[HTML]

<div class=”right”>

<h2>

<span>반가워요!</span>

<div class=”emoticon”>aa</div>

</h2>

</div>

[CSS]

.right > h2{

position:relative;

}

.right .emoticon{

Position:absolute;

Top:0px;

Right:25%;

Color:#fff;

}

DOM: 브라우저에서는 HTML코드를 DOM이라는 객체 형태의 모델로 저장.

그렇게 저장된 정보를 DOM Tree라고 한다.

getElementById() 등 브라우저는 DOM Tree찾고 조작하는 걸 쉽게 도와주는

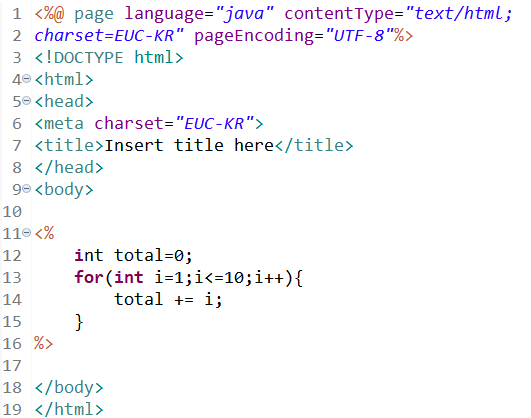
여러 가지 메서드(DOM API)를 제공한다.

예시)

document.getElementById("nav-cart-count").id;

Ajax: 비동기로 서버로부터 데이터를 가져오는 것이다.

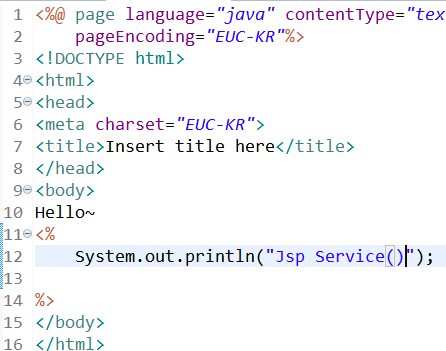
모든 JSP는 Servlet으로 바뀌어 동작한다.



**JSP의 실행순서**

1. 브라우저가 웹서버에 JSP에 대한 요청 정보를 전달한다.
2. 브라우저가 요청한 JSP가 최초로 요청했을 경우만 JSP로 작성된 코드가 서블릿으로 코드로 변환한다. (java 파일 생성)
3. 서블릿 코드를 컴파일해서 실행가능한 bytecode로 변환한다. (class 파일 생성)
4. 서블릿 클래스를 로딩하고 인스턴스를 생성한다.
5. 서블릿이 실행되어 요청을 처리하고 응답 정보를 생성한다.

<JSP파일>

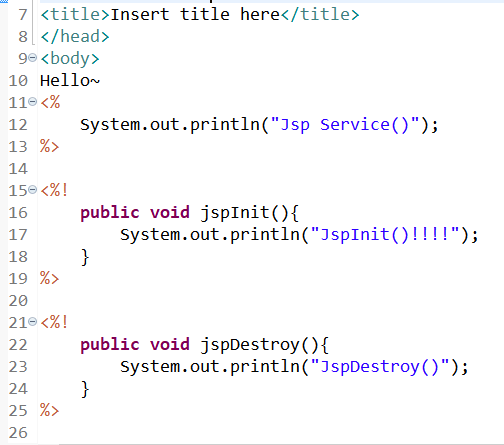


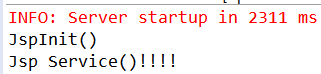
JSP파일이 JAVA파일로 바뀐 모습

JAVA파일에 public void \_jspService() 내에 작성된다.

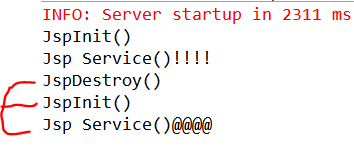


==============================





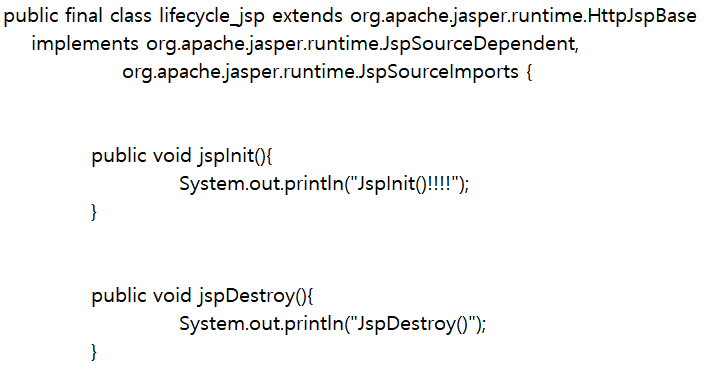
이 상태에서 페이지 jsp내용을 수정 후 재실행



빨간색으로 친 부분이 실행 결과

Jsp의 java파일을 확인해보면

jsp내에서 <%! %>로 선언한 부분은 public void \_jspService() 밖에 작성된다.



스크립트 요소

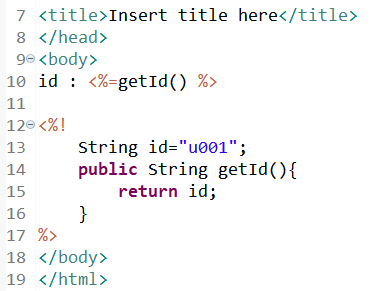
선언문, 스크립트릿, 표현식이라는 3가지 스크립트 요소가 존재한다.

선언문: <%! %>: JSP페이지 내에서 필요한 멤버변수나 전역변수 선언 및 메소드 선언해 사용

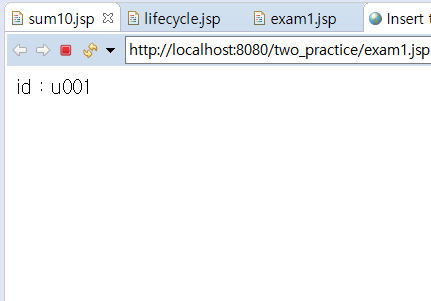
스크립트릿: <% %>: 프로그래밍 코드 기술에 사용

스크립트: <%= %>: 화면에 출력한 내용 기술에 사용

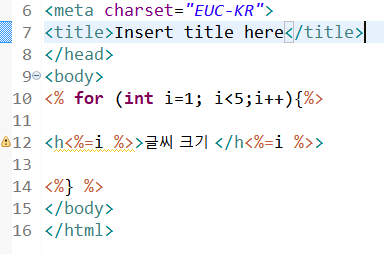
예제 1)



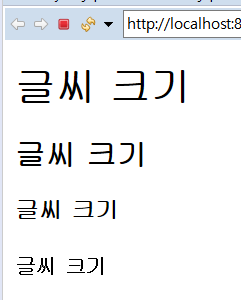
예제 1) 실행결과



예제 2)HTML + 스크립트를 활용



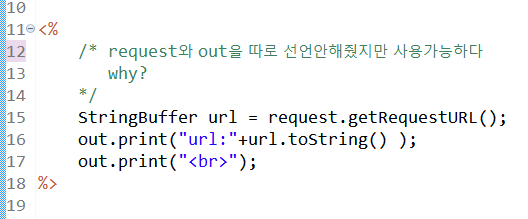
예제 2) 실행결과



HTML 주석 <!-- -->

JSP 주석 <%-- -->

JAVA 주석 // or <% %>



JSP는 서블릿으로 바뀌면서 내장객체를 사용할 수 있다.

**\*내장객체 종류**

request: HTML FORM 요소 선택 값과 같은 사용자 입력 정보를 읽어올 때 사용

response: 사용자 요청에 대한 응답을 처리할 때

pageContext: 현재 JSP 실행에 대한 context 정보를 참조하기 위해 사용

session: 클라이언트 세션 정보를 처리하기 위해 사용

out: 사용자에게 전달하기 위한 output스트림을 처리하기 위해 사용

config: 현재 JSP에 대한 환경을 처리하기 위해 사용

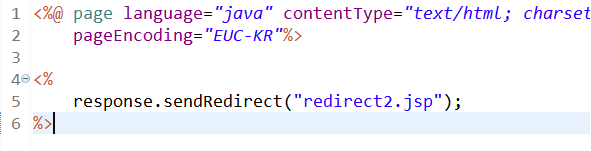
page: 현재 JSP페이지에 대한 클래스 정보

exception: 예외 어치를 위해 사용

**\*redirect**

2가지 방법이 있다.

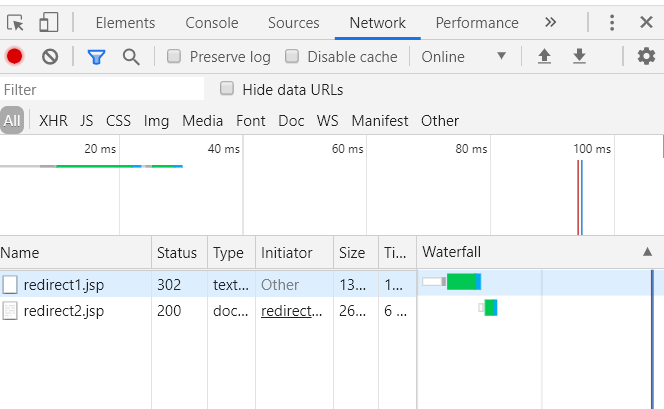
1. JSP에서 보낸다.



2. 서블릿에서 보낸다



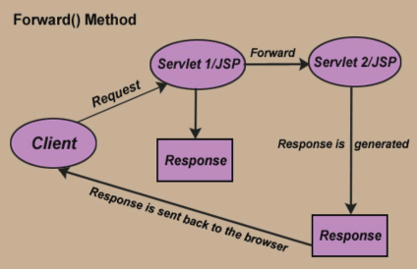
Chrome을 키고 F12를 누른 후 Network 보면



의 순서를 볼 수 있다.

**\*Forward**

서블릿이 다른 서블릿에게 작업을 넘겨줄 때 사용한다.



Rediect는 클라이언트-> 서버

서버-> 클라이언트

(보통 url이 바뀐다. Request가 여러 개)

Forward는 클라이언트-> 서버

서버->서버

서버->클라이언트

(보통 url이 같다. Request가 1개)

Servelet1은 요청을 처리한 후, 그 결과를 HttpServletRequest에 저장하고 Servlet1은 결과가 저장된 HttpServletRequest와 응답을 위한 HttpServletResponse를 같은 웹 어플리케이션 안에 있는 Servlet2에게 전송(forward)

**\*Forward 실습**

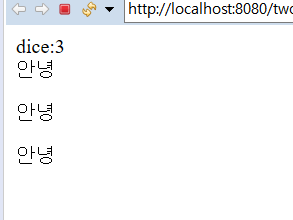
FrontServlet.java



NextServlet.java

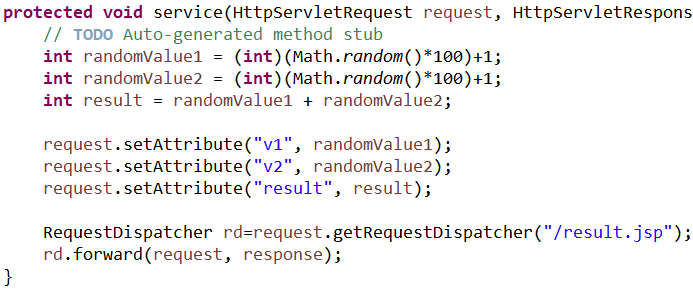


실행결과

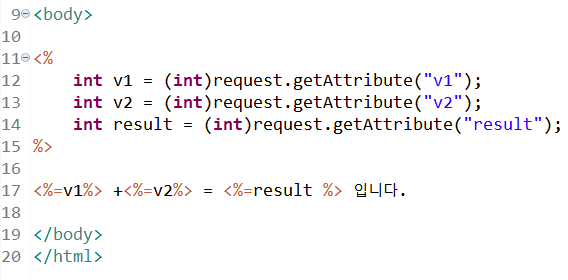


**\*Forward 실습2**

LogicServlet.java

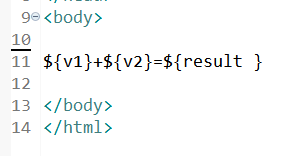
POST나 GET으로 보낼 요청이 없으면 그냥 SERVICE에 만들어도 상관없어 이렇게 만들었다.

result.jsp

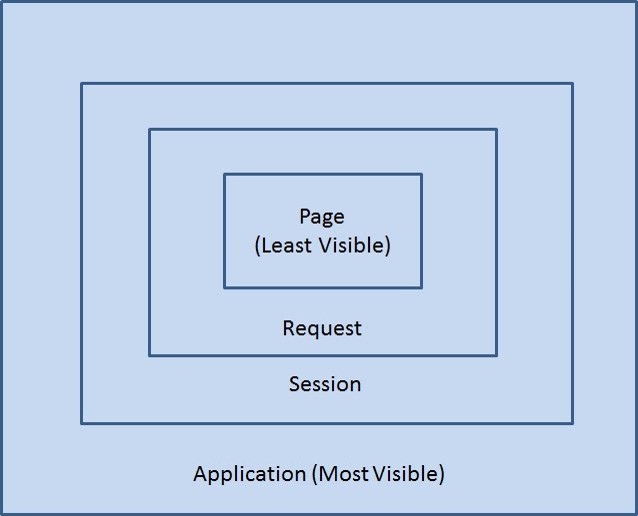


jsp파일에 java코드가 계속해서 들어가는 것이 불편하여 만들어진 개념이 JSTL, EL이다.

\*EL의 예시



**\*Scope**



-Page scope

1. PageContext 추상클래스를 사용

2. JSP페이지에서 pageContext라는 내장 객체로 사용 가능하다.

3. 마치 지역변수처럼 사용된다.

4. forward가 될 경우 해당 Page scope에 지정된 변수는 사용할 수 없다.

보통 2가지로 사용한다.

pageContext.setAttribute()

pageContext.getAttribute()

-Request scope

: http요청을 WAS가 받아서 웹 브라우저에게 응답할 때까지 변수 값을 유지하고자 할 경우 사용한다.

: HttpServletRequest객체를 사용한다.

보통 2가지 방법으로 사용한다.

: request.setAttribute()

: request.getAttribute()

-Session scope

: 웹 브라우저 별로 변수를 관리하고자 할 경우 사용한다.

: 웹 브라우저간의 Tab 간에는 세션이 공유.

: HttpSession인터페이스를 구현한 객체를 사용

: JSP에서느 session내장 변수를 사용한다.

: getAttribute(), setAttribute()

-Application Scope

: 웹 어플리케이션이 시작되고 종료될 때까지 변수를 사용할 수 있다.

: 하나의 서버에는 여러 개의 어플리케이션이 존재할 수 있다.

: 어떤 클라이언트가 특정 웹 어플리케이션에 접근하더라도 모든 클라이언트가 공통으로 사용해야할 값이 있을 때 사용합니다.

: 서블릿의 경우 getServletContext()를 이용하여 application 객체를 이용한다.

: getAttribute(), setAttribute()를 사용.

**\*ApplicationScope 실습**

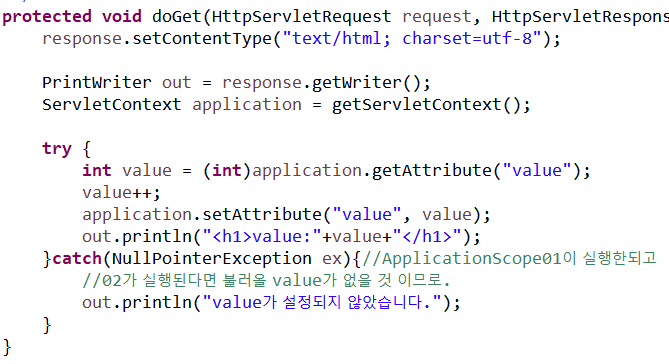
ApplicationScope01.java

-application변수에 get으로 value값 넣어준다.



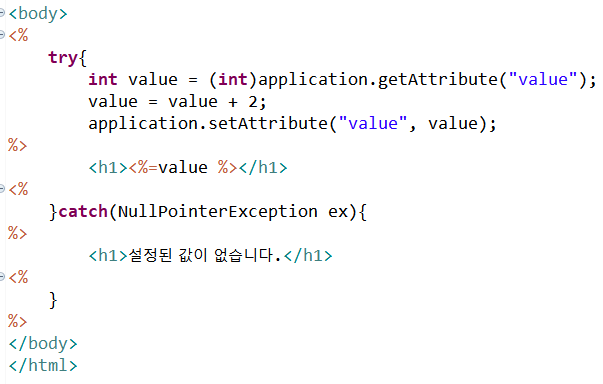
ApplicationScope02.java

- ApplicationScope01.java가 먼저 실행되지 않았다면 불러온 값이 없으므로 try 문 작성



ApplicationScope01.jsp

- ApplicationScope01.java가 먼저 실행되지 않았다면 불러온 값이 없으므로 try문 작성



**\*EL(Expression Language=표현언어)**

1.JSP의 스코프에 맞는 속성 사용

2.집합 객체에 대한 접근 방법 제공

3.자바 클래스 메소드 호출 기능 제공

표현언어의 표현 방법

${expr}

EX)

<b>${sessionScope.member.id}</b>님 환영합니다.

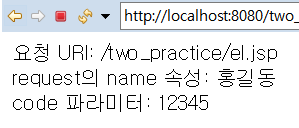
-EL 실습

URI: <http://localhost:8080/two_practice/el.jsp?&code=12345>

el.jsp파일



실행 결과



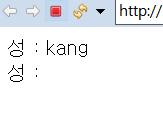
**\*JSTL(JSP Standard Tag Library)**

JSP페이지에서 조건문 처리, 반복문 처리 등을 html tag형태로 작성할 수 있게 도와준다.

실습 코드

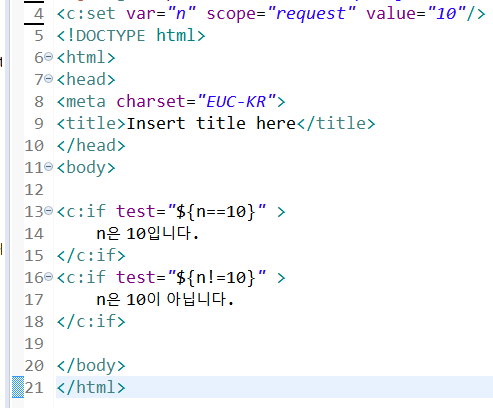


실행결과

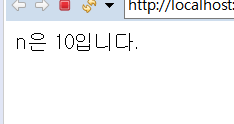


**\*JSTL 코어 태그: 변수 지원 태그**

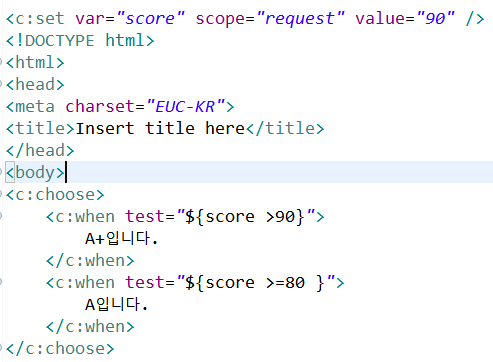
-흐름제어 태그 – if



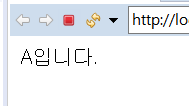
결과



-흐름제어 태그 – choose



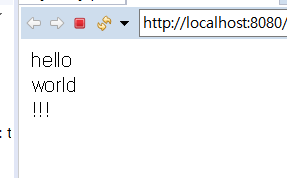
결과



-흐름제어 태그 – forEach



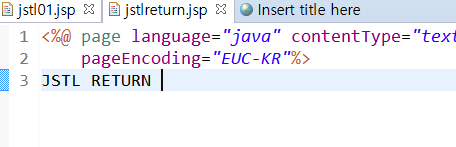
실행결과



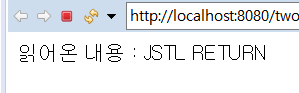
-흐름제어 태그 – import



jstlreturn.jsp



jstl01.jsp실행 결과

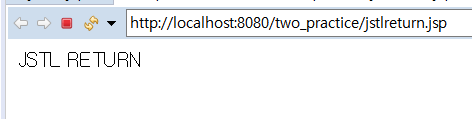


-흐름제어 태그 – redirect

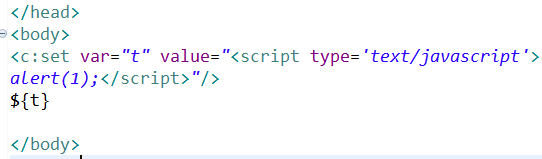
jstl01.jsp

<c:redirect url=*"http://localhost:8080/two\_practice/jstlreturn.jsp"*/>

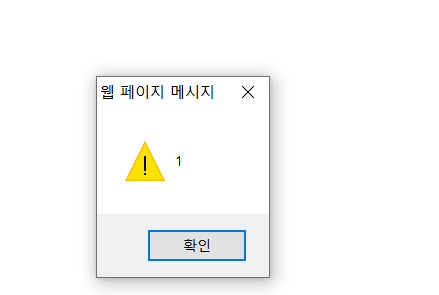
결과

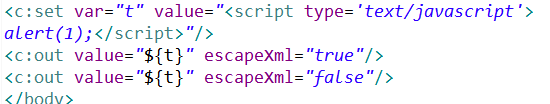


-흐름제어 태그 -out

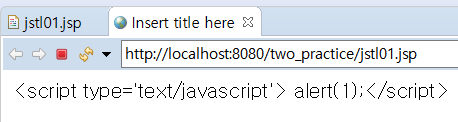


결과

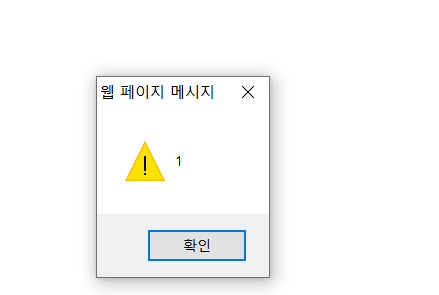




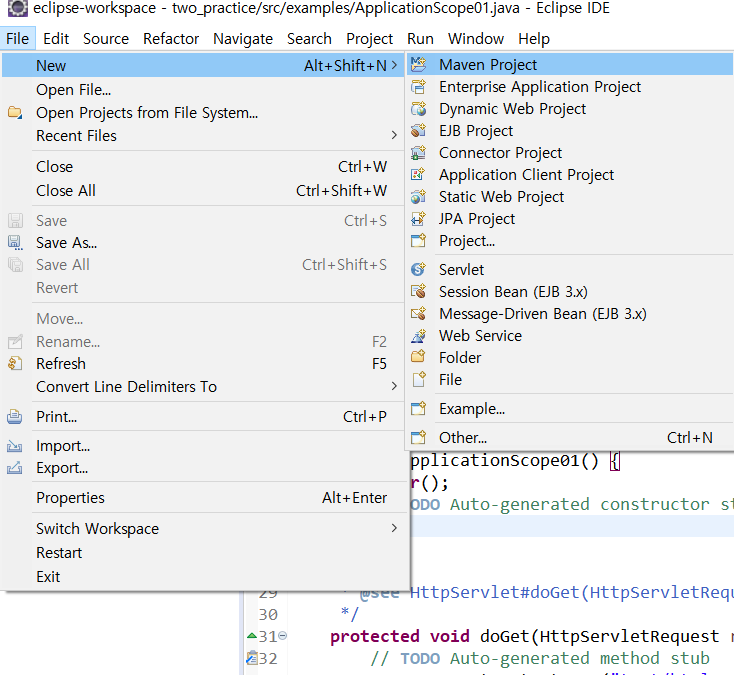
True일 경우 문자열로 출력



False일 경우 코드로 인식



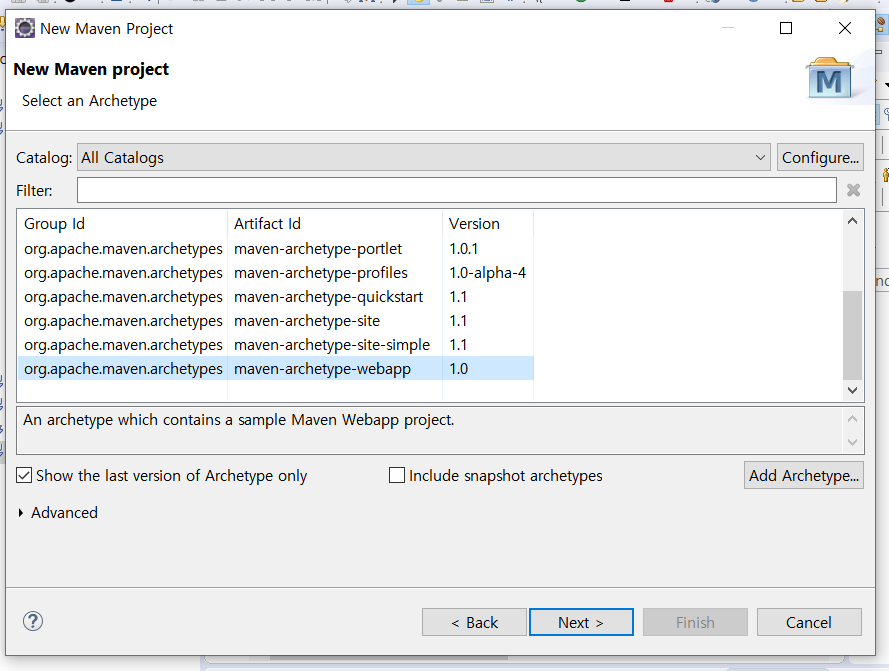
**\*Maven프로젝트 Setting**

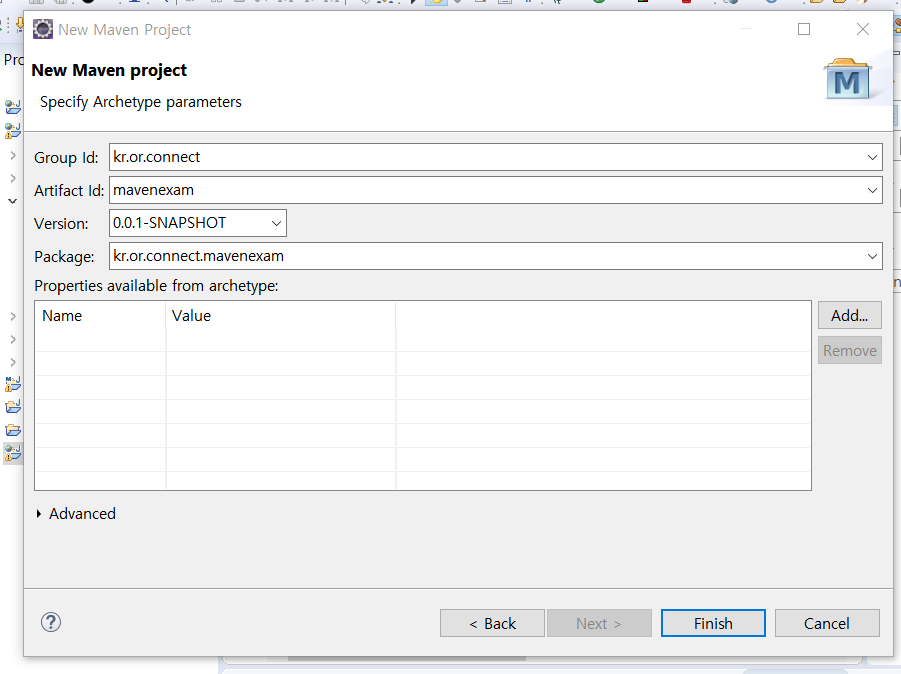


Maven Project 생성

Next를 눌러주고

Webapp 1.0을 선택해준다.

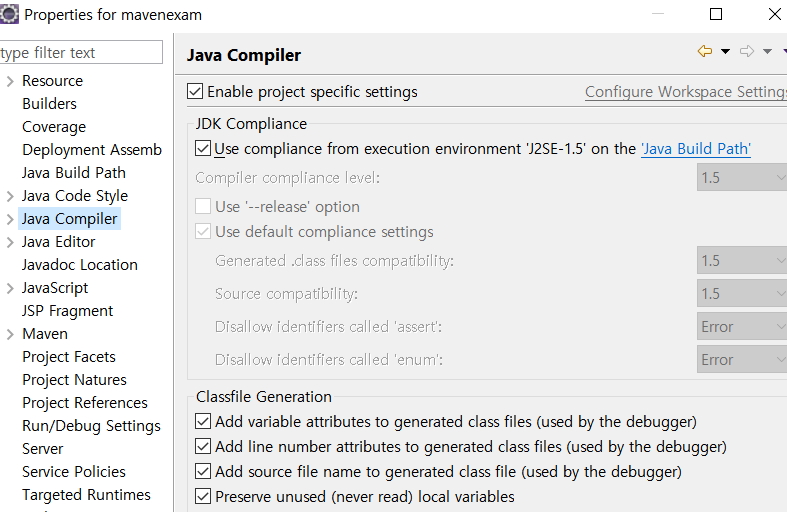




Artifact id를 mavenexam이라고 하고 Finish

-jdk버전확인

mavenexam프로젝트 우클릭-> property-> java compiler (기본이 jdk1.5로 설정)



-jdk버전바꾸기

Maven설정파일인 Pom.xml을 수정

Pom.xml의 <build>…. </build>부분에 코드 추가

코드 내용은

<finalName>mavenexam</finalName>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.6.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

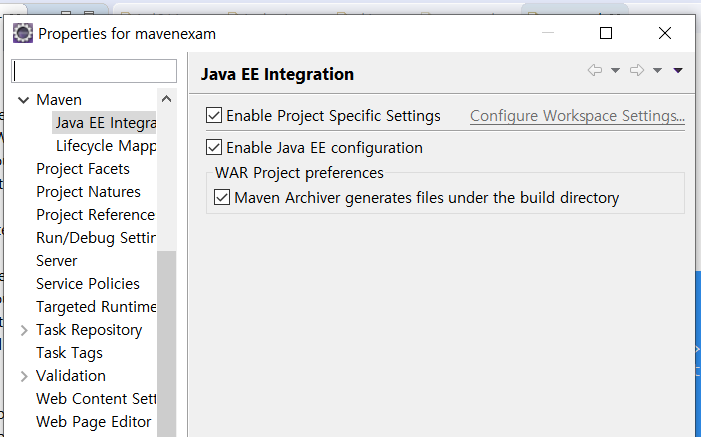
</configuration>

</plugin>

</plugins>

-정의한 컴파일러 버전 적용

Project우클릭-> property ->Maven-> Java EE Integration-> Enable Project Specific Setting 체크



-index.jsp에 들어가면 오류 발생

(HttpServlet을 찾을 수 없다 -> Maven프로젝트 경우 WAS런타임 지정을 안 했기 때문에 서블릿 라이브러리를 찾을 수 없다. 따라서 dependency 엘리먼트 아래에 다음을 추가한다.

<dependency>

<groupId>javax.servlet</groupId>

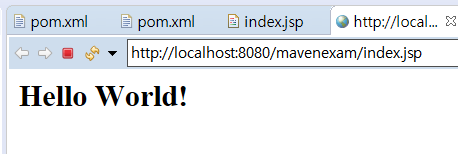
<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

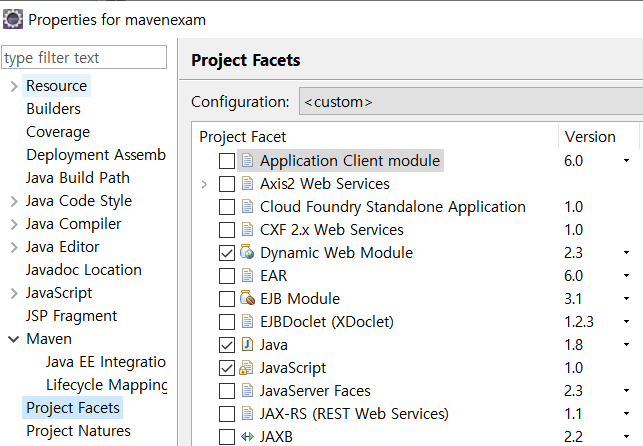
</dependency>

Index.jsp를 실행해보면



출력이 된다.

-el이나 jstl을 사용하려면 다이나믹 웹 모듈이 2.4이상이어야한다. 그러나 Maven프로젝트를 생성하면 다이나믹 웹 모듈 버전이 2.3인 것을 확인할 수 있다.



-web.xml 파일을 열어 내용을 다음과 같이 수정한다.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

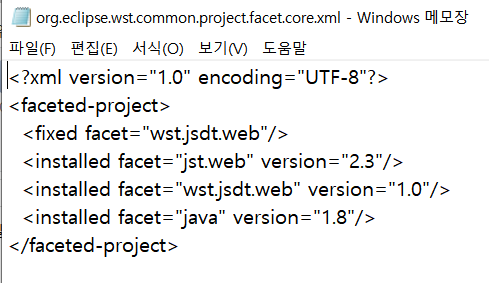
xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"* version=*"3.1"*>

<display-name>Archetype Created Web Application</display-name>

</web-app>

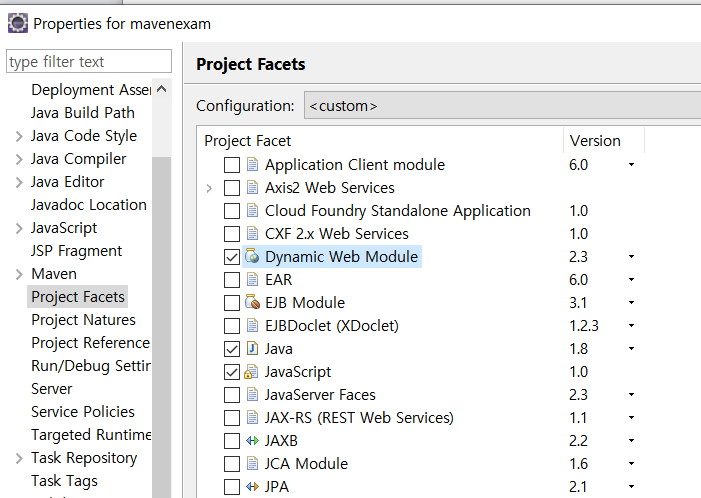
그리고 프로젝트 폴더로 가서

.settings/org.eclipse.wst.common.project.facet.core.xml 파일을 엽니다.



jst.web의 version 2.3을3.1로 수정합니다.

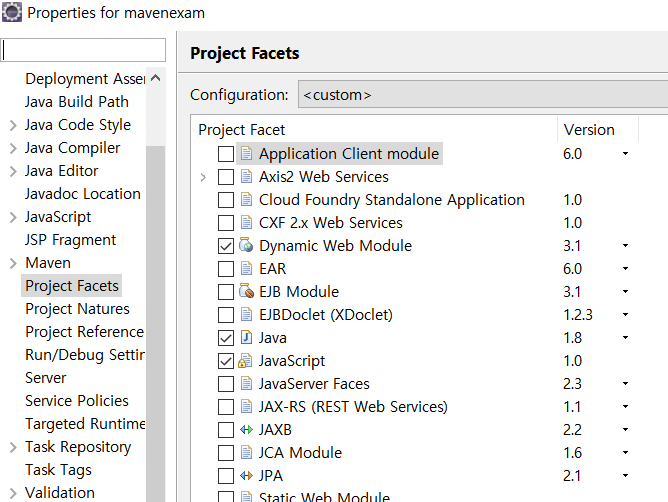
수정을 한 후 dynamic web version을 확인해보면



아직 2.3으로 뜨는데 이 경우에

1.기존 tomcat을 종료

2. 프로젝트 우클릭 Maven 메뉴 아래의 update project -> 확인

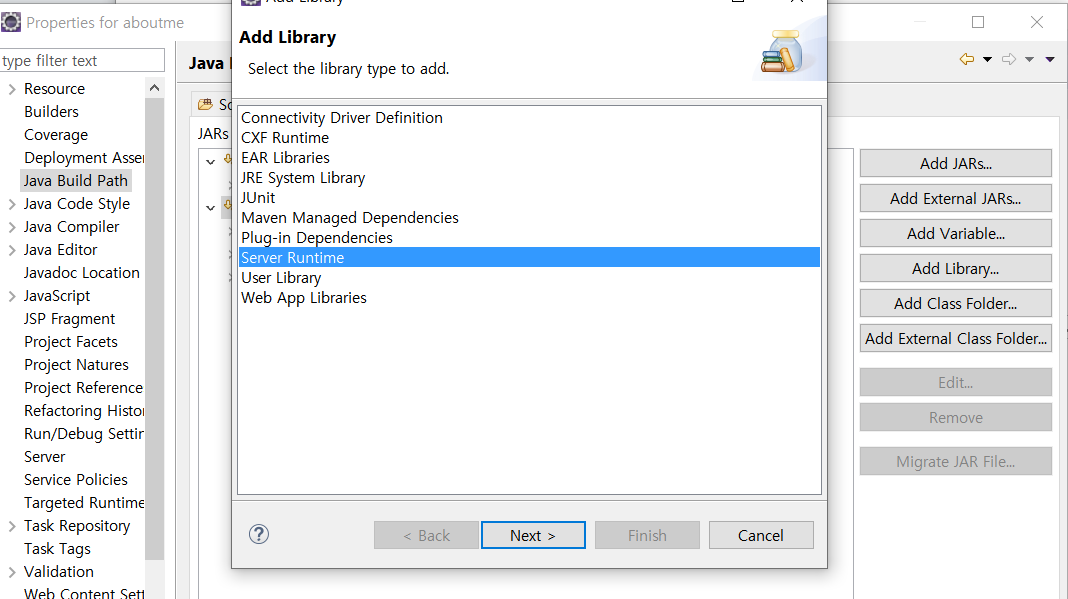


Dynamic Web Module 버전이 3.1로 변경

혹시 이렇게 해도 안된다면 상단의 Project 메뉴의 Clean선택해본다.

\*서버를 삭제 후 재생성하면 기존의 프로젝트에 x표시가 뜰 경우

프로젝트 우클릭-> Properties -> Java Build Path-> Classpath -> Add Library->ServerRuntime 해서 서버등록



**\*MYSQL**

# create user를 이용한 사용자 생성/추가

mysql> create user 'USER\_ID'@'localhost' identified by 'USER\_PASSWORD';

mysql> grant all privileges on DATABASE\_NAME.\* to USER\_ID@'localhost';

# 'localhost' 대신 '%' 을 사용할 경우 외부에서도 접속 가능

mysql> grant all privileges on DATABASE\_NAME.\* to USER\_ID@'%' identified by 'USER\_PASSWORD';

#DBMS에게 적용을 하라는 의미

mysql > flush privileges**;**

exit 눌러서 mysql종료 후 다시

mysql -hlocalhost -uconnectuser -p connectdb

패스워드 입력

쿼리 입력 중에 취소하고 싶으면

\c

**\*JDBC**

JDBC: 자바를 이용한 데이터베이스 접속과 SQL문장의 실행, 그리고 실행 결과로 얻어진 데이터의 핸들링을 제공하는 방법과 절차에 관한 규약

1. import java.sql.\*;

2. 드라이버를 로드 한다.

3. Connection 객체를 생성한다.

4. Statement 객체를 생성 및 질의 수행

5. SQL문에 결과물이 있다면 ResultSet 객체를 생성한다.

6. 모든 객체를 닫는다.

**\*RESTAPI**

클라이언트의 종류가 웹 브라우저, 안드로이드 앱, iOS 앱 등 다양해지면서 이러한 클라이언트들에게 정보를 제공하는 방식을 하나로 일원화시키고 싶어졌습니다.

일원화시키는 방식 중에 대표적인 방식이 HTTP프로토콜로 API를 제공하는 것입니다.

HTTP프로토콜로 제공하는 API를 REST API라고 합니다.

REST는 다음과 같은 스타일을 반드시 지켜야 한다고 말합니다.

* -client-server
* -stateless
* -cache
* -uniform interface
* -layered system
* -code-on-demand (optional)

여기서 스타일이란 제약조건의 집합을 의미합니다.

즉, 위에서 언급한 내용을 잘 지켜야만 REST라고 말할 수 있다는 의미입니다.

HTTP프로토콜을 사용한다면 client-server, stateless, cache, lared system, code-on-demand 등에 대해서는 모두 쉽게 구현 가능합니다.

하지만, 문제는 uniform interface입니다.

uniform interface의 스타일

* 리소스가 URI로 식별되야 합니다.
* 리소스를 생성,수정,추가하고자 할 때 HTTP메시지에 표현을 해서 전송해야 합니다.
* 메시지는 스스로 설명할 수 있어야 합니다. (Self-descriptive message)
* 애플리케이션의 상태는 Hyperlink를 이용해 전이되야 합니다.(HATEOAS)

**REST API는 쉽지 않다. 그래서, 보통은 Web API(혹은 HTTP API)를 사용한다.**

**\*Web API**

* URI는 정보의 자원을 표현해야 합니다.
* 자원에 대한 행위는 HTTP Method(GET, POST, PUT, DELETE)로 표현합니다.