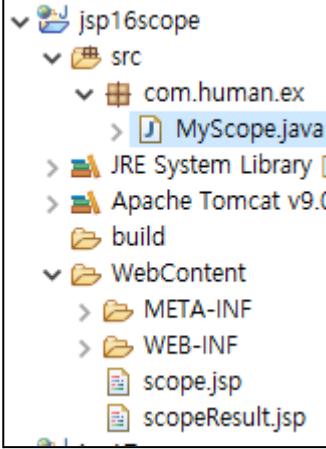
session의 경우 page1.jsp이든 page2.jsp이든 관계 없이 같은 사용자라면 언제나 모든 페이지에서 접근 가능하다.

application인 경우는 누구나 모든 페이지에서 접근 가능하다.

```
//scope종류
//1. page 해당 페이지에서만 접근가능    pageContext
pageContext.setAttribute("page1", "hello page");
//2. request 요청이 종료할때 까지 값을 유지 request
request.setAttribute("request1", "hello request");
//3. session 같은 사용자일때만 값을 유지 session
session.setAttribute("session1","hello session");
//4. application 모든 사용자가 값을 유지 application
application.setAttribute("application1", "hello application");
```

읽어올때는 `setAttribute`(읽어오고 싶음 키값) 메소드를 사용한다.

한 컴퓨터에서 2명의 유저를 테스트 하고 싶다면 브라우저를 열어서 서버에 접근하면 브라우저를 처리할 수 있다.



```
//scope.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
scope 영역별 데이터 등록<br>
<%
//scope종류
//1. page 해당 페이지에서만 접근가능    pageContext
pageContext.setAttribute("page1", "hello page");
//2. request 요청이 종료할때 까지 값을 유지 request
request.setAttribute("request1", "hello request");
//3. session 같은 사용자일때만 값을 유지 session
session.setAttribute("session1","hello session");
//4. application 모든 사용자가 값을 유지 application
application.setAttribute("application1", "hello application");
//읽어오기 같은 페이지에서 저장후 찍기때문에 모두 찍힌다.
out.println((String)pageContext.getAttribute("page1"));
out.println((String)request.getAttribute("request1"));
out.println((String)session.getAttribute("session1"));
out.println((String)application.getAttribute("application1"));
//리다이렉트 리다이렉트 되면
//response.sendRedirect("scopeResult.jsp");
//forward
/* RequestDispatcher
dispatcher=request.getRequestDispatcher("scopeResult.jsp");
    dispatcher.forward(request, response); */
%>
</body>
</html>
//scopeResult.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
```

```

<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
//읽어오기
/*
    값을 모두 삭제하기 위해서 크롬을 모두 닫은 다음에 톰캣 서버를 제시작 한다.
*/
out.println((String)pageContext.getAttribute("page1"));
out.println((String)request.getAttribute("request1"));
out.println((String)session.getAttribute("session1"));
out.println((String)application.getAttribute("application1"));
%>
</body>
</html>

//MyScope.java
package com.human.ex;
import java.io.IOException;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
@WebServlet("/MyScope")
public class MyScope extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public MyScope() { super(); }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //page 메소드 안에 변수를 생성해서 사용한다.
        String page2="hello page2";
        request.setAttribute("request2", "hello request2");
        HttpSession session=request.getSession();
        session.setAttribute("session2", "hello session2");
        //getServletContext();
        ServletContext application = getServletContext();
        application.setAttribute("application2", "hello application2");

        System.out.println(page2);
        System.out.println((String)request.getAttribute("request2"));
        System.out.println((String)session.getAttribute("session2"));
        System.out.println((String)application.getAttribute("application2"));
    }
}

```

```
        response.sendRedirect("scopeResult.jsp");
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request, response);
    }
}
```

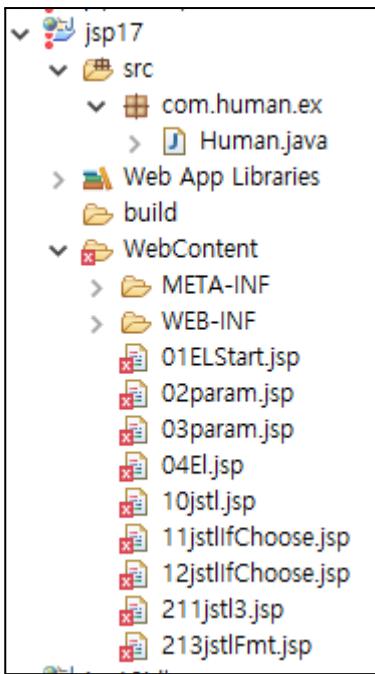
문제1) 장바구니 프로그램을 구현해 보자.

문제2) 채팅프로그램을 구현해 보자.

---

## > 13. el and jstl

---



el은 jsp에서 제공하는 각종 데이터를 화면에 출력하는데 사용한다.

jstl은 jsp에서 자바 코드를 걷어 내기 위해서 사용한다.

JSP(JavaServer Pages) EL(Expression Language)은 JSP 페이지에서 데이터를 표현하고 조작하기 위한 스크립트 언어입니다. EL은 JSP의 간결성과 가독성을 향상시키고, 데이터 접근과 연산을 간편하게 처리할 수 있도록 도와줍니다.

EL은 \${} 기호를 사용하여 중괄호 안에 기술한 데이터를 기술 한다. EL을 사용하여 데이터를 출력하거나 jstl 안에서 조건문, 반복문, 연산 등을 수행할 수 있습니다.

```
//01ELStart.jsp
<%@ page language="java" contentType="text/html;
charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
```

```
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
```

```
<!-- 표현어 (Expression Language) 값을 쉽게 가져오기 위해서 사용
다음 3가지 방법은 모두 같은 방법인데 앞으로 el를 사용하자.-->
```

```
 ${"hello" }<br>
<%="hello" %><br> <!-- 표현식 -->
<%out.println("hello"); %><!-- 스크립틀릿 -->
```

```
<!-- EL로 표시할수 있는 자료형 -->
```

```
정수형:${10 }<br>
```

```
실수형:${20.3 }<br>
```

```
문자형:${"park" }<br>
```

```
논리형:${true }<br>
```

```
null :${null}<br>
```

```
<!-- 일반적으로 null을 찍어면 null이라고 나오지만 el에서는 아무것도 출력되지 않는다. -->
```

```
<br>
```

```
<!-- EL 연산 -->
```

```
\${5+2} =${5+2}<br>
```

```

\${5/2} =${5/2}<br>
<%-- \${5 div 2} =${5 div 2}<br> --%>
\${5 mod 2} =${5 mod 2}<br>
\${5>2} =${5>2}<br>
\${2 gt 10} =${2 gt 10}<br>
\${5>2?5:2} =${5>2?5:2}<br>
\${(5>2)||(2<10)} =${(5>2)||(2<10)}<br>
</body>
</html>

```

다음 예시는 EL(Expression Language)을 사용하여 사용자의 입력을 받아 처리하는 객체를 제공한다.

param 객체는 request.getParameter로 하나의 사용자 입력을 처리하는 객체이고, paramvalues 객체는 request.getParameterValues로 여러 개의 사용자 입력을 처리하는 객체이다.

```

<%=request.getParameter("name") %><br>
${param.name }<br>

request.getParameterValues("animal")
${paramValues.animal[0]}<br>

```

02param.jsp 파일에서는 <form> 태그를 통해 사용자로부터 입력값을 받아 데이터를 "03param.jsp"로 전송 됩니다.

```

//02param.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<!-- EL getParameter 처리방법 -->
<form method="get" action="03param.jsp">
<br>      설명 : <input type="text" name="name">
<br>      숫자1:<input type="text" name="num1">
<br>      숫자2:<input type="text" name="num2">
<br>      <input type="submit" value="로그인">
</form>
<form action="03param.jsp">
아이디:<input type="text" name="id"><br>
다음중 회원이 키우고 있는 애완동물을 선택하십시오
개:<input type="checkbox" name="animal" value="dog">

```

```

고양이:<input type="checkbox" name="animal" value="cat">
금붕어:<input type="checkbox" name="animal" value="fish">
<input type="submit" value="send">
</form>
</body>
</html>

```

03param.jsp 파일에서는 스크립트릿(<% %>)과 EL을 사용하여 입력값을 처리합니다. \${param.name}은 "name" 파라미터의 값을 가져옵니다. \${param.num1}과 \${param.num2}는 각각 "num1"과 "num2" 파라미터의 값을 가져옵니다.

또한, \${paramValues.animal[0]}과 같이 \${paramValues.animal}를 사용하여 여러 개의 값을 배열 형태로 가져올 수 있습니다.

```

//03param.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
(스크립틀립)<br>
설명 : <%=request.getParameter("name") %><br>
숫자1 :<%=request.getParameter("num1") %> <br>
숫자2 : <%=request.getParameter("num2") %><br>
(EL방식)<br>
설명 : ${param.name }<br>
숫자1 :${param.num1} <br>
숫자2 :${param.num2 } <br>

```

사용자 입력 중 <br>

한개의 데이터는 param에 받고 여러개의 데이터는 paramValues로 받는다.

```

아이디:${param.id }:${param["id"] }<br>
//request.getParameterValues("animal")

```

선택한동물:\${paramValues.animal[0]}<br>

\${paramValues.animal[1]}<br>

\${paramValues.animal[2]}<br>

\${paramValues['animal'][0] }<br>

\${paramValues['animal'][1] }<br>

\${paramValues['animal'][2] }<br>

나중에 jstl배워서 반복문으로 출력하면 유동적인 데이터를 처리할 수 있다.

<br>

\${empty str1}과 \${not empty str2}는 해당 변수가 비어 있는지 여부를 확인하는 EL의

조건문 예시입니다.

empty 널인지 비어 있는지 확인하여 true, false값이 리턴

```
<%
    request.setAttribute("str1", null);
    request.setAttribute("str2", "hello");
%>
${empty str1}==true
${empty str2}==false
${not empty str2}==true
```

null를 넣었을때<br>

```
request null : <%=request.getParameter("id") %>:null찍힘<br>
EL null : ${param.id }:null안찍힘<br>
```

(스크릿틀립)<br>

```
<%=request.getParameter("name").equals("park") %><br>
```

(EL방식)<br>

```
EL ==연산자 사용 결과 : ${param.name=="park" }<br>
```

(스크릿틀립)<br>넘어온값 연산<br>

```
<br>
```

```
<%
```

```
int num1=Integer.parseInt(request.getParameter("num1"));
int num2=Integer.parseInt(request.getParameter("num2"));
%>
```

```
<%=num1 %>+<%=num2 %>=<%=num1 + num2 %> <br>
```

el 사용한 방법<br>

EL에서는 \${param.num1 + param.num2}와 같이 직접 연산을 수행할 수 있습니다.

```
${param.num1 }+${param.num2 }=${param.num1+param.num2 }<br>
```

```
</body>
```

```
</html>
```

EL을 사용하면 JSP에서 간결하고 가독성이 좋은 코드를 작성할 수 있으며, 사용자의 입력값을 쉽게 처리할 수 있습니다.

EL은 다양한 Scope(범위)에서 객체에 접근할 수 있도록 다음과 같은 객체를 지원 합니다.

pageScope: 페이지 범위에 속하는 객체에 접근합니다.

requestScope: 요청 범위에 속하는 객체에 접근합니다.

sessionScope: 세션 범위에 속하는 객체에 접근합니다.

applicationScope: 어플리케이션 범위에 속하는 객체에 접근합니다.

EL을 사용하여 Scope에서 객체에 접근하려면 \${} 표현식을 사용합니다. \${} 안에는 객체의 이름을 작성합니다. 다음은 사용 예제이다.

```
 ${pageScope.message}
 ${requestScope.message}
 ${sessionScope.message}
```

```

${applicationScope.message}

//Human.java
package com.human.ex;
public class Human {
    private String name;
    private int age;
    public Human() {}
    public Human(String name, int age) {
        this.name = name;      this.age = age;
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + age;
        result = prime * result + ((name == null) ? 0 : name.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Human other = (Human) obj;
        if (age != other.age)
            return false;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        return true;
    }
    @Override
    public String toString() {
        return "Human [name=" + name + ", age=" + age + "]";
    }
    public String getName() {      return name;      }
    public void setName(String name) {      this.name = name;  }
    public int getAge() {      return age;  }
    public void setAge(int age) {      this.age = age;  }
}

```

```
//04EL.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
EL에서 scope별 접근방법
scope          특정 객체의 생명주기
page          해당 페이지에서만 사용할수 있는 객체
request        요청이 종료될때까지 사용할 수 있는 객체
session        특정 유저만 사용할수 있는 객체
application    모든 유저가 사용할 수 있는 객체
<%
pageContext.setAttribute("name","page Name1");
request.setAttribute("name","page Name2");
session.setAttribute("name","page Name3");
application.setAttribute("name","page Name4");
%>
page 속성: ${pageScope.name }<br>
request 속성 : ${requestScope.name }<br>
session 속성 : ${sessionScope.name }<br>
application 속성 : ${applicationScope.name }<br>
name : ${name }<br>
scope 객체 없이 필드만 기술하면 page,request,session,application순으로 검색 하여
검색된 위치에 있는 필드 값이 출력 된다.
```

사용하지 않을 것을 권장

객체를 데이터로 넣으면 .를 찍어서 접근할 수 있다.

```
<%
      com.human.ex.Human human=new com.human.ex.Human("park",16);
      request.setAttribute("data",human);
%>
EL<br>
${requestScope.data.name }<br>
${requestScope.data.age }<br>
</body>
</html>
```

jstl은 jsp에서 자바 코드를 제거 하는데에 사용한다.

다음과 같이 설치해야 사용 할 수 있다.

1. <https://tomcat.apache.org/> 사이트에 들어가면 다음 이미지 와 같은 사이트가 나온다. 왼쪽 메뉴에서 taglib 를 선택하고 오른쪽 화면에서 Appache Standard Taglib 를 선택 하자.

2. 아래 이미지 화면이 나오면 체크되어 있는 download 부분을 클릭 한다.

Version	JSTL version	Requirements	Getting the Taglib
Standard 1.2.3	JSTL 1.2	Servlet 2.5, JavaServer Pages 2.1	<a href="#">download</a> <a href="#">(javadoc)</a>
Standard 1.1	JSTL 1.1	Servlet 2.4, JavaServer Pages 2.0	<a href="#">download</a> <a href="#">(javadoc)</a>
Standard 1.0	JSTL 1.0	Servlet 2.3, JavaServer Pages 1.2	<a href="#">download</a> <a href="#">(javadoc)</a>

3. 다음 이미지와 같은 화면이 나오면 binaries 폴더를 선택한다.

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>		-	
<a href="#">binaries/</a>	2005-10-05 20:39	-	
<a href="#">source/</a>	2005-10-05 20:38	-	

4. 폴더 안의 파일중에 아래 파일을 찾아 다운로드 해보자.

5. 다운로드한 파일의 압축을 풀고 아래 이미지에서 해당하는 폴더의 lib 폴더 안에 있는 jstl.jar와 standard.jar를 복사한다.

6. 새로 만든 프로젝트에서 jstl을 사용하려면 이전에 복사한 파일을 왼쪽 이미지처럼 webContent 밑에 web-inf 폴더 밑에 lib 폴더안에 복사한다. lib 폴더가 없다면 만들어서 복사해 넣으면 새로 만든 프로젝트에서 jstl을 사용할 수 있다. 새로 프로젝트를 만들 때마다 상위 작업을 해야 새로 만든 프로젝트에서 jstl을 사용할 수 있다.

다음 코드는 JSP 페이지에서 JSTL(Core)을 사용하여 EL을 통해 값을 출력하고 설정하는 예제입니다. 아래는 코드의 각 부분에 대한 설명입니다.

jstl을 사용하려면 다음 태그를 사용하고자하는 jsp파일 상단에 추가해야 한다.

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

다음과 같은 태그들을 이용해서 jstl을 사용 할 수 있다.

<c:out>은 value값의 내용을 화면에 출력 시킨다. 출력하고자 하는 문자열에 특수 문자들이 있어서 출력하는데 문제가 될때 사용하면 좋다.

```
<c:out value="hello world!"/><br>
<c:out value="${param.name }"/><br>
${param.name}은 요청 매개 변수 "name"의 값을 읽어옵니다.
```

<c:set>를 이용해서 변수에 값을 세팅할 수 있다.

```
<c:set var="msg" value="hello"/>
\${msg }=\${msg }<br>
```

```
<c:set var="msg" value="${name }"/>
\${msg }=\${msg }<br>
```

```
<c:set var="add" value="\${10+5 }"/>
\${add }=\${add }<br>
```

```
<c:set var="flag" value="\${20>5 }"/>
\${flag}=\${flag }<br>
```

<c:out> 태그를 이용해서 객체 접근하는 방법은 다음과 같다.

<c:set> 태그를 사용하여 변수 "member"에 Java 코드를 통해 생성된 객체를 설정하고, <c:set>과 \${member}를 사용하여 객체의 속성(name, age)에 접근 합니다.

```
<c:set var="member" value="<%=new com.human.ex.Human() %>" /> 선언
<c:set target="\${member }" property="name" value="홍길동" /> setter
<c:set target="\${member }" property="age" value="15" />
e1사용법
\${member } = \${member } <br>                               toString
\${member.age} = \${member.age } <br>                         getter
```

//10jstl.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
```

```

<!-- 기본 확인 -->
<!-- out은 속성 value에 값을 화면에 출력한다. -->
<c:out value="hello world!"/><br>
<c:out value="name:${param.name }"/><br>

<!-- set은 속성 value에 값을 var에 넣는다. -->
<c:set var="msg" value="hello"/>
\${msg }=\${msg }<br>

<c:set var="msg" value="\${name }"/>
\${msg }=\${msg }<br>

<c:set var="age" value="30" />
\${age }=\${age }<br>
<c:set var="age">
30
</c:set>
\${age }=\${age }<br>

<c:set var="member" value="<%new com.human.ex.Human()%"/>
<c:set target="\${member }" property="name" value="홍길동"/>
<c:set target="\${member }" property="age" value="15"/>
\${member } = \${member } <hr>
\${member.age} = \${member.age } <hr>

<c:set var="add" value="\${10+5 }"/>
\${add }=\${add }<br>
<c:set var="flag" value="\${20>5 }"/>
\${flag}=\${flag }<br>

</body>
</html>

```

### jstl를 사용한 if문과 choose문

JSTL(JSP Standard Tag Library)의 `<c:if>` 태그는 조건을 평가하고, 조건이 참인 경우에만 내부의 코드를 실행하는 기능을 제공합니다.

`<c:if>` 태그의 기본 구문은 다음과 같습니다.

```

<c:if test="조건식">
    실행할 코드
</c:if>

```

`test` 속성은 평가할 조건식을 지정합니다. 조건식은 EL(Expression Language)을 사용하여 표현합니다.

조건식이 참(true)인 경우, `<c:if>` 태그 내부의 코드가 실행됩니다.

조건식이 거짓(false)인 경우, <c:if> 태그 내부의 코드는 무시되고 넘어갑니다.

예제는 다음과 같다.

```
<c:if test="${userType eq 'admin'}">
    <p>관리자 페이지에 접속할 수 있습니다.</p>
</c:if>
```

위의 예제는 userType 변수의 값이 'admin'인 경우에만 <p> 태그 내부의 메시지가 출력됩니다. 즉, 사용자의 유형이 관리자인 경우에만 해당 메시지가 표시됩니다.

<c:if> 태그를 사용하여 조건에 따라 동적인 처리를 수행할 수 있으며, 다양한 조건식을 사용할 수 있습니다. 조건식에는 비교 연산자, 논리 연산자, EL 변수 등을 사용하여 복잡한 조건을 표현할 수 있습니다.

<c:choose> 문 설명과 예제:

<c:choose> 태그는 조건에 따라 다른 코드 블록을 실행하는 기능을 제공합니다.  
<c:choose> 태그 내부에는 <c:when>과 <c:otherwise> 태그를 자유롭게 조합하여 사용할 수 있습니다.  
<c:when> test속성의 조건이 참인 경우에 해당하는 코드 블록을 실행합니다.  
여러 개의 <c:when> 태그를 사용하여 다중 조건을 처리할 수 있습니다.  
<c:otherwise> 태그는 어떤 조건에도 해당하지 않는 경우에 실행할 코드 블록을 정의합니다.

<c:choose> 태그의 기본 구문은 다음과 같다

```
<c:choose>
    <c:when test="조건식1">
        실행할 코드1
    </c:when>
    <c:when test="조건식2">
        실행할 코드2
    </c:when>
    ...
    <c:otherwise>
        기본 실행할 코드
    </c:otherwise>
</c:choose>
```

```
<c:choose>
    <c:when test="${userType eq 'admin'}">
        <p>관리자 페이지에 접속할 수 있습니다.</p>
    </c:when>
    <c:when test="${userType eq 'member'}">
        <p>회원 페이지에 접속할 수 있습니다.</p>
    </c:when>
    <c:otherwise>
        <p>접속 권한이 없습니다.</p>
    </c:otherwise>
```

```
</c:otherwise>  
</c:choose>
```

eq는 == 과 같은 의미이다.

위의 예제는 userType 변수의 값에 따라 다른 메시지를 출력합니다. userType 값이 'admin'인 경우에는 관리자 페이지에 접속할 수 있는 메시지가 출력되고, 'member'인 경우에는 회원 페이지에 접속할 수 있는 메시지가 출력됩니다. 그 외의 경우에는 접속 권한이 없다는 메시지가 출력됩니다.

<c:choose> 태그를 사용하여 다중 조건을 처리할 수 있으며, 각 조건에 따라 실행할 코드 블록을 유연하게 작성할 수 있습니다.

#### foreach

JSTL의 <forEach> 태그는 컬렉션 객체나 배열의 각각의 요소를 하나씩 반복적으로 처리하는데 사용됩니다. 반복문을 대체하는 역할을 한다.

**var:** 현재 반복 요소를 참조할 변수의 이름을 지정 합니다. 예를 들어, var="item"이라고 하면 각 요소는 item이라는 변수로 참조 됩니다. item은 반복할 데이터들중 반복될때마다 새로운 하나의 데이터를 가지고 있다. items에 있는 값중 하나의 값을 반복적으로 교체하면서 반복된다.

**items:** 반복할 컬렉션 객체나 배열을 지정합니다.

**begin, end, step:** 선택적으로 사용할 수 있는 속성으로, 반복 범위를 지정합니다. begin은 시작 인덱스, end는 끝 인덱스, step은 증가량을 의미합니다.

**varStatus:** 선택적으로 사용할 수 있는 속성으로, 반복 상태를 저장하는 객체의 이름을 지정합니다. 주로 반복 횟수, 인덱스 등을 추적할 때 사용됩니다.

JSTL을 사용하려면 JSTL 라이브러리를 프로젝트에 추가하고, JSP 페이지에서 해당 라이브러리를 태그 라이브러리로 선언해야 합니다.

<forEach> 태그의 예시 코드는 다음과 같습니다:

<forEach> 태그와 배열(Array) 사용:

```
<c:set var="myArray" value="${['apple', 'banana', 'orange']} />  
<c:forEach items="${myArray}" var="fruit">  
    <li>${fruit}</li>  
</c:forEach>
```

위의 예시에서는 배열 ['apple', 'banana', 'orange']를 myArray 변수에 할당하고, <forEach> 태그를 사용하여 배열을 반복하여 배열 요소를 fruit 변수로 참조하여 출력합니다.

```
<c:forEach items="${userList}" var="user">  
    <li>  
        <c:url value="/user/profile">  
            <c:param name="userId" value="${user.id}" />  
        </c:url>  
        <a href="${url}">${user.name}</a>
```

```
</li>
</c:forEach>
```

위의 예시에서는 \${userList}의 각 요소를 반복하면서 <c:url> 태그를 사용하여 동적인 URL을 생성합니다. value="/user/profile"을 통해 경로를 설정하고, <c:param> 태그를 사용하여 매개변수를 설정합니다. 그 후 <a> 태그의 href 속성에 생성된 URL을 할당하여 링크를 출력합니다.

```
<c:forEach items="${myList}" var="item">
  <c:if test="${item > 5}">
    <li>${item}</li>
  </c:if>
</c:forEach>
```

위의 예시에서는 \${myList}의 각 요소를 검사하여 해당 요소가 5보다 큰 경우에만 출력합니다. <c:if> 태그를 사용하여 요소를 조건부로 처리할 수 있습니다.

varStatus는 반복 작업 중에 현재 인덱스, 반복 횟수, 첫 번째 요소 여부 등을 제공합니다. 다음은 varStatus 속성을 사용하는 예시 코드입니다:

```
<c:forEach items="${myList}" var="item" varStatus="status">
  <li>Index: ${status.index+1}, Count: ${status.count}, First: ${status.first},
Last: ${status.last}</li>
</c:forEach>
```

위의 예시에서는 \${myList}의 요소를 반복하면서 item 변수에 각 요소를 할당하고, varStatus 속성을 사용하여 status 변수에 현재 반복의 상태를 할당합니다. status.index는 현재 인덱스, status.count는 반복 횟수, status.first는 첫 번째 요소 여부, status.last는 마지막 요소 여부를 나타냅니다. 이를 통해 반복 작업 중에 현재 반복의 상태를 활용할 수 있습니다.

```
<c:forEach items="${myList}" var="item">
  <c:out value="${item}" />
</c:forEach>
```

위의 예시에서는 \${myList}의 각 요소를 출력합니다. <c:out>은 특수 문자를 이스케이프하고 HTML 문자 엔터티를 처리하는 등의 추가적인 기능을 제공합니다.

step 속성: 이 속성은 반복문에서 요소를 건너뛸 때 사용됩니다. 예를 들어, step="2"로 설정하면 요소를 2개씩 건너뛰며 반복합니다. 다음은 step 속성을 사용하는 예시 코드입니다

```
<c:forEach items="${myList}" var="item" step="2">
  <li>${item}</li>
</c:forEach>
```

위의 예시에서는 \${myList}의 요소 중에서 인덱스가 0, 2, 4, ... 인 요소만 반복하여 출력합니다.

step, begin, end 속성의 함께 사용: step, begin, end 속성을 함께 사용하여 반복 범위를 조정할 수 있습니다. 예를 들어, begin="1", end="10", step="2"로 설정하면 1부터 10까지의 요소 중에서 인덱스가 홀수인 요소만 반복합니다. 다음은 이러한 속성의 조합을 사용하는 예시 코드입니다:

```
<c:forEach items="${myList}" var="item" begin="1" end="10" step="2">
  <li>${item}</li>
</c:forEach>
```

위의 예시에서는 \${myList}의 요소 중에서 인덱스가 1, 3, 5, ..., 9인 요소만 반복하여 출력합니다.

```
<c:forEach var="item" begin="1" end="10" step="2">
    <li>${item}</li>
</c:forEach>
```

위의 예시에서는 \${item}에 1, 3, 5, ..., 9인 데이터가 담겨 반복하여 출력합니다.

```
<c:forEach items="${myList}" var="item" step="-1">
    <li>${item}</li>
</c:forEach>
```

위의 예시에서는 \${myList}의 요소를 역순으로 반복하며 출력합니다. step="-1"을 사용하여 역순으로 반복하도록 설정합니다.

JSTL의 <forEach> 태그를 활용하여 반복 작업을 처리할 때, 조건문과 선택문을 함께 사용하여 원하는 동작을 지정할 수 있습니다. <c:if>, <c:choose>, <c:when>, <c:otherwise> 태그와 함께 사용하여 더욱 다양한 로직을 구현할 수 있습니다.

다음 코드는 사용자로부터 선택한 색상과 항목을 처리하기 위한 JSP 페이지입니다. 사용자에게 색상과 항목을 선택하도록 폼을 제공합니다. 사용자가 선택한 값을 "12jstlIfChoose.jsp"로 전송합니다. 11jstlIfChoose.jsp 파일은 사용자가 색상과 아이템을 선택하는 페이지이고, 사용자가 선택한 색상과 항목을 "12jstlIfChoose.jsp"라는 JSP 페이지에 선택된 결과를 출력한다.

```
//11jstlIfChoose.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action="12jstlIfChoose.jsp">
색상을 선택하세요<br>
    <select id="color" name="color">
        <option value="1" > 빨강</option>
        <option value="2" > 노랑</option>
        <option value="3" > 파랑</option>
    </select>
    <input type="submit" value="전송">
</form>
<form method="get" action="12jstlIfChoose.jsp">
    <input type="checkbox" name="items" value="신발">신발
    <input type="checkbox" name="items" value="가방">가방

```

```

<input type="checkbox" name="items" value="벨트">벨트
<input type="checkbox" name="items" value="모자">모자
<input type="checkbox" name="items" value="시계">시계
<input type="checkbox" name="items" value="쥬얼리">쥬얼리
<input type="submit" value="전송">

</form>
</body>
</html>

//12jstlIfChoose.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<c:if test="${param.color==1 }">
    <span style="color:red;">빨강</span>
</c:if>
<c:if test="${param.color==2 }">
    <span style="color:yellow;">노랑</span>
</c:if>
<c:if test="${param.color==3 }">
    <span style="color:blue;">파랑</span>
</c:if>
<c:choose>
    <c:when test="${param.color==1 }">
        <span style="color:red;">빨강</span>
    </c:when>
    <c:when test="${param.color==2 }">
        <span style="color:yellow;">노랑</span>
    </c:when>
    <c:when test="${param.color==3 }">
        <span style="color:blue;">파랑</span>
    </c:when>
    <c:otherwise>
        <span style="font-weight:bold;color:yellow;">무색</span>
    </c:otherwise>
</c:choose>

```

```

<hr>
당신이 선택한 항목입니다.
<c:forEach var="item" items="${paramValues.items }" varStatus="status">
    ${item }<c:if test="${not status.last }">,</c:if>
</c:forEach>
</body>
</html>

```

```

//13jstlForEach.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    String[] movieList={"아바타","ET","공주는 외로워","쥬라기공원"};
    pageContext.setAttribute("movieList", movieList);
%>
<c:forEach var="movie" items="${movieList }">
    ${movie }<br>
</c:forEach>
<br>
<table border="1" style="width:100%;text-align:center;">
<tr>
    <th>index</th><th>count</th><th>title</th>
</tr>
<c:forEach var="movie" items="${movieList }" varStatus="status">
    <tr>
        <td>${status.index}</td><td>${status.count }</td><td>${movie
}</td>
    </tr>
</c:forEach>
</table>
<br>
<ul>
    <c:forEach var="movie" items="${movieList }" varStatus="status">
        <c:choose>
            <c:when test="${status.first }">
                <li style="font-weight:bold;color:red;">${movie }</li>

```

```

        </c:when>
        <c:when test="${status.last }">
            <li style="font-weight:bold;color:blue;">${movie }</li>
        </c:when>
        <c:otherwise>
            <li style="font-weight:bold;color:yellow;">${movie }</li>
        </c:otherwise>
    </c:choose>
</c:forEach>
</ul>
<br>
<c:forEach var="movie" items="${movieList }" varStatus="status">
    ${movie }<c:if test="${not status.last }">,</c:if>
</c:forEach>
<br>
<c:forEach var="cnt" begin="0" end="10" varStatus="status">
    ${status.index }:${cnt }<c:if test="${not status.last }">,</c:if>
</c:forEach>
<br>
<c:forEach var="cnt" begin="7" end="10" varStatus="status">
    ${status.index }:${status.count }:${cnt }<c:if test="${not status.last }">,</c:if>
</c:forEach>
<br>
<c:forEach var="cnt" begin="2" end="10" varStatus="status" step="2">
    ${status.index }:${status.count }:${cnt }<c:if test="${not status.last }">,</c:if>
</c:forEach>
<br>

```

`forTokens` 태그는 주어진 문자열을 구분자(delimiter)를 기준으로 분리하여 각각의 토큰(token)으로 처리합니다. 이를 통해 문자열을 분리하고 각 토큰에 대해 작업을 수행할 수 있습니다.

아래는 `forTokens` 태그의 기본 구문입니다:

```
<c:forTokens var="변수명" items="문자열" delims="구분자">
    <!-- 작업 수행 -->
```

`var:` 각 토큰을 할당할 변수의 이름을 지정합니다.

`items:` 분리할 대상 문자열을 지정합니다.

`delims:` 구분자를 지정합니다. 구분자는 토큰을 분리하는 기준이 됩니다.

예를 들어, 다음과 같은 문자열을 공백을 구분자로 나누어 각각의 토큰에 대해 작업하고자 한다면:

```
<c:forTokens var="token" items="Hello World JSTL" delims=" ">
    <c:out value="${token}"/>
</c:forTokens>
```

위 코드는 "Hello", "World", "JSTL"이라는 세 개의 토큰으로 분리된 후, 각 토큰을

출력합니다. 따라서 출력 결과는 다음과 같을 것입니다:

Hello

World

JSTL

이와 같이 `forTokens` 태그를 사용하면 문자열을 구분자를 기준으로 나누어 각각의 토큰에 대해 작업할 수 있습니다. JSTL의 다른 태그들도 유용한 기능을 제공하므로, 프론트엔드 개발에서 JSP를 사용할 경우 JSTL을 적극적으로 활용할 수 있습니다.

```
<c:forTokens var="city" items="서울.인천.대구.부산" delims=",">
    ${city }<br>
</c:forTokens>
<c:forTokens var="city" items="서울.인천.대구.부산" delims=". .">
    ${city }<br>
</c:forTokens>
<hr><hr><hr>

<hr><hr><hr>
    <c:url value="images/pic.jpg" var="data"/>
    <h3>${data }</h3>
    
<hr>

<%--
    <c:redirect url="http://www.google.com"></c:redirect>
--%>
</body>
</html>
```

`fmt`는 데이터를 변환하여 문자를 출력할 때 사용한다. 아래와 같은 코드를 사용 한다.

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
//14jstlFmt.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<pre>
<c:set var="now" value="<%=java.time.LocalDateTime.now()%>"></c:set>
\${now } : ${now }

<fmt:parseDate value="<%=java.time.LocalDateTime.now().toString()%>">
```

```

pattern="yyyy-MM-dd'T'HH:mm:ss.SSS"
var="now" type="both" />

<fmt:formatDate value="${now }"></fmt:formatDate>
date :<fmt:formatDate value="${now }" type="date"/>
time :<fmt:formatDate value="${now }" type="time"/>
both :<fmt:formatDate value="${now }" type="both"/>

```

pattern="yyyy년 MM월 dd일 hh시 mm분 ss초" :

```

<fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일 hh시 mm분
ss초"></fmt:formatDate>
</pre>

```

```

<%
//jstl에서 localdatetime클래스를 이용해서 현재시간을 처리하는 방법을 다루고 있다.
    java.time.LocalDateTime ld=java.time.LocalDateTime.now();
    request.setAttribute("ld", ld);
%>
<hr>
-${requestScope.ld }-
<fmt:parseDate value="${requestScope.ld }"
    pattern="yyyy-MM-dd'T'HH:mm:ss.SSS"
    var="now" type="both" />
<hr>
<fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일 hh시 mm분
ss초"></fmt:formatDate>
<hr>

<!-- 2일차 -->
 timeZone 설정으로 특정 지역의 시간을 변경해서 출력할 수 있다.
default : <c:out value="${now }"></c:out>
Korea KST : <fmt:formatDate value="${now }" type="both" dateStyle="full"
    timeStyle="full"></fmt:formatDate>
<fmt:timeZone value="GMT">
Swiss GMT : <fmt:formatDate value="${now }" type="both" dateStyle="full"
    timeStyle="full"></fmt:formatDate>
</fmt:timeZone>
<fmt:timeZone value="GMT-8">
NewYork GMT-8 : <fmt:formatDate value="${now }" type="both" dateStyle="full"
    timeStyle="full"></fmt:formatDate>
</fmt:timeZone>

```

로케일 사용자 사용환경을 확인할 수 있다.

톰캣 서버의 기본 로케일 : <%=response.getLocale() %>

```

<fmt:setLocale value="ko_kr"/>

```

```

ko_kr 기본 로케일 : <%=response.getLocale() %>

통화(currency) : <fmt:formatNumber value="10000"
type="currency"></fmt:formatNumber>
날짜 : <fmt:formatDate value="${now }"></fmt:formatDate>
<fmt:setLocale value="ja_jP"></fmt:setLocale>
ja_jP 기본 로케일 : <%=response.getLocale() %>
통화(currency) : <fmt:formatNumber value="10000"
type="currency"></fmt:formatNumber>
날짜 : <fmt:formatDate value="${now }"></fmt:formatDate>
<fmt:setLocale value="en_US"></fmt:setLocale>
en_US 기본 로케일 : <%=response.getLocale() %>
통화(currency) : <fmt:formatNumber value="10000"
type="currency"></fmt:formatNumber>
날짜 : <fmt:formatDate value="${now }"></fmt:formatDate>

<!-- 인코딩 -->
<fmt:requestEncoding value="utf-8"/>
<test></test>
<test/>

<fmt:formatNumber value="10000" type="currency"/><br>
<fmt:formatNumber value="10000" type="number"></fmt:formatNumber><br>
<fmt:formatNumber value="10000" type="percent"></fmt:formatNumber><br>
<hr>
<fmt:formatNumber value="1234567.89"></fmt:formatNumber><br>
<fmt:formatNumber value="1234567.89" type="number">
</fmt:formatNumber><br>
<fmt:formatNumber value="1234567.89" type="number" groupingUsed="false">
</fmt:formatNumber><br>
<hr>
<fmt:formatNumber value="10000" type="currency"
currencySymbol="$"></fmt:formatNumber><br>
<fmt:formatNumber value="10000" type="number"></fmt:formatNumber><br>
<fmt:formatNumber value="0.5" type="percent"></fmt:formatNumber><br>
<hr>
<fmt:formatNumber value="1234567.89"></fmt:formatNumber><br>
<fmt:formatNumber value="1234567.899"
pattern="#,#00.0#"></fmt:formatNumber><br>
<fmt:formatNumber value="1234567.89"
pattern="#,#00.0#"></fmt:formatNumber><br>
<fmt:formatNumber value="1234567.8"
pattern="#,#00.0#"></fmt:formatNumber><br>
<fmt:formatNumber value="1234567.8" pattern=".000"></fmt:formatNumber><br>

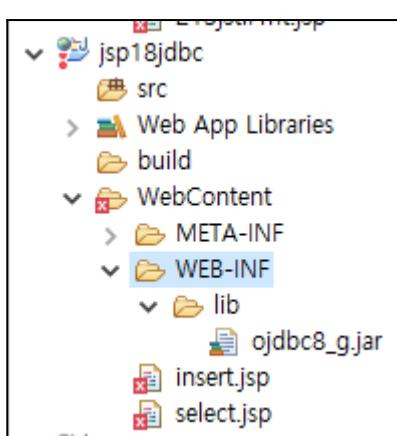
```

```
</body>  
</html>
```

1. 세션 관리프로그램을 e1과 jstl로 업그레이드 해보자.
2. 장바구니 프로그램을 e1과 jstl로 업그레이드 해보자.

## > 14. jdbc 데이터베이스 연결

다음 예제는 jsp에서 jdbc를 이용해서 DB작업을 하는 예제입니다.



```
//database
drop table member CASCADE CONSTRAINTS;
create table member(
    no number,
    name varchar(30),
    height number(6,3),
    birthday date
);
desc member;
insert into member values(1,'park1',123.21,sysdate);
insert into member values(2,'park2',123.22,sysdate);
insert into member values(3,'park3',123.23,sysdate);
```

commit; //commit를 이용해서 작업을 마무리 해야 한다.

WebContent/lib/폴더 및에 ojdbc8\_g.jar를 복사해 놓는다.

내 PC > 로컬 디스크 (C:) > app > admin > product > 18.0.0 > dbhomeXE > jdbc > lib				
이름	수정한 날짜	유형	크기	
ojdbc8.jar	2018-06-26 오후 1:50	JAR 파일	4,065KB	
ojdbc8_g.jar	2018-06-26 오후 1:56	JAR 파일	6,776KB	
ojdbc8dms.jar	2018-06-26 오후 1:50	JAR 파일	5,667KB	
ojdbc8dms_g.jar	2018-06-26 오후 1:59	JAR 파일	6,805KB	
simplefan.jar	2018-06-26 오후 12:06	JAR 파일	29KB	

다음 각각의 jsp 파일은 jdbc를 사용해서 JSP에서 CRUD관련 DB작업을 하는 예제이다.  
확인해 보자.

```
//select.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.time.LocalDateTime" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%!Connection connection;
    Statement statement;
    ResultSet resultSet;

    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    String uid = "c##human";
    String upw = "human";
    String query = "select * from member";%>
    <%
        try {
            Class.forName(driver);
            connection = DriverManager.getConnection(url, uid, upw);
            statement = connection.createStatement();
            resultSet = statement.executeQuery(query);
            while (resultSet.next()) {
                int no = resultSet.getInt("no");
                String name = resultSet.getString("name");
                double height= resultSet.getDouble("height");
                String birthday = resultSet.getString("birthday");
                out.println("번호 : " + no +"  
" + " 이름 : " + name
+ "<BR>" + "\n 키 : " + height +"  
"+ "\n 생일 : " + birthday+"<BR>"+<BR>");
            }
        } catch (Exception e) {
            out.println(e.getMessage());
        } finally {
            try {
                if (resultSet != null) {resultSet.close();}
                if (statement != null) {statement.close();}
                if (connection != null) {connection.close();}
            } catch (Exception e) {
                out.println(e.getMessage());
            }
        }
    %>
</body>
</html>

//insert.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.Statement"%>
```

```

<%@page import="java.sql.DriverManager"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    Connection connection;
    Statement statement;
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    String uid = "c##human";
    String upw = "human";
    String query= "insert into member values(6, '추가해보기',188.33,sysdate)";
%>
<%
try {
    Class.forName(driver);
    connection = DriverManager.getConnection(url, uid, upw);
    statement = connection.createStatement();
    statement.executeUpdate(query);
} catch (Exception e) {
    out.println(e.getMessage());
} finally {
    try {
        if (statement != null) {statement.close();}
        if (connection != null) {connection.close();}
    } catch (Exception e) {
    }
}
%>
    새로운데이터를 입력함
</body>
</html>
상위를 참조해서 CRUD작업을 완성해 보자.
//delete.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>

<!DOCTYPE html >
<html>

```

```

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<%!Connection connection;
Statement statement;
ResultSet resultSet;

String driver = "oracle.jdbc.driver.OracleDriver";
String url = "jdbc:oracle:thin:@localhost:1521:xe";
String uid = "c##human";
String upw = "human";
String query = "delete member where name='ceb2'";%>
<%
try {
    Class.forName(driver);
    connection = DriverManager.getConnection(url, uid, upw);
    statement = connection.createStatement();
    resultSet = statement.executeQuery(query);
} catch (Exception e) {
    out.println(e.getMessage());
} finally {
    try {
        if (resultSet != null) {resultSet.close();}
        if (statement != null) {statement.close();}
        if (connection != null) {connection.close();}
    } catch (Exception e) {
        out.println(e.getMessage());
    }
}
%>
</body>
</html>

//update.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.Connection"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>

```

```

</head>
<body>
    <%!Connection connection;
    Statement statement;
    ResultSet resultSet;

    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    String uid = "c##human";
    String upw = "human";
    String query = "update member set no=44 where name='ceb3'";%>
    <%
        try {
            Class.forName(driver);
            connection = DriverManager.getConnection(url, uid, upw);
            statement = connection.createStatement();
            resultSet = statement.executeQuery(query);
        } catch (Exception e) {
            out.println(e.getMessage());
        } finally {
            try {
                if (resultSet != null) {resultSet.close();}
                if (statement != null) {statement.close();}
                if (connection != null) {connection.close();}
            } catch (Exception e) {
                out.println(e.getMessage());
            }
        }
    %>
</body>
</html>

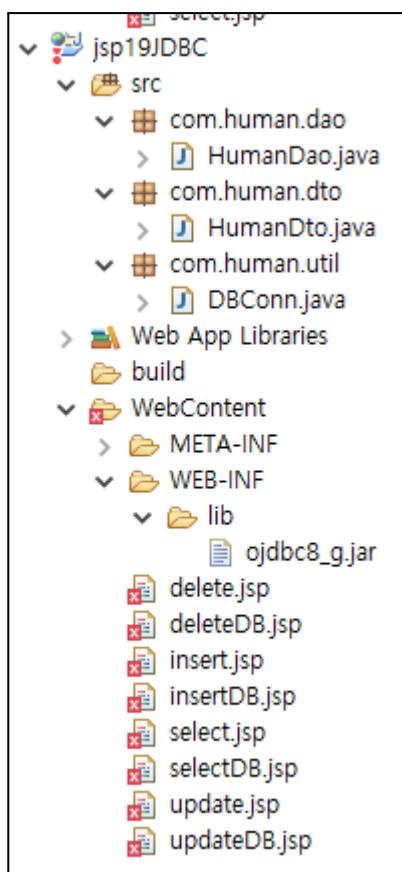
```

## > 15. jdbc model1

모델1은 데이터베이스 관련 코드를 dao 클래스로 만들고 데이터 베이스에서 읽어온 데이터를 dto클래스로 만들어 프로그램을 구현하는 개발 방식중 하나이다.

DBConn은 jdbc를 쉽게 사용하려는 클래스이고, HumanDto는 데이터베이스의 데이터를 담는 클래스이고, HumanDao는 Human테이블 관련 데이터베이스를 사용하기 위한 클래스이다.

insert.jsp, select.jsp, update.jsp, delete.jsp는 사용자로 부터 해당 작업에 필요한 입력을 받는 페이지이고 insertDB.jsp, selectDB.jsp, updateDB.jsp, deleteDB.jsp는 각각의 사용자 입력 데이터를 받아서 DB작업을 하는 페이지이다.



ojdbc8\_g.jar파일을 복사하는 작업을 잊지 말자.  
//sql  
drop table human;  
CREATE TABLE HUMAN (  
 NAME VARCHAR2(50),  
 AGE NUMBER(3),  
 TALL NUMBER(5,2),  
 BIRTH DATE  
);  
INSERT INTO HUMAN (NAME, AGE, TALL, BIRTH)  
VALUES ('Alice', 25, 170.5, TO\_DATE('1997-04-15',  
'YYYY-MM-DD'));  
INSERT INTO HUMAN (NAME, AGE, TALL, BIRTH)  
VALUES ('Bob', 33, 185.0, TO\_DATE('1989-12-27',  
'YYYY-MM-DD'));  
INSERT INTO HUMAN (NAME, AGE, TALL, BIRTH)  
VALUES ('Charlie', 42, 175.2, TO\_DATE('1980-09-03',  
'YYYY-MM-DD'));  
commit;

//com.human.util.DBConn.java jdbc작업을 쉽게 하기 위해서 만들어 놓은 클래스이다.

```
package com.human.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class DBConn {
    private static Connection con = null;
```

```

private static Statement st = null;
private static ResultSet rs = null;
private static String user = "c##human";
private static String pw = "human";
private static String url = "jdbc:oracle:thin:@localhost:1521:xe";
public static void connect()
{
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("드라이버 연결");
        con = DriverManager.getConnection(url,user,pw);
        System.out.println("데이터베이스에 접속 성공");
        st=con.createStatement();
    } catch (SQLException e)
    {
        e.printStackTrace();
    } catch (ClassNotFoundException e)
    {
        e.printStackTrace();
    }
}

public static void connect(String user, String pw)
{
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("드라이버 연결");

        con = DriverManager.getConnection(url,user,pw);
        System.out.println("데이터베이스에 접속 성공");
        st=con.createStatement();
    } catch (SQLException e)
    {
        e.printStackTrace();
    } catch (ClassNotFoundException e)
    {
        e.printStackTrace();
    }
}

public static ResultSet statementQuery(String sql)
{
    connect();
    try {
        rs = st.executeQuery(sql);
    } catch (SQLException e) {

```

```

        e.printStackTrace();
    }
    return rs;
}

public static void statementUpdate(String sql)
{
    connect();
    if(con == null)
    {
        System.out.println("fail");
    }else
    {
        try {
            int n = st.executeUpdate(sql);
            if(n != 0)
                System.out.println(n + "개의 작업을 성공적으로
수행했습니다");
            else
                System.out.println("작업된 내용이 없습니다");
        }catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}

public static void close() {
    try {
        if (rs != null)
            rs.close();
        if (st != null)
            st.close();
        if (con != null)
            con.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        rs = null;
        st = null;
        con = null;
    }
}
}

```

//com.human.dto.HumanDto.java dto클래스는 원하는 테이블의 하나의 데이터를 담을 용도로 만든 클래스 이다.

```
package com.human.dto;
import java.text.ParseException;
import java.util.Date;
public class HumanDto {
    private String name;
    private int age;
    private double tall;
    private Date birth;

    @Override
    public String toString() {
        return "HumanDto [name=" + name + ", age=" + age + ", tall=" +
tall + ", birth=" + birth + "]";
    }
    public HumanDto() {}
    public HumanDto(String name, int age, double tall, Date birth) {
        super();
        this.name = name;
        this.age = age;
        this.tall = tall;
        this.birth = birth;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public double getTall() {
        return tall;
    }
    public void setTall(double tall) {
        this.tall = tall;
    }
    public Date getBirth() {
        return birth;
    }
    public void setBirth(Date birth) {
        this.birth = birth;
    }
}
```

```

    }
    public String getBirthString() {
        String rValue="1990-01-01 00:00:00";
        java.text.DateFormat formatter=
            new java.text.SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
        if(birth!=null) {
            rValue= formatter.format(birth);
        }
        return rValue;
    }
    // "yyyy-MM-dd'T'HH:mm"
    public void setBirthString(String time) {
        java.text.DateFormat formatter=
            new java.text.SimpleDateFormat("yyyy-MM-dd'T'HH:mm");
        try {
            birth=formatter.parse(time);
        } catch (ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

//com.human.dao.HumanDao.java 클래스는 human작업 crud관련 작업을 하기 위한  
클래스이다.

```

package com.human.dto;

import java.text.ParseException;
import java.util.Date;

public class HumanDto {
    private String name;
    private int age;
    private double tall;
    private Date birth;

    @Override
    public String toString() {
        return "HumanDto [name=" + name + ", age=" + age + ", tall=" +
tall + ", birth=" + birth + "]";
    }
    public HumanDto() {}
    public HumanDto(String name, int age, double tall, Date birth) {
        super();
        this.name = name;
        this.age = age;
    }
}

```

```

        this.tall = tall;
        this.birth = birth;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public double getTall() {
        return tall;
    }
    public void setTall(double tall) {
        this.tall = tall;
    }
    public Date getBirth() {
        return birth;
    }
    public void setBirth(Date birth) {
        this.birth = birth;
    }
    public String getBirthString() {
        String rValue="1990-01-01 00:00:00";
        java.text.DateFormat formatter=
            new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        if(birth!=null) {
            rValue= formatter.format(birth);
        }
        return rValue;
    }
    // "yyyy-MM-dd'T'HH:mm"
    public void setBirthString(String time) {
        java.text.DateFormat formatter=
            new java.text.SimpleDateFormat("yyyy-MM-dd'T'HH:mm");
        try {
            birth=formatter.parse(time);
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
}

```

```

//select.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<!-- <script>localhost.href="selectDB.jsp";</script> -->
</head>
<body>
메뉴
<a href='insert.jsp'>입력</a><a href='update.jsp'>수정</a>
<a href='select.jsp'>검색</a><a href='delete.jsp'>삭제</a>
<form action="selectDB.jsp" method="get" >
    name:<input type="text" name="name"><br>
    <input type="submit" value="제출"><br>
</form>
</body>
</html>

//insert.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.jsp'>입력</a><a href='update.jsp'>수정</a>
<a href='select.jsp'>검색</a><a href='delete.jsp'>삭제</a>
<form action="insertDB.jsp" method="get" >
    name:<input type="text" name="name"><br>
    age:<input type="text" name="age"><br>
    height:<input type="text" name="height"><br>
    birthday:<input type="datetime-local" name="birthday"><br>
    <input type="submit" value="제출"><br>
</form>
</body>
</html>

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

```

```

pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.jsp'>입력</a><a href='update.jsp'>수정</a>
<a href='select.jsp'>검색</a><a href='delete.jsp'>삭제</a>
<form action="deleteDB.jsp" method="get" >
    name:<input type="text" name="name"><br>
    <input type="submit" value="제출"><br>
</form>
</body>
</html>

//update.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.jsp'>입력</a><a href='update.jsp'>수정</a>
<a href='select.jsp'>검색</a><a href='delete.jsp'>삭제</a>
<form action="updateDB.jsp" method="get" >
    name:<input type="text" name="name"><br>
    age:<input type="text" name="age"><br>
    <input type="submit" value="제출"><br>
</form>
</body>
</html>

//deleteDB.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>

```

```

</head>
<body>
<%
    com.human.dao.HumanDao dao=new com.human.dao.HumanDao();
    int result=dao.delete(request.getParameter("name"));
    if(result>=1){out.println("성공"); }else{
        out.println("실패");
    }
%>
메뉴
<a href='insert.jsp'>입력</a><a href='update.jsp'>수정</a>
<a href='select.jsp'>검색</a><a href='delete.jsp'>삭제</a>
</body>
</html>

//insertDB.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="com.human.dao.HumanDao" %>
<%@ page import="com.human.dto.HumanDto" %>
<%@ page import="java.time.LocalDateTime" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    HumanDto dto=new HumanDto();
    dto.setName(request.getParameter("name"));
    dto.setAge(Integer.parseInt(request.getParameter("age")));
    dto.setHeight(Double.parseDouble(request.getParameter("height")));
    // "yyyy-MM-dd'T'hh:mm"
    dto.setBirthday(LocalDateTime.parse(request.getParameter("birthday")));
    out.println(dto);
    HumanDao dao =new HumanDao();
    int i=dao.insert(dto);
    if(i==1){         out.println("입력 성공");      }else{
        out.println("문제 발생");
    }
%>
<!-- <script>
alert("완료");
Location.href="insert.jsp";
</script> -->
메뉴

```

```

<a href='insert.jsp'>입력</a><a href='update.jsp'>수정</a>
<a href='select.jsp'>검색</a><a href='delete.jsp'>삭제</a>
</body>
</html>

//selectDB.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="com.human.dao.HumanDao" %>
<%@ page import="com.human.dto.HumanDto" %>
<%@ page import="java.util.ArrayList" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.jsp'>입력</a><a href='update.jsp'>수정</a>
<a href='select.jsp'>검색</a><a href='delete.jsp'>삭제</a>
<br>
검색데이터<br>
<%
    HumanDao dao=new HumanDao();
    ArrayList<HumanDto> dtos =
        dao.select(request.getParameter("name"));
    if(dtos!=null){
        for(HumanDto dto:dtos){
            out.println(dto);
            out.println("<br>");
        }
    }
    System.out.println("end");
%>
</body>
</html>

//updateDB.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>

```

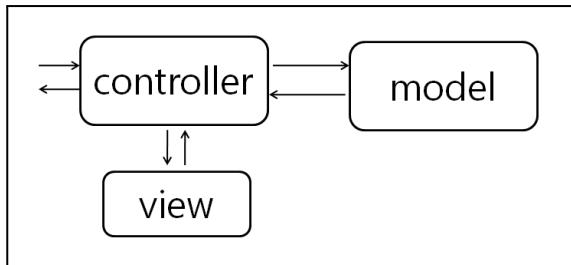
```
<body>

<%
    com.human.dao.HumanDao dao=new com.human.dao.HumanDao();
    int result=dao.update(request.getParameter("name"),
        request.getParameter("age"));
    if(result==1){           out.println("성공");           }else{
        out.println("실패");
    }
%>
<br>
메뉴
<a href='insert.jsp'>입력</a><a href='update.jsp'>수정</a>
<a href='select.jsp'>검색</a><a href='delete.jsp'>삭제</a>
</body>
</html>
```

## > 16. jdbc model2

model1은 데이터베이스 접근시 dto와 dao를 사용하여 구조화하여 만든 프로그램을 의미하고, model2는 mvc패턴을 이용한 방법이다.

model1를 frontcontroller와 jstl을 이용하여 model2로 만들어 보자.



mvc패턴이란? 프로그램을 구현할때 하나로 만들지 않고 나누어 개발하는 방식중 하나로 MVC를 명확히 구분하여 프로그램 하는것을 의미한다. m은 model로 사용자의 요청을 처리하는 비즈니스로직을 의미하고, v는 view로 사용자에게 데이터를 입력받거나 결과를 보여주는 화면을 의미하고 c는 컨트롤러로 사용자가 요청한

데이터를 판별하여 알맞은 데이터와 화면을 연결해 주는 작업을 한다. mvc패턴은 각각의 코드들을 명확하게 구분하여 프로그램을 하여 코드 이해와 유지 보수를 쉽게 할 수 있다. 전 페이지에서 작성한 model1를 업그레이드 하여 model2를 만들어 보자.

Model 2 MVC(Model-View-Controller)는 웹 애플리케이션을 개발하기 위한 디자인 패턴 중 하나입니다. Model 2 MVC 패턴은 JSP 페이지와 Servlet을 조합하여 사용하는 방식으로, 웹 애플리케이션의 비즈니스 로직과 프레젠테이션 로직을 분리하여 개발할 수 있게 합니다.

Model 2 MVC 패턴에서 각각의 역할은 다음과 같습니다.

Model: 데이터를 처리하고 비즈니스 로직을 수행하는 역할을 합니다. 데이터베이스나 사용자가 요청한 데이터 처리를 담당합니다. HumanController.java에서 사용자요청에 따른 DB작업이 이에 해당한다.

View: 웹 페이지를 표시하는 역할을 합니다. HTML, CSS, JavaScript 등으로 구성된 화면을 담당합니다. 사용자에게 보여줄 화면 .jsp가 뷰에 해당한다.

Controller: 클라이언트의 요청을 처리하고, Model과 View를 연결하여 전체적인 웹 애플리케이션의 흐름을 제어하는 역할을 합니다. HumanController.java가 컨트롤러에 해당한다.

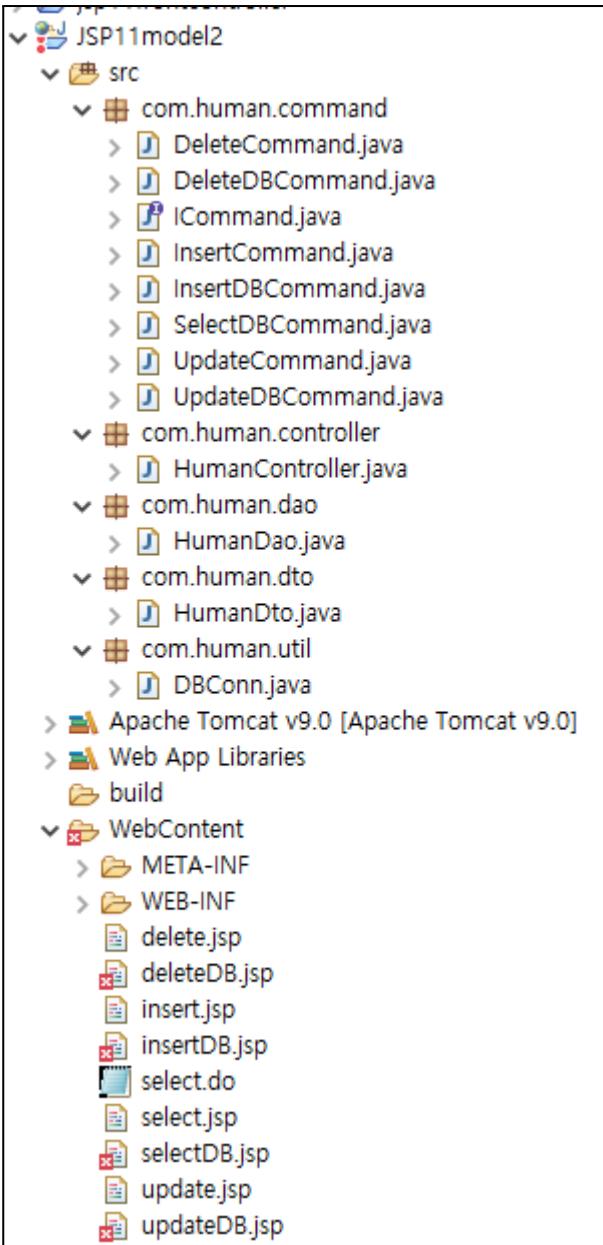
Model 2 MVC 패턴은 다음과 같은 특징을 갖습니다.

비즈니스 로직과 프레젠테이션 로직을 분리하여 개발할 수 있습니다.

유지보수와 확장성이 뛰어나며, 코드의 재사용성을 높일 수 있습니다.

Servlet과 JSP를 함께 사용하여 개발하기 때문에, 자바 기반의 웹 애플리케이션 개발에 적합합니다.

Model 2 MVC 패턴을 사용하면 웹 애플리케이션을 구조적으로 개발할 수 있으며, 코드의 가독성과 유지보수성을 높일 수 있습니다.



왼쪽 이미지에서 com.human.command 패키지 부분이 모델부분에 해당한다.

com.human.controller 패키지가 컨트롤러에 해당한다.

webcontent밑에 있는 .jsp파일들이 뷰에 해당한다.

select.do 파일은 프로젝트에서 필요없는 파일로 실행할 때 자동을 주소를 생성하려고 만든 파일이다. 프로젝트가 실행된후 실행 결과를 확인하려면 주소에 다음과 같이 입력해야 웹프로젝트의 시작 페이지인 select.jsp 뷰를 보여준다.

<http://localhost:8081/jsp11model2>

/select.do

저런 주소를 처리 하려면 프로젝트를 실행한 후 본인이 직접 주소를 입력 해야 하는데 select.do파일을 만들어 놓고 오른쪽 마우스 클릭 run as >> run on server를 선택하면 주소가 자동으로 상위와 같이 들어 간다.

따라서, 이프로젝트를 실행 하려면 select.do를 실행하면 된다.

모델1에서 모델 2로 변경하는 방법은 다음과 같다.

이전 jsp파일은 java 코드 와 html코드가 합쳐져 있다. java코드는 모델로 옮기고 html코드는 jsp파일에 남긴 다음 모델과 view를 컨트롤로 연결하여 사용자에게 보여 준다.

뷰(view) 부분은 .jsp로 만든 파일들이다.

모델(model) 부분은 인터페이스(ICommand.java)를 상속해서 xxxCommand.java 파일로 만들어 분리하여 데이터베이스 작업이나 비즈니스 로직과 같은 작업을 기술한다.

컨트롤러 부분은 HumanController.java파일에 해당한다.

1. 뷰(view) 부분은 기존 <% %> 스크립틀릿 으로 만들어진 부분을 모델로 옮기고 model에서 만들어진 데이터를 request에 담아서 jsp 뷰에서 jstl을 이용해서 보여 준다.

모델 1 프로젝트에서 해당 코드를 모델로 옮겨가고 jsp파일에서는 화면을 구성하는 코드는 jstl로 변경 하였다.

다음 예를 확인해 보자.

모델 1에서의 jsp코드 자바와 html이 섞여 있다.

```
<%--  
    HumanDao dao=new HumanDao();  
    ArrayList<HumanDto> dtos =  
        dao.select(request.getParameter("name"));  
    if(dtos!=null){  
        for(HumanDto dto:dtos){  
            out.println(dto); out.println("<br>");  
        }  
    }  
    System.out.println("end");  
  
--%>
```

view : 자바를 걷어내고 생성된 데이터를 request객체의 dtos에 담아서 처리 하는 형태로 jstl로 만들었다.

```
<table width="500" cellpadding="0" cellspacing="0" border="1">  
<tr><td>이름</td><td>나이</td><td>키</td><td>생일</td></tr>  
<c:forEach items="${requestScope.dtos }" var="dto">  
    <tr>  
        <td>${dto.name }</td><td>${dto.age }</td>  
        <td>${dto.tall }</td><td>${dto.birth }</td>  
    </tr>  
</c:forEach>  
</table>
```

model : 자바 코드를 모델에 인터페이스를 상속해서 request객체에 dtos 데이터를 담는 형태로 변경함

//SelectDBCommand.java

```
package com.human.command;  
  
import java.util.ArrayList;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import com.human.dao.HumanDao;  
import com.human.dto.HumanDto;  
  
public class SelectDBCommand implements ICommand {  
  
    @Override  
    public void execute(HttpServletRequest request, HttpServletResponse response) {
```

```

        HumanDao dao=new HumanDao();
        ArrayList<HumanDto> dtos =
            dao.select(request.getParameter("name"));

        request.setAttribute("dtos", dtos);
    }
}

controller : jsp파일에 자바 코드를 jstl로 완벽히 분리하면 나중에 분리한 view와
모델을 합치는 작업을 해야 한다. 사용자가 selectDB.do 와 같은 주소를 입력하면 입력한
주소에 맡는 viewPage="selectDB.jsp"; 뷰와 ic= new SelectDBCommand(); 모델을
선택해서 원하는 결과화면을 만들어 준다.

```

#### com.human.controller 파일

```

}else if(command.equals("/selectDB.do")){
    System.out.println("selectDB.do");
    viewPage="selectDB.jsp";
/* model1 코드에서는 주석된 코드를 사용 하였으나 model2에서는 주석된 해당 코드
대신에 ICommand.java 인터페이스를 상속받은 XXXCommand클래스 재정의된
execute메소드로 코드를 옮겨서 Model 코드 부분을 컨트롤러에서 완전히 제거 하였다.
    HumanDao dao=new HumanDao();
    ArrayList<HumanDto> dtos =
        dao.select(request.getParameter("name"));
    request.setAttribute("dtos", dtos);
*/
    ic= new SelectDBCommand();
    ic.execute(request,response);
}else if(command.equals("/delete.do")){

```

다음은 전체 코드이다.

```

//사용자 입력 view부분들 //insert.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.do'>입력</a>
<a href='update.do'>수정</a>
<a href='select.do'>검색</a>
<a href='delete.do'>삭제</a>

```

```

<form action="insertDB.do" method="get" >
    name:<input type="text" name="name"><br>
    age:<input type="text" name="age"><br>
    tall:<input type="text" name="tall"><br>
    birth:<input type="datetime-local" name="birth"><br>
    <input type="submit" value="제출"><br>
</form>
</body>
</html>

//select.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<!-- <script>
localhost.href="selectDB.jsp";
</script> -->
</head>
<body>
메뉴
<a href='insert.do'>입력</a><a href='update.do'>수정</a>
<a href='select.do'>검색</a><a href='delete.do'>삭제</a>
<form action="selectDB.do" method="get" >
    name:<input type="text" name="name"><br>
    <input type="submit" value="제출"><br>
</form>
</body>
</html>

//update.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.do'>입력</a><a href='update.do'>수정</a>
<a href='select.do'>검색</a><a href='delete.do'>삭제</a>

```

```

<form action="updateDB.do" method="get" >
    name:<input type="text" name="name"><br>
    age:<input type="text" name="age"><br>
    <input type="submit" value="제출"><br>
</form>
</body>
</html>

//delete.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.do'>입력</a><a href='update.do'>수정</a>
<a href='select.do'>검색</a><a href='delete.do'>삭제</a>
<form action="deleteDB.do" method="get" >
    name:<input type="text" name="name"><br>
    <input type="submit" value="제출"><br>
</form>
</body>
</html>

//사용자 요청 결과 view //insertDB.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ page import="com.human.dao.HumanDao" %>
<%@ page import="com.human.dto.HumanDto" %>
<%@ page import="java.util.ArrayList" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.do'>입력</a><a href='update.do'>수정</a>
<a href='select.do'>검색</a><a href='delete.do'>삭제</a>
<br>
검색데이터<br>

```

```

<table width="500" cellpadding="0" cellspacing="0" border="1">
<tr><td>이름</td><td>나이</td><td>키</td><td>생일</td></tr>
<tr><td>${dto.name }</td><td>${dto.age }</td><td>${dto.tall
}</td><td>${dto.birth }</td></tr>
</table>
<c:if test= "${i eq 1}" >      성공</c:if>
<c:if test= "${i ne 1}" >      실패</c:if>
</body>
</html>

//select.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<!-- <script>
localhost.href="selectDB.jsp";
</script> -->
</head>
<body>
메뉴
<a href='insert.do'>입력</a><a href='update.do'>수정</a>
<a href='select.do'>검색</a><a href='delete.do'>삭제</a>
<form action="selectDB.do" method="get" >
    name:<input type="text" name="name"><br>
    <input type="submit" value="제출"><br>
</form>
</body>
</html>

//selectDB.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ page import="com.human.dao.HumanDao" %>
<%@ page import="com.human.dto.HumanDto" %>
<%@ page import="java.util.ArrayList" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

```

```

메뉴
<a href='insert.do'>입력</a><a href='update.do'>수정</a>
<a href='select.do'>검색</a><a href='delete.do'>삭제</a>
<br>검색데이터<br>
<table width="500" cellpadding="0" cellspacing="0" border="1">
<tr><td>이름</td><td>나이</td><td>키</td><td>생일</td></tr>
<c:forEach items="${requestScope.dtos }" var="dto">
    <tr>
        <td>${dto.name }</td><td>${dto.age }</td>
        <td>${dto.tall }</td><td>${dto.birth }</td>
    </tr>
</c:forEach>
</table>
<%-- //모델 1에서 다음과 같이 주속된 방법을 사용 하였으나 해당 코드 일부는 모델로
옮겨가고 jsp파일에서는 화면을 구성하는 코드는 jstl로 변경하였다.
HumanDao dao=new HumanDao();
ArrayList<HumanDto> dtos =
    dao.select(request.getParameter("name"));
if(dtos!=null){
    for(HumanDto dto:dtos){
        out.println(dto); out.println("<br>");
    }
}
System.out.println("end");
--%>
</body>
</html>

//updateDB.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="com.human.dao.HumanDao" %>
<%@ page import="com.human.dto.HumanDto" %>
<%@ page import="java.util.ArrayList" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.do'>입력</a><a href='update.do'>수정</a>
<a href='select.do'>검색</a><a href='delete.do'>삭제</a>
<br>검색데이터<br>
<c:if test= "${result eq 1}" >      성공</c:if>

```

```

<c:if test= "${result ne 1}">      실패</c:if>
</body>
</html>

//deleteDB.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ page import="com.human.dao.HumanDao" %>
<%@ page import="com.human.dto.HumanDto" %>
<%@ page import="java.util.ArrayList" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
메뉴
<a href='insert.do'>입력</a><a href='update.do'>수정</a>
<a href='select.do'>검색</a><a href='delete.do'>삭제</a>
<br>검색데이터<br>
<c:if test= "${result eq 1}">      성공</c:if>
<c:if test= "${result ne 1}">      실패</c:if>
</body>
</html>

```

//모델관련 클래스 모든 모델이 동일한 형태로 동작하게 하기 위해서 ICommand 인터페이스를 상속받아서 execute메소드를 재정의 하여 필요한 기능을 구현하고 있다.

```

// ICommand.java
package com.human.command;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public interface ICommand {
    void execute(HttpServletRequest request,
                 HttpServletResponse response);
}

// DeleteCommand.java
package com.human.command;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class DeleteCommand implements ICommand {
    @Override
    public void execute(HttpServletRequest request, HttpServletResponse
response) {

```

```

        System.out.println("delete.do");
        //비지니스 작업
        //데이터베이스작업
    }
}

//DeleteDBCommand.java
package com.human.command;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class DeleteDBCommand implements ICommand {
    @Override
    public void execute(HttpServletRequest request, HttpServletResponse
response) {
        com.human.dao.HumanDao dao=new com.human.dao.HumanDao();
        int result=dao.delete(request.getParameter("name"));
        request.setAttribute("result",result);
        if(result==1){   out.println("성공"); }else{
        //          out.println("실패");
        //}
    }
}

//InsertCommand.java
package com.human.command;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class InsertCommand implements ICommand {
    @Override
    public void execute(HttpServletRequest request, HttpServletResponse
response) {
        System.out.println("insert.do");
    }
}

//InsertDBCommand.java
package com.human.command;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.human.dao.HumanDao;
import com.human.dto.HumanDto;
public class InsertDBCommand implements ICommand {
    @Override
    public void execute(HttpServletRequest request, HttpServletResponse
response) {
        HumanDto dto=new HumanDto();
        dto.setName(request.getParameter("name"));

```

```

        dto.setAge(Integer.parseInt(request.getParameter("age")));
        dto.setTall(Double.parseDouble(request.getParameter("tall")));
        // "yyyy-MM-dd'T'hh:mm"
        dto.setBirthString(request.getParameter("birth"));
        HumanDao dao =new HumanDao();
        int i=dao.insert(dto);

        request.setAttribute("dto", dto);
        request.setAttribute("i", i);
    }
}

//SelectDBCommand.java
package com.human.command;
import java.util.ArrayList;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.human.dao.HumanDao;
import com.human.dto.HumanDto;
public class SelectDBCommand implements ICommand {
    @Override
    public void execute(HttpServletRequest request,HttpServletResponse response){
        HumanDao dao=new HumanDao();
        ArrayList<HumanDto> dtos =
            dao.select(request.getParameter("name"));
        request.setAttribute("dtos", dtos);
    }
}

//UpdateCommand.java
package com.human.command;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class UpdateCommand implements ICommand {
    @Override
    public void execute(HttpServletRequest request, HttpServletResponse response) {
        System.out.println("update.do");
    }
}

//UpdateDBCommand.java
package com.human.command;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class UpdateDBCommand implements ICommand {
    @Override

```

```

public void execute(HttpServletRequest request, HttpServletResponse response){
    com.human.dao.HumanDao dao=new com.human.dao.HumanDao();
    int result=dao.update(request.getParameter("name"),
                          request.getParameter("age"));
    request.setAttribute("result",result);
    //      if(result==1){    out.println("성공"); }else{
    //          out.println("실패");
    //      }
}
}

//컨트롤러 HumanController.java
package com.human.controller;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.human.command.ArrayList;
import com.human.command.DeleteCommand;
import com.human.command.DeleteDBCommand;
import com.human.command.ICommand;
import com.human.command.InsertCommand;
import com.human.command.InsertDBCommand;
import com.human.command.SelectDBCommand;
import com.human.command.UpdateCommand;
import com.human.command.UpdateDBCommand;
import com.human.dao.HumanDao;
import com.human.dto.HumanDto;
@WebServlet("*.do")
public class HumanController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public HumanController() { super(); }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String uri=request.getRequestURI();
        System.out.println(uri);///jsp10frontController/insert.do
        String conPath=request.getContextPath();///jsp10frontController
        String command=uri.substring(conPath.length());
        String viewPage="view.jsp";

        ICommand ic=null;
        if(command.equals("/insert.do")){
            System.out.println("insert.do");
            viewPage="insert.jsp";
        }
    }
}

```

```

}else if(command.equals("/insertDB.do")){
    System.out.println("select.do");
    viewPage="insertDB.jsp";
    ic=new InsertDBCommand();
    ic.execute(request, response);
}else if(command.equals("/select.do")){
    System.out.println("select.do");
    viewPage="select.jsp";
}else if(command.equals("/selectDB.do")){
    System.out.println("selectDB.do");
    viewPage="selectDB.jsp";
    /* model1 코드에서는 주석된 코드를 사용 하였으나 model2에서는
주석된 해당 코드 대신에 ICommand.java 인터페이스를 상속받은 XXXCommand클래스
재정의된 execute메소드로 코드를 옮겨서 Model 코드 부분을 컨트롤러에서 완전히 제거
하였다.
HumanDao dao=new HumanDao();
ArrayList<HumanDto> dtos =
        dao.select(request.getParameter("name"));
request.setAttribute("dtos", dtos);
*/
    ic= new SelectDBCommand();
    ic.execute(request,response);
}else if(command.equals("/delete.do")){
    viewPage="delete.jsp";
}else if(command.equals("/deleteDB.do")){
    viewPage="deleteDB.jsp";
    ic= new DeleteDBCommand();
    ic.execute(request,response);
}else if(command.equals("/update.do")){
    viewPage="update.jsp";
}else if(command.equals("/updateDB.do")){
    viewPage="updateDB.jsp";
    ic= new UpdateDBCommand();
    ic.execute(request,response);
}
RequestDispatcher dispatcher=
        request.getRequestDispatcher(viewPage);
dispatcher.forward(request, response);
}

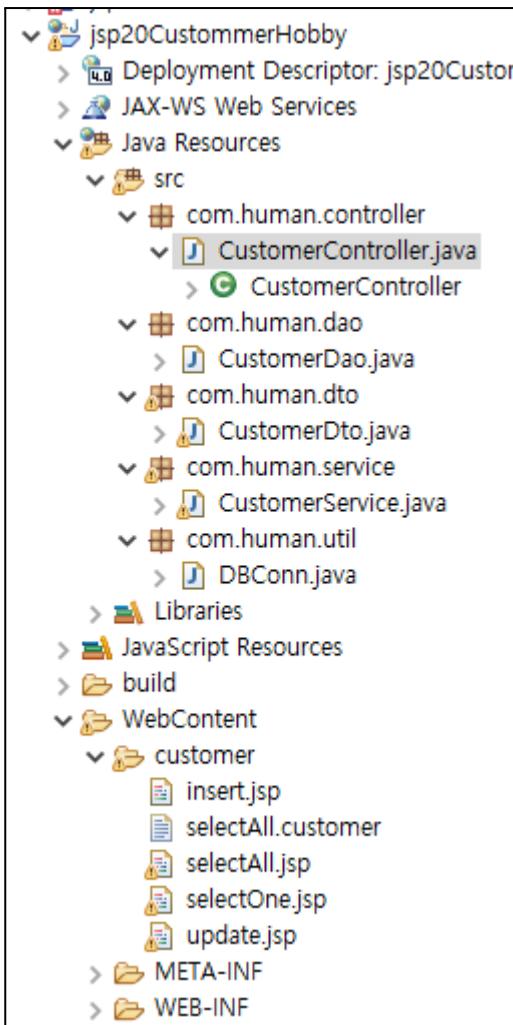
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}

다음 코드는 모델 1과 동일함 //HumanDao.java//HumanDto.java//DBConn.java
다음을 구현해 보자.

```

1. session login를 db를 이용해서 model2로 구현해 보자.

## > 17. CustomHobby-Custom만들기



다음은 customerHobby 테이블을 조작하는 프로그램을 만들어 보자. 두 테이블은 1:N의 관계를 가진다. 하나의 고객은 여러개의 하비를 가지고 하나의 하비는 하나의 고객을 가진다.

다음과 같은 순으로 Custom 관리 프로그램 만들기를 만들어 보자.

```
//데이터 베이스 만들기  
//dto만들기 com.human.dto.CustomerDto.java  
//dao만들때 필요한 DBconn.java 생성  
//dao만들기 com.human.dao.CustomerDao.java  
//service만들기  
com.human.dao.CustomerService.java  
//Custom폴더에 insert.jsp  
/custom/insert.customer  
//com.human.controller.CustomerController.java  
*.customer  
//selectAll.jsp      /custom/selectAll.customer  
//selectOne.jsp     /custom/selectOne.customer  
//update.jsp        /custom/update.customer  
//update.jsp에서의 사용자 입력 처리하는  
controller등록 /custom/updateDB.customer
```

### Custom 관리프로그램 만들기

```
//데이터 베이스 만들기  
--c##human / human  
DROP TABLE customer CASCADE CONSTRAINTS;  
create table customer(  
    id number,  
    name nvarchar2(30),  
    height number(5,2),  
    birthday date  
);  
  
drop table hobby;  
create table hobby(  
    id number,  
    hobby nvarchar2(100)  
);
```

```

drop sequence id_counter;
create sequence id_counter;

//dto만들기 com.human.dto.CustomerDto.java
package com.human.dto;

import java.time.LocalDateTime;
import java.util.Objects;

//dto에 추가해야 할것들
//1. getter,setter
//2. 생성자
//3. equals hashCode
//4. toString
public class CustomerDto {
    private Integer id;
    private String name;
    private Double height;
    private LocalDateTime birthday;

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((id == null) ? 0 : id.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        CustomerDto other = (CustomerDto) obj;
        if (id == null) {
            if (other.id != null)
                return false;
        } else if (!id.equals(other.id))
            return false;
        return true;
    }
    public CustomerDto() {}
    public CustomerDto(Integer id, String name, Double height, LocalDateTime
birthday) {
        super();

```

```

        this.id = id;
        this.name = name;
        this.height = height;
        this.birthday = birthday;
    }
    @Override
    public String toString() {
        return "HumanDto [id=" + id + ", name=" + name + ", height=" +
height + ", birthday=" + birthday + "]";
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Double getHeight() {
        return height;
    }
    public void setHeight(Double height) {
        this.height = height;
    }
    public LocalDateTime getBirthday() {
        return birthday;
    }
    public void setBirthday(LocalDateTime birthday) {
        this.birthday = birthday;
    }
}

```

//dao만들때 필요한 DBconn.java 생성  
package com.human.util;

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
//ctl+a 전체 선택
//ctl+shift +f 코드 정리
public class DBConn {
    private DBConn() {

```

```

}

private static Connection dbConn = null;
private static Statement st = null;
private static ResultSet rs = null;

public static Connection getInstance() {
    if (dbConn == null) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            String url = "jdbc:oracle:thin:@localhost:1521:xe";
            String id = "c##human";
            String pw = "human";
            dbConn = DriverManager.getConnection(url, id, pw);
            System.out.println("DBConnection....");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return dbConn;
}

public static void dbClose() {
    try {
        if (rs != null)
            rs.close();
        if (st != null)
            st.close();
        if (dbConn != null)
            dbConn.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        rs = null;
        st = null;
        dbConn = null;
    }
}

public static int statementUpdate(String sql) {
    DBConn.getInstance();
    int rValue = -1;
    if (dbConn != null) {
        try {
            if (st == null) {
                st = dbConn.createStatement();
            }
        }

```

```

//insert,delete,update
                rValue = st.executeUpdate(sql);

            } catch (SQLException e) {
                e.printStackTrace();
            }
        } else {
            System.out.println("not connected...");
        }

        return rValue;
    }

    public static ResultSet statementQuery(String sql) {
        DBConn.getInstance();
        if (DBConn.dbConn != null) {
            try {
                if (st == null) {
                    st = dbConn.createStatement();
                }
//insert,delete,update
                rs = st.executeQuery(sql);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        } else {
            System.out.println("not connected...");
        }

        return rs;
    }
}

```

```

//dao만들기 com.human.dao.CustomerDao.java
package com.human.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;

import com.human.dto.CustomerDto;
import com.human.dto.CustomerHobbyDto;
import com.human.dto.HobbyDto;
import com.human.util.DBConn;

```

```

//ctr+a 모든 코드 선택 ctr+shift+f 코드정리
public class CustomerDao {
    // 1. 디비를 이용해서 데이터 관리를 하므로 더이상 필요없음
    // private ArrayList<HumanDto> dtos = new ArrayList<HumanDto>();

    // 2. 데이터 초기화 init
    public void init() {
        // 1번데이터
        CustomerDto dto = new CustomerDto(null, "홍길동1", 152.1,
                                         LocalDateTime.parse("2000-02-03 00:00:00"
                                         , DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss")));
        String sql = String.format(
                    "insert into customer
values(id_counter.nextval,'%s',%f,to_date('%s','YYYY:MM:DD HH24:MI:SS'))",
                    dto.getName(), dto.getHeight(),

        dto.getBirthDay().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")));
        DBConn.createStatementUpdate(sql);
        // 2번데이터
        dto = new CustomerDto(null, "홍길동2", 152.1,
                             LocalDateTime.parse("2000-02-03 00:00:00"
                             , DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss")));
        // to_date('%s', 'YYYY:MM:DD HH24:MI:SS')
        sql = String.format(
                    "insert into customer
values(id_counter.nextval,'%s',%f,to_date('%s','YYYY:MM:DD HH24:MI:SS'))",
                    dto.getName(), dto.getHeight(),

        dto.getBirthDay().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")));
        DBConn.createStatementUpdate(sql);
        // 3번데이터
        dto = new CustomerDto(null, "홍길동3", 152.1,
                             LocalDateTime.parse("2000-02-03 00:00:00",
                                         DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")));
        sql = String.format("insert into customer
values(id_counter.nextval,'%s',%f,to_date('%s','YYYY:MM:DD HH24:MI:SS'))",
                    dto.getName(), dto.getHeight(),

        dto.getBirthDay().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")));
        DBConn.createStatementUpdate(sql);
    }
    //crud작업
    //insert작업
    public void insert(CustomerDto dto) {
        String sql = String.format(

```

```

        "insert into customer
values(id_counter.nextval,'%s',%f,to_date('%s','YYYY:MM:DD HH24:MI:SS'))",
                dto.getName(), dto.getHeight(),

dto.getBirthday().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")));
        DBCConn.createStatement(sql);
    }
// select 작업
public ArrayList<CustomerDto> selectAll() {
    ArrayList<CustomerDto> resultDtos = new ArrayList<CustomerDto>();
ResultSet rs = DBCConn.createStatementQuery(String.format("select * from customer"));
    try {
        while (rs.next()) {
            resultDtos.add(new CustomerDto(rs.getInt("id"),
rs.getString("name"),rs.getDouble("height"),
rs.getTimestamp("birthday").toLocalDateTime()));
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return resultDtos;
}
// select 작업
public CustomerDto selectId(int id) {
    CustomerDto resultDtos = new CustomerDto();
    ResultSet rs = DBCConn.createStatementQuery(String.format("select * from
customer where id=%d",id));
    if(rs!=null) {
        try {
            rs.next();
            resultDtos=new CustomerDto(rs.getInt("id"),
rs.getString("name"), rs.getDouble("height"),
rs.getTimestamp("birthday").toLocalDateTime());
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
    return resultDtos;
}

// update작업
public void update(int id, String name) {
    // update 이름을 이용해서 나이를 변경하는 형태

```

```

        DBConn.statementUpdate(String.format("update customer set name='%s'
where id=%d", name,id));
    }

// delete작업
public void delete(int id) {
    DBConn.statementUpdate(String.format("delete customer where id =
%d", id));
}
public ArrayList<Integer> getIds() {
    ArrayList<Integer> ids=new ArrayList<Integer>();

    ResultSet rs = DBConn.createStatementQuery(String.format("select * from
customer"));
    try {
        while (rs.next()) {
            ids.add(rs.getInt("id"));
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return ids;
}

//CustomerHobby 작업으로 생긴 부분
public int getMaxId() {
    int maxIdValue=-1;

    ResultSet rs = DBConn.createStatementQuery(String.format("select max(id)
as maxId from customer"));
    if(rs!=null) {
        try {
            rs.next();
            maxIdValue=rs.getInt("maxId");
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
    return maxIdValue;
}

public void update(int id, double height) {
    DBConn.statementUpdate(String.format("update customer set height=%f
where id=%d", height,id));
}

```

```

    }
}

//service만들기 com.human.dao.CustomerService.java
package com.human.service;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.human.dao.CustomerDao;
import com.human.dto.CustomerDto;

public class CustomerService {
    public static CustomerDao customerDao=new CustomerDao();

    public static void insertDB(HttpServletRequest request,
HttpServletResponse response) {
        CustomerDto dto=new CustomerDto(null,
                request.getParameter("name"),
                Double.parseDouble(request.getParameter("height")),
                //yyyy-MM-dd'T'HH:mm
                LocalDateTime.parse(request.getParameter("birthday")))

        );
        System.out.println(dto);
        customerDao.insert(dto);
    }

    public static void selectAll(HttpServletRequest request,
HttpServletResponse response) {
        ArrayList<CustomerDto> dtos =customerDao.selectAll();
        request.setAttribute("dtos",dtos);
    }

    public static void selectOne(HttpServletRequest request,
HttpServletResponse response) {
        CustomerDto dto
=customerDao.selectId(Integer.parseInt(request.getParameter("id")));

        request.setAttribute("dto", dto);
    }
}

```

```

    public static void delete(HttpServletRequest request, HttpServletResponse response) {
        // TODO Auto-generated method stub
        customerDao.delete(Integer.parseInt(request.getParameter("id")));
    }

    public static void update(HttpServletRequest request, HttpServletResponse response) {
        customerDao.update(Integer.parseInt(request.getParameter("id"))
            , request.getParameter("name"));
    }

}

```

```

//Custom 폴더에 insert.jsp      /custom/insert.customer
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <form action="/jsp20CustomerHobby/customer/insertDB.customer"
method="get">
        이름 :<input type=text name=name ><br>
        키:<input type=text name=height ><br>
        생일:<input type="datetime-local" name=birthday ><br>
        <input type="submit" value=전송 >
    </form>

</body>
</html>

//com.human.controller.CustomController.java *.customer
package com.human.controller;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;

```

```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.human.service.CustomerService;

/**
 * Servlet implementation class CustomerController
 */
@WebServlet("*.customer")
public class CustomerController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public CustomerController() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // URI:/jsp10/hello.do
        // conPath:/jsp10
        // command:/hello.do
        String uri = request.getRequestURI();
        System.out.println("URI:" + uri);
        // 원하는 수소에 대한 처리 방법
        String conPath = request.getContextPath();
        System.out.println("conPath:" + conPath);

        String command = uri.substring(conPath.length());
        System.out.println("command:" + command);

        String viewPage = "selectAll.jsp";

//http://localhost:8081/jsp20CustommerHobby/customer/insert.customer
        if (command.equals("/customer/insert.customer")) {
            viewPage = "insert.jsp";
    }

    System.out.println("/customer/insert.customer");
}

```

```

        } else if
(command.equals("/customer/insertDB.customer")) {
                viewPage = "selectAll.customer";
CustomerService.insertDB(request, response);

System.out.println("/customer/insertDB.customer");
        } else if
(command.equals("/customer/selectAll.customer")) {
                viewPage = "selectAll.jsp";
CustomerService.selectAll(request, response);

System.out.println("/customer/selectAll.customer");
        }else
if(command.equals("/customer/selectOne.customer")) {
                viewPage = "selectOne.jsp";
CustomerService.selectOne(request, response);

System.out.println("/customer/selectOne.customer");
        }else if(command.equals("/customer/delete.customer")) {
                viewPage ="selectAll.customer";
CustomerService.delete(request,response);
System.out.println("/customer/delete.customer");
        }else if(command.equals("/customer/update.customer")) {
                viewPage = "update.jsp";
CustomerService.selectOne(request, response);
System.out.println("/customer/update.customer");
        }else if(command.equals("/customer/updateDB.customer"))
{
                viewPage ="selectAll.customer";
CustomerService.update(request, response);

System.out.println("/customer/updateDB.customer");
        }

        System.out.println(viewPage);
RequestDispatcher dispatcher =
request.getRequestDispatcher(viewPage);
dispatcher.forward(request, response);
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);

```

```

}

}

//selectAll.jsp /custom/selectAll.customer
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>고객 정보</h1>
<%
    java.time.LocalDateTime ld=java.time.LocalDateTime.now();
    request.setAttribute("ld", ld);
%>
<hr>
현재시간
<hr>
-${requestScope.ld }-
<fmt:parseDate value="${requestScope.ld }"
   pattern="yyyy-MM-dd'T'HH:mm:ss.SSS"
   var="now" type="both" ></fmt:parseDate>
<hr>
<fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일 hh시 mm분
ss초"></fmt:formatDate>
<hr>
<table border="1" width="90%" id="customers">
    <tr>
        <th>id</th>
        <th>name</th>
        <th>height</th>
        <th>birthday</th>
        <th>birthday</th>
    </tr>
    <c:forEach items="${dtos }" var="customerDto">
        <fmt:parseDate value="${customerDto.birthday}"
           pattern="yyyy-MM-dd'T'HH:mm"
           var="now" type="both" ></fmt:parseDate>
        <tr>
            <td>${customerDto.id }</td>
            <td><a
href='/jsp20CustomerHobby/customer/selectOne.customer?id=${customerDto.id}'>

```

```

        ${customerDto.name }</a></td>
        <td>${customerDto.height}</td>
        <td><fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일
hh시 mm분 ss초"/></td>
        <td>${customerDto.birthday}</td>
    </tr>
</c:forEach>
</table>
<a href="/jsp20CustommerHobby/customer/insert.customer">고객추가</a>
</body>
</html>

```

```

//selectOne.jsp    /custom/selectOne.customer
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<table border="1" width="90%" id="customers">
    <tr>
        <th>id</th>
        <th>name</th>
        <th>height</th>
        <th>birthday</th>
    </tr>
    <fmt:parseDate value="${requestScope.dto.birthday}"
       pattern="yyyy-MM-dd'T'HH:mm"
       var="now" type="both" ></fmt:parseDate>
    <tr>

        <td>${requestScope.dto.id }</td>
        <td>${requestScope.dto.name }</td>
        <td>${requestScope.dto.height}</td>
        <td><fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일 hh시
mm분 ss초"/></td>
    </tr>
</table>
<a href="/jsp20CustommerHobby/customer/selectAll.customer">목록가기</a>
<a href="/jsp20CustommerHobby/customer/update.customer?id=${requestScope.dto.id
}">수정하기</a>

```

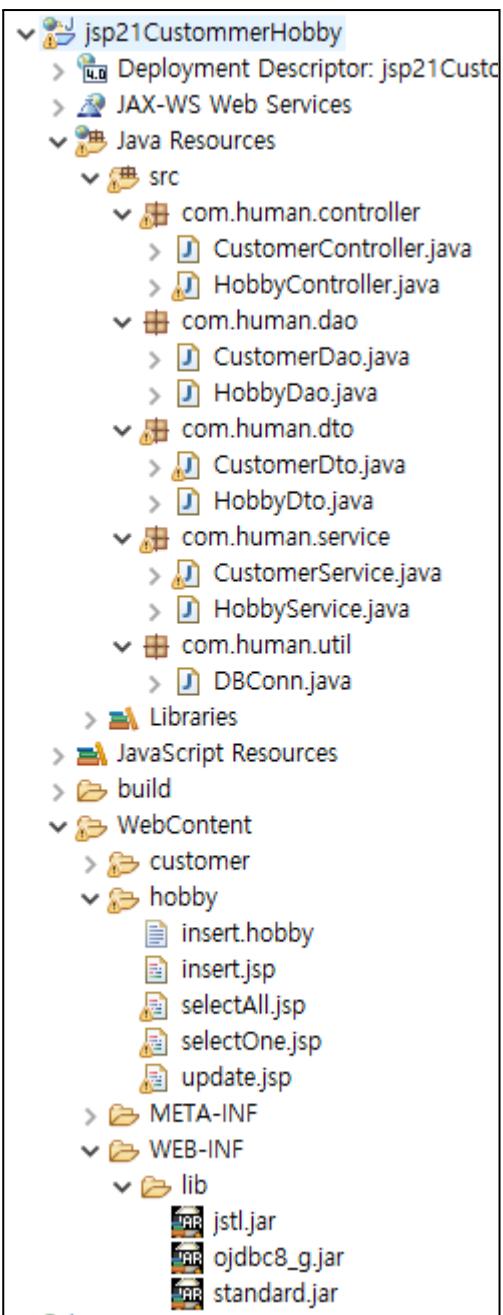
```

<a href="/jsp20CustommerHobby/customer/delete.customer?id=${requestScope.dto.id }">삭제하기</a>
</body>
</html>

//update.jsp      /custom/update.customer
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action='/jsp20CustommerHobby/customer/updateDB.customer' method="get">
<table border="1" width="90%" id="customers">
    <tr>
        <th>id</th>
        <th>name</th>
        <th>height</th>
        <th>birthday</th>
    </tr>
    <fmt:parseDate value="${requestScope.dto.birthday}"
       pattern="yyyy-MM-dd'T'HH:mm"
       var="now" type="both" ></fmt:parseDate>
    <tr>
        <input type='hidden' name="id" value='${requestScope.dto.id }'>
        <td>${requestScope.dto.id }</td>
        <td><input type='text' name="name" value='${requestScope.dto.name
}'></td>
        <td>${requestScope.dto.height}</td>
        <td><fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일 hh시
mm분 ss초"/></td>
    </tr>
</table>
<a href="/jsp20CustommerHobby/customer/selectAll.customer">목록가기</a>
<input type="submit" value='수정'>
</form>
</body>
</html>

```

## > 18. CustomHobby-Hobby만들기



왼쪽이미지는 이전 챕터에서 제작한 프로젝트를 복사해서 hobby관련 프로그램을 추가한 프로젝트 이미지이다.

hobby 관리프로그램 만들기

```
//dto만들기 com.human.dto.HobbyDto.java  
//dao만들기 com.human.dao.HobbyDao.java  
//service만들기 com.human.dao.HobbyService.java  
//hobby폴더에 insert.jsp  
/hobby/insert.hobby  
//com.human.controller.HobbyController.java  
*.hobby  
//selectAll.jsp/hobby/selectAll.hobby  
//selectOne.jsp/hobby/selectOne.hobby  
//update.jsp /hobby/update.hobby
```

```
//dto만들기 com.human.dto.HobbyDto.java  
package com.human.dto;
```

```
public class HobbyDto {  
    private Integer id;  
    private String hobby;  
  
    public HobbyDto() {}  
    public HobbyDto(Integer id, String hobby) {  
        super();  
        this.id = id;  
        this.hobby = hobby;  
    }  
    @Override  
    public int hashCode() {  
        final int prime = 31;  
  
        int result = 1;  
        result = prime * result + ((hobby == null) ? 0 : hobby.hashCode());  
        result = prime * result + ((id == null) ? 0 : id.hashCode());  
        return result;  
    }  
    @Override  
    public boolean equals(Object obj) {  
        if (this == obj)  
            return true;
```

```

        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        HobbyDto other = (HobbyDto) obj;
        if (hobby == null) {
            if (other.hobby != null)
                return false;
        } else if (!hobby.equals(other.hobby))
            return false;
        if (id == null) {
            if (other.id != null)
                return false;
        } else if (!id.equals(other.id))
            return false;
        return true;
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getHobby() {
        return hobby;
    }
    public void setHobby(String hobby) {
        this.hobby = hobby;
    }
    @Override
    public String toString() {
        return "HobbyDto [id=" + id + ", hobby=" + hobby + "]";
    }
}

//dao만들기 com.human.dao.HobbyDao.java
package com.human.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import com.human.dto.HobbyDto;
import com.human.util.DBConn;

```

```

public class HobbyDao {
    //insert
    public void insert(HobbyDto dto) {
        String sql = String.format(
            "insert into hobby values(%d,'%s')",
            dto.getId(), dto.getHobby());
        DBConn.createStatementUpdate(sql);
    }
    //select
    public ArrayList<HobbyDto> selectAll(){
        ArrayList<HobbyDto> resultDtos = new ArrayList<HobbyDto>();

        ResultSet rs = DBConn.createStatementQuery(String.format("select * from hobby order by id"));

        try {
            while(rs.next()) {
                resultDtos.add(new HobbyDto(rs.getInt("id"),rs.getString("hobby")));
            }
        }catch(SQLException e) {
            e.printStackTrace();
        }
        return resultDtos;
    }
    //select id
    public HobbyDto selectOne(int id,String hobby){
        HobbyDto resultDto = new HobbyDto();
        ArrayList<HobbyDto> resultDtos = new ArrayList<HobbyDto>();

        ResultSet rs = DBConn.createStatementQuery(String.format("select * from hobby where id=%d and hobby='%s'",id,hobby));
        if(rs!=null) {
            try {
                rs.next();
                resultDto = new HobbyDto(rs.getInt("id"),rs.getString("hobby"));

            }catch(SQLException e) {
                e.printStackTrace();
            }catch(Exception e) {
                e.printStackTrace();
            }
        }
        return resultDto;
    }
    //update
    public void update(int id,String beforehobby,String afterhobby) {

```

```

        DBConn.createStatement().executeUpdate(String.format("update hobby set
hobby=%s' where id=%d and hobby=%s'", afterhobby, id, beforehobby));
    }
    //delete
    public void delete(int id, String hobby) {
        DBConn.createStatement().executeUpdate(String.format("delete hobby where
id=%d and hobby=%s'", id, hobby));
    }
}

//service 만들기 com.human.dao.HobbyService.java
package com.human.service;

import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.human.dao.CustomerDao;
import com.human.dao.HobbyDao;
import com.human.dto.HobbyDto;

public class HobbyService {

    public static HobbyDao hobbyDao = new HobbyDao();
    public static CustomerDao customerDao = new CustomerDao();

    public static void insertDB(HttpServletRequest request,
HttpServletResponse response) {
        HobbyDto dto = new
HobbyDto(Integer.parseInt(request.getParameter("id")),
            request.getParameter("hobby"));
        hobbyDao.insert(dto);
    }

    public static void selectAll(HttpServletRequest request,
HttpServletResponse response) {
        ArrayList<HobbyDto> dtos = hobbyDao.selectAll();
        request.setAttribute("dtos", dtos);
    }

    public static void selectOne(HttpServletRequest request,
HttpServletResponse response) {
        HobbyDto dto
= hobbyDao.selectOne(Integer.parseInt(request.getParameter("id")),
            request.getParameter("hobby"));
        request.setAttribute("dto", dto);
    }
}

```

```

    }

    public static void delete(HttpServletRequest request, HttpServletResponse response) {
        hobbyDao.delete(Integer.parseInt(request.getParameter("id")),request.getParameter("hobby"));
    }

    public static void update(HttpServletRequest request, HttpServletResponse response) {
        hobbyDao.update(Integer.parseInt(request.getParameter("id")),request.getParameter("beforehobby"),request.getParameter("afterhobby"));
    }

    public static void insert(HttpServletRequest request, HttpServletResponse response) {
        ArrayList<Integer> ids=customerDao.getIds();

        request.setAttribute("ids", ids);
    }

}

//hobby 풀더에 insert.jsp      /hobby/insert.hobby
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>취미 추가 페이지</h1>
<form action="/jsp19CustommerHobby/hobby/insertDB.hobby" method="get">
    ID :
    <select name="id">
        <c:forEach items="${ids }" var="id">

```

```

                <option value="${id }">${id }
            </c:forEach>
        </select>
        <br>
        취미 : <input type='text' name='hobby'><br>
        <input type='submit' value=전송>
    </form>
</body>
</html>

//com.human.controller.HobbyController.java *.hobby
package com.human.controller;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.human.service.HobbyService;
import com.human.util.DBConn;

/**
 * Servlet implementation class HobbyController
 */
@WebServlet("*.hobby")
public class HobbyController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public HobbyController() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String uri = request.getRequestURI();
        String conPath = request.getContextPath();

```

```

String command = uri.substring(conPath.length()));

String viewPage = "selectAll.jsp";
if (command.equals("/hobby/insert.hobby")) {
    viewPage = "insert.jsp";
    HobbyService.insert(request, response);

    System.out.println("insert.hobby");

} else if (command.equals("/hobby/insertDB.hobby")) {
    viewPage = "selectAll.hobby";
    HobbyService.insertDB(request, response);
    System.out.println("insertDB.hobby");

} else if (command.equals("/hobby/selectAll.hobby")) {
    viewPage = "selectAll.jsp";
    HobbyService.selectAll(request, response);
    System.out.println("select.hobby");

} else if (command.equals("/hobby/selectOne.hobby")) {
    viewPage = "selectOne.jsp";
    HobbyService.selectOne(request, response);
    System.out.println("select.hobby");

} else if (command.equals("/hobby/delete.hobby")) {
    viewPage = "selectAll.hobby";
    HobbyService.delete(request, response);
    System.out.println("delete.hobby");
} else if (command.equals("/hobby/update.hobby")) {
    viewPage = "update.jsp";
    HobbyService.selectOne(request, response);
    System.out.println("update.hobby");
} else if (command.equals("/hobby/updateDB.hobby")) {
    viewPage = "selectAll.hobby";
    HobbyService.update(request, response);
    System.out.println("updateDB.hobby");
}

System.out.println(viewPage);
RequestDispatcher dispatcher =
request.getRequestDispatcher(viewPage);
dispatcher.forward(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}

```

```

//selectAll.jsp    /hobby/selectAll.hobby
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>HOBBY 정보</h1>
  <table border="1" width="40%" id="hobbys">
    <tr>
      <th>ID</th>
      <th>취미</th>
    </tr>
    <c:forEach items="${dtos }" var="hobbyDto">
      <tr>
        <td>${hobbyDto.id }</td>
        <td><a href='/jsp19CustommerHobby/hobby/selectOne.hobby?id=${hobbyDto.id}&hobby=${hobbyDto.hobby}'>
          ${hobbyDto.hobby }</a></td>
      </tr>
    </c:forEach>
  </table>
  <a href='/jsp19CustommerHobby/hobby/insert.hobby'>추가</a>
</body>
</html>

//selectOne.jsp    /hobby/selectOne.hobby
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>HOBBY 정보</h1>
  <table border="1" width="40%" id="hobbys">
    <tr>
      <th>ID</th>
      <th>취미</th>
    </tr>

```

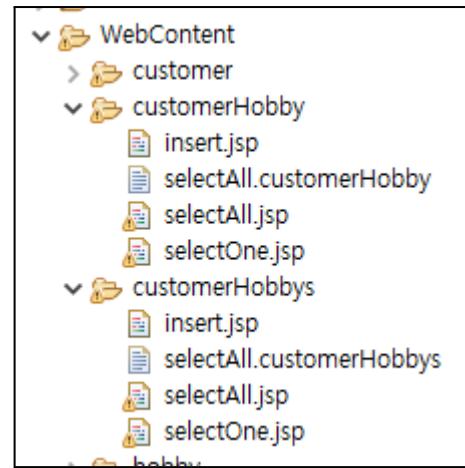
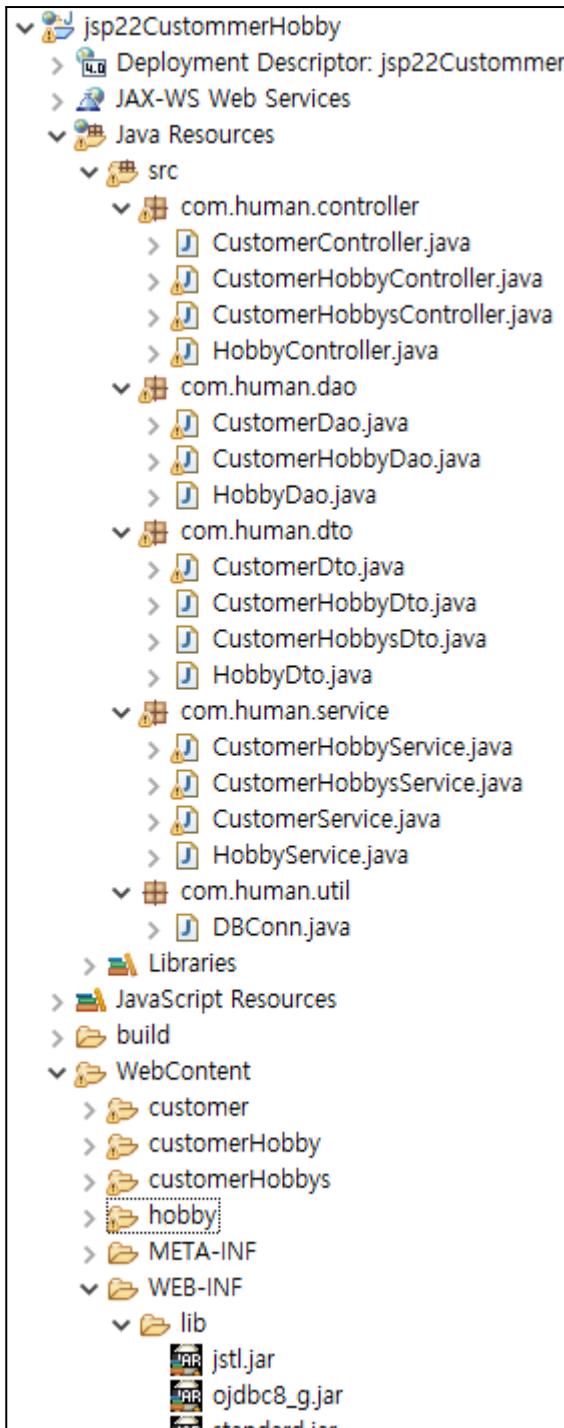
```

<tr>
    <td>${requestScope.dto.id }</td>
    <td>${requestScope.dto.hobby }</td>
</tr>
</table>
<a href="/jsp19CustommerHobby/hobby/selectAll.hobby">목록으로</a>
<a
href="/jsp19CustommerHobby/hobby/update.hobby?id=${requestScope.dto.id}&hobby=$
{requestScope.dto.hobby}">수정</a>
<a
href="/jsp19CustommerHobby/hobby/delete.hobby?id=${requestScope.dto.id}&hobby=$
{requestScope.dto.hobby}">삭제</a>
</body>
</html>

//update.jsp      /hobby/update.hobby
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action="/jsp19CustommerHobby/hobby/updateDB.hobby" method="get">
<h1>고객 정보</h1>
    <table border="1" width="90%" id="hobbys">
        <tr>
            <th>ID</th>
            <th>취미</th>
        </tr>
        <tr>
            <input type="hidden" value="${requestScope.dto.id }" name='id'>
                <td>${requestScope.dto.id }</td>
            <input type="hidden" value="${requestScope.dto.hobby}" name='beforehobby'>
                <td><input type="text" value="${requestScope.dto.hobby}">
name='afterhobby'></td>
            </tr>
        </table>
        <a href="/jsp19CustommerHobby/hobby/selectAll.hobby">목록으로</a>
        <input type="submit" value='수정'>
    </form>
</body>
</html>

```

## > 19. CustomHobby-CustomHobby join 만들기



상위 이미지는 다음 차트에서 제작할 CustomerHobbies를 포함한 이미지이다. 다음 챕터에서 사용할 때도 참고하자.

```
//CustomerHobbyDto.java  
//CustomerHobbyDao.java  
//CustomerHoobyService.java  
//CustomerHoobyController.java  
//customerHobby/insert.jsp  
//customerHobby/selectAll.jsp  
//customerHobby/selectOne.jsp  
  
//CustomerHobbyDto.java  
package com.human.dto;  
  
import java.time.LocalDateTime;  
  
public class CustomerHobbyDto {  
    private int id;  
    private String name;  
    private double height;  
    private LocalDateTime birthday;  
  
    private HobbyDto hobby;  
    public CustomerHobbyDto() {}  
  
    public CustomerHobbyDto(int id, String name, double height,  
    LocalDateTime birthday, HobbyDto hobby) {  
        super();  
        this.id = id;  
        this.name = name;  
        this.height = height;
```

```

        this.birthday = birthday;
        this.hobby = hobby;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((birthday == null) ? 0 :
birthday.hashCode());
        long temp;
        temp = Double.doubleToLongBits(height);
        result = prime * result + (int) (temp ^ (temp >>> 32));
        result = prime * result + ((hobby == null) ? 0 :
hobby.hashCode());
        result = prime * result + id;
        result = prime * result + ((name == null) ? 0 :
name.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        CustomerHobbyDto other = (CustomerHobbyDto) obj;
        if (birthday == null) {
            if (other.birthday != null)
                return false;
        } else if (!birthday.equals(other.birthday))
            return false;
        if (Double.doubleToLongBits(height) !=
Double.doubleToLongBits(other.height))
            return false;
        if (hobby == null) {
            if (other.hobby != null)
                return false;
        } else if (!hobby.equals(other.hobby))
            return false;
        if (id != other.id)
            return false;
        if (name == null) {
            if (other.name != null)
                return false;

```

```

        } else if (!name.equals(other.name))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "CustomerHobbyDto [id=" + id + ", name=" + name + ",
height=" + height + ", birthday=" + birthday
                + ", hobby=" + hobby + "]";
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }

    public LocalDateTime getBirthday() {
        return birthday;
    }

    public void setBirthday(LocalDateTime birthday) {
        this.birthday = birthday;
    }

    public HobbyDto getHobby() {
        return hobby;
    }
}

```

```

        public void setHobby(HobbyDto hobby) {
            this.hobby = hobby;
        }

    }

//CustomerHobbyDao.java
package com.human.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;

import com.human.dto.CustomerDto;
import com.human.dto.CustomerHobbyDto;
import com.human.dto.HobbyDto;
import com.human.util.DBConn;

//ctr+a 모든 코드 선택 ctr+shift+f 코드정리
public class CustomerHobbyDao {
    public ArrayList<CustomerHobbyDto> selecAllCustomerAndHobby() {
        ArrayList<CustomerHobbyDto> resultDtos = new
ArrayList<CustomerHobbyDto>();
//        select customer.*,hobby.hobby from customer, hobby
//        where customer.id=hobby.id;
        ResultSet rs = DBConn.createStatement(String.format(
                "select customer.*,hobby.hobby from customer, hobby " +
                "where customer.id=hobby.id(+)"));

        try {
            while (rs.next()) {
                resultDtos.add(new CustomerHobbyDto(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getDouble("height"),

rs.getTimestamp("birthday").toLocalDateTime(),
                    new
HobbyDto(rs.getInt("id"),rs.getString("hobby"))
                )
            );
        }
    } catch (Exception e) {

```

```

        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return resultDtos;
}

public CustomerHobbyDto selectOneCustomerAndHobby(int id, String hobby) {
    CustomerHobbyDto resultDto = null;
//    select customer.* , hobby.hobby from customer, hobby
//    where customer.id=hobby.id;

//id가 있을때
//select customer.* , hobby.hobby from customer, hobby
//where customer.id=hobby.id(+)
//and customer.id=31 and hobby.hobby is null;

//id가 없을때
//select customer.* , hobby.hobby from customer, hobby
//where customer.id=hobby.id(+)
//and customer.id=101 and hobby.hobby ='1';

    String sql="";
    if(hobby.equals("")) {

        sql="select customer.* , hobby.hobby from customer, hobby " +
            "where customer.id=hobby.id(+) and
customer.id="+id+" and hobby.hobby is null";
    }else {
        sql="select customer.* , hobby.hobby from customer, hobby " +
            "where customer.id=hobby.id(+) and
customer.id="+id+" and hobby.hobby='"+hobby+"'";

    }
    ResultSet rs = DBConn.createStatementQuery(String.format(sql));

    try {
        while (rs.next()) {
            resultDto=new CustomerHobbyDto(
                rs.getInt("id"),
                rs.getString("name"),
                rs.getDouble("height"),

rs.getTimestamp("birthday").toLocalDateTime(),
                new
HobbyDto(rs.getInt("id"),rs.getString("hobby"))
            );
        }
    }
}

```

```

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return resultDto;
    }
}

//CustomerHobbyService.java
package com.human.service;
import java.time.LocalDateTime;
import java.util.ArrayList;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.human.dao.CustomerDao;
import com.human.dao.CustomerHobbyDao;
import com.human.dao.HobbyDao;
import com.human.dto.CustomerDto;
import com.human.dto.CustomerHobbyDto;
import com.human.dto.HobbyDto;
public class CustomerHobbyService {
    public static CustomerDao customerDao=new CustomerDao();
    public static HobbyDao hobbyDao=new HobbyDao();
    public static CustomerHobbyDao customerHobbyDao=new CustomerHobbyDao();

    public static void insertDB(HttpServletRequest request,
HttpServletResponse response) {

        CustomerDto customerDto=new CustomerDto(null,
                request.getParameter("name"),
                Double.parseDouble(request.getParameter("height")),
                LocalDateTime.parse(request.getParameter("birthday"))
        );

        customerDao.insert(customerDto);

        String hobbys[]=request.getParameterValues("hobbys");
        HobbyDto hobbyDto= null;
        int hobbyId=customerDao.getMaxId();
        for(String hobbyName:hobbys) {
            hobbyDao.insert(new HobbyDto(hobbyId,hobbyName));
        }
    }

    public static void selectAll(HttpServletRequest request,
HttpServletResponse response) {
        ArrayList<CustomerHobbyDto> dtos=new ArrayList<CustomerHobbyDto>();
        dtos=customerHobbyDao.selectAllCustomerAndHobby();
    }
}

```

```

        System.out.println(dtos);
        request.setAttribute("dtos", dtos);
    }
    public static void selectOne(HttpServletRequest request,
HttpServletResponse response) {
    CustomerHobbyDto dto=customerHobbyDao.selectOneCustomerAndHobby(
        Integer.parseInt(request.getParameter("id")),
        request.getParameter("hobby")
    );
    System.out.println(dto);
    request.setAttribute("dto", dto);
}
public static void deleteHobby(HttpServletRequest request,
HttpServletResponse response) {
    hobbyDao.delete(Integer.parseInt(request.getParameter("id"))
        , request.getParameter("hobby"));

}
public static void deleteCustomer(HttpServletRequest request,
HttpServletResponse response) {
    hobbyDao.delete(Integer.parseInt(request.getParameter("id")));
    customerDao.delete(Integer.parseInt(request.getParameter("id")));
}
// <form action="/jsp20CustommerHobby/customerHobby/update.customerHobby">
// <input type="hidden" name="id" value="${requestScope.dto.id }>
// <input type="hidden" name="hobby" value="${requestScope.dto.hobby.hobby }>
// 고객의 키 수정:<input type="text" name='newHeight' ><br>
// 고객의 취미 변경:<input type="text" name="newHobby"><br>
// <input type="submit" value="수정">
//</form>

public static void update(HttpServletRequest request, HttpServletResponse
response) {
    customerDao.update(Integer.parseInt(request.getParameter("id")),
        Double.parseDouble(request.getParameter("newHeight")));
    hobbyDao.update(Integer.parseInt(request.getParameter("id")),
        request.getParameter("hobby"),
        request.getParameter("newHobby"));
}
}

//CustommerHobbyController.java
package com.human.controller;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;

```

```

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.human.service.CustomerHobbyService;
import com.human.service.CustomerService;
@WebServlet("*.customerHobby")
public class CustomerHobbyController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public CustomerHobbyController() { super(); }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // URI:/jsp10/hello.do
        // conPath:/jsp10
        // command:/hello.do
        String uri = request.getRequestURI();
        System.out.println("URI:" + uri);
        // 원하는 수소에 대한 처리 방법
        String conPath = request.getContextPath();
        System.out.println("conPath:" + conPath);

        String command = uri.substring(conPath.length());
        System.out.println("command:" + command);

        String viewPage = "selectAll.jsp";

//http://localhost:8081/jsp20CustomerHobby/customer/insert.customer
        if (command.equals("/customerHobby/insert.customerHobby")) {
            viewPage = "insert.jsp";
        System.out.println("/customerHobby/insert.customerHobby");
        } else if (command.equals("/customerHobby/insertDB.customerHobby")) {
            viewPage = "selectAll.customerHobby";
            CustomerHobbyService.insertDB(request, response);
            System.out.println("/customerHobby/insertDB.customerHobby");
        } else if (command.equals("/customerHobby/selectAll.customerHobby")) {
            viewPage = "selectAll.jsp";
            CustomerHobbyService.selectAll(request, response);
            System.out.println("/customerHobby/selectAll.customerHobby");
        }else if(command.equals("/customerHobby/selectOne.customerHobby")) {
            viewPage = "selectOne.jsp";
            CustomerHobbyService.selectOne(request, response);
            System.out.println("/customerHobby/selectOne.customerHobby");
        }else if(command.equals("/customerHobby/deleteCustomer.customerHobby")) {
            viewPage ="selectAll.customerHobby";
            CustomerHobbyService.deleteCustomer(request,response);
            System.out.println("/customerHobby/deleteCustomer.customerHobby");
        }else if(command.equals("/customerHobby/deleteHobby.customerHobby")) {
            viewPage ="selectAll.customerHobby";

```

```

        CustomerHobbyService.deleteHobby(request, response);
        System.out.println("/customerHobby/deleteHobby.customerHobby");
    }else if(command.equals("/customerHobby/update.customerHobby")) {
        viewPage ="selectAll.customerHobby";
        CustomerHobbyService.update(request, response);
        System.out.println("/customerHobby/update.customerHobby");
    }
    System.out.println(viewPage);
    RequestDispatcher dispatcher = request.getRequestDispatcher(viewPage);
    dispatcher.forward(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doGet(request, response);
}
}

//customerHobby/insert.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
<
<script>
$(function(){
    $("button").click(function(){
        /* alert("Test"); */
        event.preventDefault();
        /* $("#displayHobby").html($("#displayHobby").html()
           +"<div><input type=text name=hobbys
><button>X</button></div>"); */
        //append,prepend after before
        $("#displayHobby").append("<div><input type=text name=hobbys
><button>X</button></div>");
    })

    $("#displayHobby").on("click","button",function(){
        event.preventDefault();
        //alert("test");
        $(this).parent().remove();
    })
})

```

```

</script>
</head>
<body>
<form action="/jsp20CustommerHobby/customerHobby/insertDB.customerHobby"
method="get">
    이름:<input type=text name=name ><br>
    키:<input type=text name=height ><br>
    생일:<input type="datetime-local" name=birthday ><br>
    하비:
        <!-- <input type=text name=hobbys >
        <input type=text name=hobbys >
        <input type=text name=hobbys ><br> -->
        <div id='displayHobby'>

    </div>
    <button>하비 추가</button>
    <input type="submit" value=전송 >
</form>
</body>
</html>

//customerHobby/selectAll.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<table border="1" width="90%" id="customers">
    <tr>
        <th>id</th>
        <th>name</th>
        <th>height</th>
        <th>birthday</th>
        <th>hobby</th>
    </tr>
    <c:forEach items="${requestScope.dtos }" var="customerHobby">
    <tr>
        <td>${customerHobby.id }</td>
        <td><a href="/jsp20CustommerHobby/customerHobby/selectOne.customerHobby
            ?id=${customerHobby.id }&hobby=${customerHobby.hobby.hobby }">
            ${customerHobby.name }</a></td>
        <td>${customerHobby.height}</td>

```

```

<td>${customerHobby.birthday }</td>
<td>${customerHobby.hobby.hobby }</td>

</tr>
</c:forEach>
</table>
<a href="/jsp20CustommerHobby/customerHobby/insert.customerHobby">추가</a>
</body>
</html>

//customerHobby/selectOne.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<table border="1" width="90%" id="customers">
  <tr>
    <th>id</th>
    <th>name</th>
    <th>height</th>
    <th>birthday</th>
  </tr>
  <fmt:parseDate value="${requestScope.dto.birthday}"
    pattern="yyyy-MM-dd'T'HH:mm"
    var="now" type="both" ></fmt:parseDate>
  <tr>
    <td>${requestScope.dto.id }</td>
    <td>${requestScope.dto.name }</td>
    <td>${requestScope.dto.height}</td>
    <td><fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일 hh시
mm분 ss초"/></td>
  </tr>
</table>

<h1>HOBBY 정보</h1>
<table border="1" width="40%" id="hobbys">
  <tr>
    <th>ID</th>
    <th>취미</th>
  </tr>
  <tr>

```

```

        <td>${requestScope.dto.id }</td>
        <td>${requestScope.dto.hobby }</td>
    </tr>
</table>
<form action="/jsp20CustommerHobby/customerHobby/update.customerHobby">
    <input type="hidden" name="id" value=${requestScope.dto.id }>
    <input type="hidden" name="hobby"
value=${requestScope.dto.hobby.hobby }>
        고객의 키 수정:<input type="text" name='newHeight' ><br>
        고객의 취미 변경:<input type="text" name="newHobby"><br>
        <input type="submit" value="수정">
</form>
<a
href="/jsp20CustommerHobby/customerHobby/selectAll.customerHobby">목록가기</a>
<a href="/jsp20CustommerHobby/customerHobby/deleteCustomer.customerHobby
?id=${requestScope.dto.id}">고객삭제</a>
<a href="/jsp20CustommerHobby/customerHobby/deleteHobby.customerHobby
?id=${requestScope.dto.id}&hobby=${requestScope.dto.hobby.hobby}">하비삭제</a>
</body>
</html>

```

---

## > 20. CustomHobby-CustomHobbys join 만들기

---

```
//CustomerHobbysDto.java
package com.human.dto;
import java.time.LocalDateTime;
import java.util.List;
public class CustomerHobbysDto {
    private int id;
    private String name;
    private double height;
    private LocalDateTime birthday;

    private List<HobbyDto> hobbys;

    public CustomerHobbysDto() {}
    public CustomerHobbysDto(int id, String name, double height,
LocalDateTime birthday, List<HobbyDto> hobbys) {
        super();
        this.id = id;
        this.name = name;
        this.height = height;
        this.birthday = birthday;
        this.hobbys = hobbys;
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((birthday == null) ? 0 :
birthday.hashCode());
        long temp;
        temp = Double.doubleToLongBits(height);
        result = prime * result + (int) (temp ^ (temp >>> 32));
        result = prime * result + ((hobbys == null) ? 0 :
hobbys.hashCode());
        result = prime * result + id;
        result = prime * result + ((name == null) ? 0 : name.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
```

```

        if (getClass() != obj.getClass())
            return false;
        CustomerHobbysDto other = (CustomerHobbysDto) obj;
        if (birthday == null) {
            if (other.birthday != null)
                return false;
        } else if (!birthday.equals(other.birthday))
            return false;
        if (Double.doubleToLongBits(height) !=
Double.doubleToLongBits(other.height))
            return false;
        if (hobbys == null) {
            if (other.hobbys != null)
                return false;
        } else if (!hobbys.equals(other.hobbys))
            return false;
        if (id != other.id)
            return false;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "CustomerHobbysDto [id=" + id + ", name=" + name + ",
height=" + height + ", birthday=" + birthday
                + ", hobbys=" + hobbys + "]";
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getHeight() {
        return height;
    }
}

```

```

public void setHeight(double height) {
    this.height = height;
}
public LocalDateTime getBirthday() {
    return birthday;
}
public void setBirthday(LocalDateTime birthday) {
    this.birthday = birthday;
}
public List<HobbyDto> getHobbys() {
    return hobbys;
}
public void setHobbys(List<HobbyDto> hobbys) {
    this.hobbys = hobbys;
}
}

//CustomerHobbysService.java
package com.human.service;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.human.dao.CustomerDao;
import com.human.dao.CustomerHobbyDao;
import com.human.dao.HobbyDao;
import com.human.dto.CustomerDto;
import com.human.dto.CustomerHobbyDto;
import com.human.dto.CustomerHobbysDto;
import com.human.dto.HobbyDto;

public class CustomerHobbysService {
    public static CustomerDao customerDao=new CustomerDao();
    public static HobbyDao hobbyDao=new HobbyDao();
    public static CustomerHobbyDao customerHobbyDao=new CustomerHobbyDao();

    public static void insertDB(HttpServletRequest request,
HttpServletResponse response) {

        CustomerDto customerDto=new CustomerDto(null,
            request.getParameter("name"),
            Double.parseDouble(request.getParameter("height")),
            LocalDateTime.parse(request.getParameter("birthday"))
        );

        customerDao.insert(customerDto);
    }
}

```

```

String hobbys[] = request.getParameterValues("hobbys");
HobbyDto hobbyDto = null;

int hobbyId = customerDao.getMaxId();

for (String hobbyName : hobbys) {
    hobbyDao.insert(new HobbyDto(hobbyId, hobbyName));
}

}

//(int id, String name, double height, LocalDateTime birthday,
List<HobbyDto> hobbys) {

    public static void selectAll(HttpServletRequest request,
HttpServletRequest response) {
        ArrayList<CustomerHobbiesDto> customerHobbiesDtos = new
ArrayList<CustomerHobbiesDto>();
        ArrayList<CustomerDto> customerDtos = customerDao.selectAll();
        for (CustomerDto dto : customerDtos) {
            ArrayList<HobbyDto> hobbyDtos = hobbyDao.selectId(dto.getId());

            customerHobbiesDtos.add(new CustomerHobbiesDto(
                dto.getId(),
                dto.getName(),
                dto.getHeight(),
                dto.getBirthDay(),
                hobbyDtos
            ));
        }

        request.setAttribute("dtos", customerHobbiesDtos);
    }

    public static void selectOne(HttpServletRequest request,
HttpServletRequest response) {
        CustomerHobbyDto dto = customerHobbyDao.selectOneCustomerAndHobby(
            Integer.parseInt(request.getParameter("id")),
            request.getParameter("hobby")
        );
        System.out.println(dto);
        request.setAttribute("dto", dto);
    }
}

```

```

    public static void deleteHobby(HttpServletRequest request,
HttpServletResponse response) {
    hobbyDao.delete(Integer.parseInt(request.getParameter("id"))
        , request.getParameter("hobby"));

}

    public static void deleteCustomer(HttpServletRequest request,
HttpServletResponse response) {
    hobbyDao.delete(Integer.parseInt(request.getParameter("id")));

    customerDao.delete(Integer.parseInt(request.getParameter("id")));
}
// <form action="/jsp20CustommerHobby/customerHobby/update.customerHobby">
// <input type="hidden" name="id" value=${requestScope.dto.id }>
// <input type="hidden" name="hobby" value=${requestScope.dto.hobby.hobby }>
// 고객의 키 수정:<input type="text" name='newHeight' ><br>
// 고객의 취미 변경:<input type="text" name="newHobby"><br>
// <input type="submit" value="수정">
//</form>

    public static void update(HttpServletRequest request, HttpServletResponse
response) {
    // TODO Auto-generated method stub

    customerDao.update(Integer.parseInt(request.getParameter("id")),
        Double.parseDouble(request.getParameter("newHeight")));

    hobbyDao.update(Integer.parseInt(request.getParameter("id")),
        request.getParameter("hobby"),
        request.getParameter("newHobby"));

}

}

```

```

//CustomerHobbysController.java
package com.human.controller;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.human.service.CustomerHobbysService;
import com.human.service.CustomerService;

/**
 * Servlet implementation class CustomerController
 */
@WebServlet("*.customerHobbys")
public class CustomerHobbysController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public CustomerHobbysController() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // URI:/jsp10/hello.do
        // conPath:/jsp10
        // command:/hello.do
        String uri = request.getRequestURI();
        System.out.println("URI:" + uri);
        // 원하는 주소에 대한 처리 방법
        String conPath = request.getContextPath();
        System.out.println("conPath:" + conPath);
        String command = uri.substring(conPath.length());
        System.out.println("command:" + command);

        String viewPage = "selectAll.jsp";

```

```

//http://localhost:8081/jsp20CustomerHobby/customer/insert.customer
    if (command.equals("/customerHobbys/insert.customerHobbys")) {
        viewPage = "insert.jsp";
        System.out.println("/customerHobbys/insert.customerHobbys");
    }else if(command.equals("/customerHobbys/insertDB.customerHobbys")){
        viewPage = "selectAll.customerHobbys";
        CustomerHobbysService.insertDB(request, response);
        System.out.println("/customerHobbys/insertDB.customerHobbys");
    } else if (command.equals("/customerHobbys/selectAll.customerHobbys")) {
        viewPage = "selectAll.jsp";
        CustomerHobbysService.selectAll(request, response);
        System.out.println("/customerHobbys/selectAll.customerHobbys");
    }else if(command.equals("/customerHobbys/deleteHobby.customerHobbys")) {
        viewPage ="selectAll.customerHobbys";
        CustomerHobbysService.deleteHobby(request,response);
        System.out.println("/customerHobbys/deleteHobby.customerHobbys");
    }else if(command.equals("/customerHobbys/deleteCustomer.customerHobbys")) {
        viewPage ="selectAll.customerHobbys";
        CustomerHobbysService.deleteCustomer(request,response);
        System.out.println("/customerHobbys/deleteCustomer.customerHobbys");
    }
    System.out.println(viewPage);
    RequestDispatcher dispatcher = request.getRequestDispatcher(viewPage);
    dispatcher.forward(request, response);
}
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    doGet(request, response);
}
}

```

```

//customerHobby/insert.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
<
<script>
$(function(){
    $("button").click(function(){

```

```

        /* alert("Test"); */
        event.preventDefault();
        /* $("#displayHobby").html($("#displayHobby").html()
            +"<div><input type=text name=hobbys
        ><button>X</button></div>"); */
            //append,prepend    after before
            $("#displayHobby").append("<div><input type=text name=hobbys
        ><button>X</button></div>");
    })

    $("#displayHobby").on("click","button",function(){
        event.preventDefault();
        //alert("test");
        $(this).parent().remove();
    })
}

</script>
</head>
<body>
<form action="/jsp20CustommerHobby/customerHobbies/insertDB.customerHobbies"
method="get">
    이름:<input type=text name=name ><br>
    나이:<input type=text name=height ><br>
    생일:<input type="datetime-local" name=birthday ><br>
    하비:
    <!-- <input type=text name=hobbys >
    <input type=text name=hobbys >

    <input type=text name=hobbys ><br> -->
    <div id='displayHobby'>

    </div>
    <button>하비 추가</button>
    <input type="submit" value=전송 >
</form>
</body>
</html>

//customerHobby/selectAll.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html>
<html>
<head>
```

```

<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>고객 정보</h1>
<table border="1" width="90%" id="customers">
    <tr>
        <th>id</th>
        <th>name</th>
        <th>height</th>
        <th>birthday</th>
        <th>hobbies</th>
    </tr>
    <c:forEach items="${dtos }" var="customerHobby">
        <tr>
            <td>${customerHobby.id }</td>
            <td><a href='/jsp20CustommerHobby/customerHobbies/deleteCustomer.customerHobbies?id=${customerHobby.id}'>${customerHobby.name }</a></td>
            <td>${customerHobby.height}</td>
            <td>${customerHobby.birthday }</td>
            <td>
                <c:forEach items="${customerHobby.hobbies }" var="hobby">
                    <a href='/jsp20CustommerHobby/customerHobbies/deleteHobby.customerHobbies?id=${hobby.id}&hobby=${hobby.hobby}'>${hobby.hobby} </a>
                </c:forEach>
            </td>
        </tr>
    </c:forEach>
</table>
<a href="/jsp20CustommerHobby/customerHobbies/insert.customerHobbies">고객추가</a>
</body>
</html>

//customerHobby/selectOne.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
```

```

<table border="1" width="90%" id="customers">
    <tr>
        <th>id</th>
        <th>name</th>
        <th>height</th>
        <th>birthday</th>
    </tr>
    <fmt:parseDate value="${requestScope.dto.birthday}"
        pattern="yyyy-MM-dd'T'HH:mm"
        var="now" type="both" ></fmt:parseDate>
    <tr>

        <td>${requestScope.dto.id }</td>
        <td>${requestScope.dto.name }</td>
        <td>${requestScope.dto.height}</td>
        <td><fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일 hh시
mm분 ss초"/></td>
    </tr>
</table>

<h1>HOBBY 정보</h1>
<table border="1" width="40%" id="hobbys">
    <tr>
        <th>ID</th>
        <th>취미</th>
    </tr>
    <tr>
        <td>${requestScope.dto.id }</td>
        <td>${requestScope.dto.hobby }</td>
    </tr>
</table>

<form action="/jsp20CustommerHobby/customerHobby/update.customerHobby">
    <input type="hidden" name="id" value=${requestScope.dto.id }>
    <input type="hidden" name="hobby"
value=${requestScope.dto.hobby.hobby }>
    고객의 키 수정:<input type="text" name='newHeight' ><br>
    고객의 취미 변경:<input type="text" name="newHobby"><br>
    <input type="submit" value="수정">
</form>

<a
href="/jsp20CustommerHobby/customerHobby/selectAll.customerHobby">목록가기</a>
<a href="/jsp20CustommerHobby/customerHobby/deleteCustomer.customerHobby
?id=${requestScope.dto.id}">고객삭제</a>
<a href="/jsp20CustommerHobby/customerHobby/deleteHobby.customerHobby
?id=${requestScope.dto.id}&hobby=${requestScope.dto.hobby.hobby}">하비삭제</a>

```

```
</body>
</html>
```

//sql에서는 다음과 같이 기술 하여 시간을 넣으면 yyyy-MM-dd'T'HH:mm:ss.SSS형태로 들어간다.

```
insert into customer values(1000, 'asd', 123, sysdate);
//html에서 받아서 넣으면 "yyyy-MM-dd'T'HH:mm"형태로 들어간다.
LocalDateTime.parse(request.getParameter("birthday"),
DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm")));
```

1000	<a href="#">asd</a>	123.0	2022년 12월 26일 02시 32분 00초	2022-12-26T14:32:48
1	<a href="#">park</a>	123.4	2022년 12월 25일 12시 19분 00초	2022-12-25T12:19

//다음 코드에서 밑줄 부분을 pattern="yyyy-MM-dd'T'HH:mm:ss.SSS"으로 변경하면 초, 분 데이터가 들어 있지 않은 html에서 받아서 처리한 데이터에 문제가 발생한다. 다음과 같은 형태로 출력하면 초, 미리세컨드가 없는 데이터 둘다 출력이 된다.

```
<%
    java.time.LocalDateTime ld=java.time.LocalDateTime.now();
    request.setAttribute("ld", ld);
%>

<fmt:parseDate value="${requestScope.ld }"
pattern="yyyy-MM-dd'T'HH:mm:ss.SSS"
var="now" type="both" ></fmt:parseDate>

<c:forEach items="${dtos }" var="customerDto">
<fmt:parseDate value="${customerDto.birthday}"
pattern="yyyy-MM-dd'T'HH:mm"
var="now" type="both" ></fmt:parseDate>
    <tr>
        <td>${customerDto.id }</td>
        <td><a href='/jsp20CustomerHobby/customer/selectOne?id=${customerDto.id}'>
            ${customerDto.name }</a></td>
        <td>${customerDto.height}</td>
        <td><fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일 hh시
mm분 ss초"/></td>
        <td>${customerDto.birthday}</td>
    </tr>
</c:forEach>
```